Module: tf.compat.v1.debugging

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

Public API for tf.debugging namespace.

Functions

[Assert(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/Assert): Asserts that the given condition is true.

[assert\_all\_finite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/verify_tensor_all_finite): Assert that the tensor does not contain any NaN's or Inf's.

[assert\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_equal): Assert the condition x == y holds element-wise.

[assert\_greater(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_greater): Assert the condition x > y holds element-wise.

[assert\_greater\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_greater_equal): Assert the condition x >= y holds element-wise.

[assert\_integer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_integer): Assert that x is of integer dtype.

[assert\_less(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_less): Assert the condition x < y holds element-wise.

[assert\_less\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_less_equal): Assert the condition x <= y holds element-wise.

[assert\_near(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_near): Assert the condition x and y are close element-wise.

[assert\_negative(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_negative): Assert the condition x < 0 holds element-wise.

[assert\_non\_negative(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_non_negative): Assert the condition x >= 0 holds element-wise.

[assert\_non\_positive(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_non_positive): Assert the condition x <= 0 holds element-wise.

[assert\_none\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_none_equal): Assert the condition x != y holds for all elements.

[assert\_positive(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_positive): Assert the condition x > 0 holds element-wise.

[assert\_proper\_iterable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/assert_proper_iterable): Static assert that values is a "proper" iterable.

[assert\_rank(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_rank): Assert x has rank equal to rank.

[assert\_rank\_at\_least(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_rank_at_least): Assert x has rank equal to rank or higher.

[assert\_rank\_in(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_rank_in): Assert x has rank in ranks.

[assert\_same\_float\_dtype(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/assert_same_float_dtype): Validate and return float type based on tensors and dtype.

[assert\_scalar(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_scalar): Asserts that the given tensor is a scalar (i.e. zero-dimensional).

[assert\_shapes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/debugging/assert_shapes): Assert tensor shapes and dimension size relationships between tensors.

[assert\_type(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/assert_type): Statically asserts that the given Tensor is of the specified type.

[check\_numerics(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/check_numerics): Checks a tensor for NaN and Inf values.

[get\_log\_device\_placement(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/get_log_device_placement): Get if device placements are logged.

[is\_finite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_finite): Returns which elements of x are finite.

[is\_inf(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_inf): Returns which elements of x are Inf.

[is\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_nan): Returns which elements of x are NaN.

[is\_non\_decreasing(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_non_decreasing): Returns True if x is non-decreasing.

[is\_numeric\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/is_numeric_tensor): Returns True if the elements of tensor are numbers.

[is\_strictly\_increasing(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_strictly_increasing): Returns True if x is strictly increasing.

[set\_log\_device\_placement(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/set_log_device_placement): Set if device placements should be logged.

# tf.compat.v1.debugging.assert\_shapes

Assert tensor shapes and dimension size relationships between tensors.

tf.compat.v1.debugging.assert\_shapes(  
    shapes,  
    data=None,  
    summarize=None,  
    message=None,  
    name=None  
)

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

This Op checks that a collection of tensors shape relationships satisfies given constraints.

#### Example:

tf.assert\_shapes({  
  x: ('N', 'Q'),  
  y: ('N', 'D'),  
  param: ('Q',),  
  scalar: ()  
})

Example of adding a dependency to an operation:

with tf.control\_dependencies([tf.assert\_shapes(shapes)]):  
  output = tf.matmul(x, y, transpose\_a=True)

If x, y, param or scalar does not have a shape that satisfies all specified constraints, message, as well as the first summarize entries of the first encountered violating tensor are printed, andInvalidArgumentError is raised.

Size entries in the specified shapes are checked against other entries by their **hash**, except: - a size entry is interpreted as an explicit size if it can be parsed as an integer primitive. - a size entry is interpreted as any size if it is None or '.'.

If the first entry of a shape is ... (type Ellipsis) or '\*' that indicates a variable number of outer dimensions of unspecified size, i.e. the constraint applies to the inner-most dimensions only.

Scalar tensors and specified shapes of length zero (excluding the 'inner-most' prefix) are both treated as having a single dimension of size one.

#### Args:

* **shapes**: dictionary with (Tensor to shape) items. A shape must be an iterable.
* **data**: The tensors to print out if the condition is False. Defaults to error message and first few entries of the violating tensor.
* **summarize**: Print this many entries of the tensor.
* **message**: A string to prefix to the default message.
* **name**: A name for this operation (optional). Defaults to "assert\_shapes".

#### Returns:

Op raising InvalidArgumentError unless all shape constraints are satisfied. If static checks determine all constraints are satisfied, a no\_op is returned.

#### Raises:

* **ValueError**: If static checks determine any shape constraint is violated.

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)

Module: tf.compat.v1.train / tf.train

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

Support for training models.

See the [Training](https://tensorflow.org/api_guides/python/train) guide.

Modules

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

[queue\_runner](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/queue_runner) module: Public API for tf.train.queue\_runner namespace.

Classes

[class AdadeltaOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer): Optimizer that implements the Adadelta algorithm.

[class AdagradDAOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer): Adagrad Dual Averaging algorithm for sparse linear models.

[class AdagradOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer): Optimizer that implements the Adagrad algorithm.

[class AdamOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer): Optimizer that implements the Adam algorithm.

[class BytesList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList)

[class Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint): Groups trackable objects, saving and restoring them.

[class CheckpointManager](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager): Deletes old checkpoints.

[class CheckpointSaverHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/CheckpointSaverHook): Saves checkpoints every N steps or seconds.

[class CheckpointSaverListener](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/CheckpointSaverListener): Interface for listeners that take action before or after checkpoint save.

[class ChiefSessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator): Creates a tf.compat.v1.Session for a chief.

[class ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef)

[class ClusterSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec): Represents a cluster as a set of "tasks", organized into "jobs".

[class Coordinator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator): A coordinator for threads.

[class Example](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example)

[class ExponentialMovingAverage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage): Maintains moving averages of variables by employing an exponential decay.

[class Feature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature)

[class FeatureList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList)

[class FeatureLists](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists)

[class Features](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features)

[class FeedFnHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/FeedFnHook): Runs feed\_fn and sets the feed\_dict accordingly.

[class FinalOpsHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/FinalOpsHook): A hook which evaluates Tensors at the end of a session.

[class FloatList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList)

[class FtrlOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer): Optimizer that implements the FTRL algorithm.

[class GlobalStepWaiterHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/GlobalStepWaiterHook): Delays execution until global step reaches wait\_until\_step.

[class GradientDescentOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer): Optimizer that implements the gradient descent algorithm.

[class Int64List](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List)

[class JobDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef)

[class LoggingTensorHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/LoggingTensorHook): Prints the given tensors every N local steps, every N seconds, or at end.

[class LooperThread](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread): A thread that runs code repeatedly, optionally on a timer.

[class MomentumOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer): Optimizer that implements the Momentum algorithm.

[class MonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession): Session-like object that handles initialization, recovery and hooks.

[class NanLossDuringTrainingError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/NanLossDuringTrainingError)

[class NanTensorHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/NanTensorHook): Monitors the loss tensor and stops training if loss is NaN.

[class Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer): Base class for optimizers.

[class ProfilerHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/ProfilerHook): Captures CPU/GPU profiling information every N steps or seconds.

[class ProximalAdagradOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer): Optimizer that implements the Proximal Adagrad algorithm.

[class ProximalGradientDescentOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer): Optimizer that implements the proximal gradient descent algorithm.

[class QueueRunner](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner): Holds a list of enqueue operations for a queue, each to be run in a thread.

[class RMSPropOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer): Optimizer that implements the RMSProp algorithm.

[class Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver): Saves and restores variables.

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

[class Scaffold](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold): Structure to create or gather pieces commonly needed to train a model.

[class SecondOrStepTimer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SecondOrStepTimer): Timer that triggers at most once every N seconds or once every N steps.

[class SequenceExample](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample)

[class Server](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Server): An in-process TensorFlow server, for use in distributed training.

[class ServerDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef)

[class SessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator): A factory for tf.Session.

[class SessionManager](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager): Training helper that restores from checkpoint and creates session.

[class SessionRunArgs](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SessionRunArgs): Represents arguments to be added to a Session.run() call.

[class SessionRunContext](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SessionRunContext): Provides information about the session.run() call being made.

[class SessionRunHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SessionRunHook): Hook to extend calls to MonitoredSession.run().

[class SessionRunValues](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SessionRunValues): Contains the results of Session.run().

[class SingularMonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession): Session-like object that handles initialization, restoring, and hooks.

[class StepCounterHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/StepCounterHook): Hook that counts steps per second.

[class StopAtStepHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/StopAtStepHook): Hook that requests stop at a specified step.

[class SummarySaverHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/SummarySaverHook): Saves summaries every N steps.

[class Supervisor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor): A training helper that checkpoints models and computes summaries.

[class SyncReplicasOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer): Class to synchronize, aggregate gradients and pass them to the optimizer.

[class VocabInfo](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/VocabInfo): Vocabulary information for warm-starting.

[class WorkerSessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator): Creates a tf.compat.v1.Session for a worker.

Functions

[MonitoredTrainingSession(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredTrainingSession): Creates a MonitoredSession for training.

[NewCheckpointReader(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/NewCheckpointReader)

[add\_queue\_runner(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/add_queue_runner): Adds a QueueRunner to a collection in the graph. (deprecated)

[assert\_global\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/assert_global_step): Asserts global\_step\_tensor is a scalar int Variable or Tensor.

[basic\_train\_loop(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/basic_train_loop): Basic loop to train a model.

[batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/batch): Creates batches of tensors in tensors. (deprecated)

[batch\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/batch_join): Runs a list of tensors to fill a queue to create batches of examples. (deprecated)

[checkpoint\_exists(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/checkpoint_exists): Checks whether a V1 or V2 checkpoint exists with the specified prefix. (deprecated)

[checkpoints\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/checkpoints_iterator): Continuously yield new checkpoint files as they appear.

[cosine\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/cosine_decay): Applies cosine decay to the learning rate.

[cosine\_decay\_restarts(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/cosine_decay_restarts): Applies cosine decay with restarts to the learning rate.

[create\_global\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/create_global_step): Create global step tensor in graph.

[do\_quantize\_training\_on\_graphdef(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/do_quantize_training_on_graphdef): A general quantization scheme is being developed in tf.contrib.quantize. (deprecated)

[exponential\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/exponential_decay): Applies exponential decay to the learning rate.

[export\_meta\_graph(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/export_meta_graph): Returns MetaGraphDef proto.

[generate\_checkpoint\_state\_proto(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/generate_checkpoint_state_proto): Generates a checkpoint state proto.

[get\_checkpoint\_mtimes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_checkpoint_mtimes): Returns the mtimes (modification timestamps) of the checkpoints. (deprecated)

[get\_checkpoint\_state(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/get_checkpoint_state): Returns CheckpointState proto from the "checkpoint" file.

[get\_global\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_global_step): Get the global step tensor.

[get\_or\_create\_global\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_or_create_global_step): Returns and create (if necessary) the global step tensor.

[global\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/global_step): Small helper to get the global step.

[import\_meta\_graph(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/import_meta_graph): Recreates a Graph saved in a MetaGraphDef proto.

[init\_from\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/init_from_checkpoint): Replaces [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) initializers so they load from a checkpoint file.

[input\_producer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/input_producer): Output the rows of input\_tensor to a queue for an input pipeline. (deprecated)

[inverse\_time\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/inverse_time_decay): Applies inverse time decay to the initial learning rate.

[latest\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint): Finds the filename of latest saved checkpoint file.

[limit\_epochs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/limit_epochs): Returns tensor num\_epochs times and then raises an OutOfRange error. (deprecated)

[linear\_cosine\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/linear_cosine_decay): Applies linear cosine decay to the learning rate.

[list\_variables(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/list_variables): Returns list of all variables in the checkpoint.

[load\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/load_checkpoint): Returns CheckpointReader for checkpoint found in ckpt\_dir\_or\_file.

[load\_variable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/load_variable): Returns the tensor value of the given variable in the checkpoint.

[match\_filenames\_once(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/match_filenames_once): Save the list of files matching pattern, so it is only computed once.

[maybe\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/maybe_batch): Conditionally creates batches of tensors based on keep\_input. (deprecated)

[maybe\_batch\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/maybe_batch_join): Runs a list of tensors to conditionally fill a queue to create batches. (deprecated)

[maybe\_shuffle\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/maybe_shuffle_batch): Creates batches by randomly shuffling conditionally-enqueued tensors. (deprecated)

[maybe\_shuffle\_batch\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/maybe_shuffle_batch_join): Create batches by randomly shuffling conditionally-enqueued tensors. (deprecated)

[natural\_exp\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/natural_exp_decay): Applies natural exponential decay to the initial learning rate.

[noisy\_linear\_cosine\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/noisy_linear_cosine_decay): Applies noisy linear cosine decay to the learning rate.

[piecewise\_constant(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/piecewise_constant): Piecewise constant from boundaries and interval values.

[piecewise\_constant\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/piecewise_constant): Piecewise constant from boundaries and interval values.

[polynomial\_decay(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/polynomial_decay): Applies a polynomial decay to the learning rate.

[range\_input\_producer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/range_input_producer): Produces the integers from 0 to limit-1 in a queue. (deprecated)

[remove\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/remove_checkpoint): Removes a checkpoint given by checkpoint\_prefix. (deprecated)

[replica\_device\_setter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/replica_device_setter): Return a device function to use when building a Graph for replicas.

[sdca\_fprint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/sdca_fprint): Computes fingerprints of the input strings.

[sdca\_optimizer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/sdca_optimizer): Distributed version of Stochastic Dual Coordinate Ascent (SDCA) optimizer for

[sdca\_shrink\_l1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/sdca_shrink_l1): Applies L1 regularization shrink step on the parameters.

[shuffle\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/shuffle_batch): Creates batches by randomly shuffling tensors. (deprecated)

[shuffle\_batch\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/shuffle_batch_join): Create batches by randomly shuffling tensors. (deprecated)

[slice\_input\_producer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/slice_input_producer): Produces a slice of each Tensor in tensor\_list. (deprecated)

[start\_queue\_runners(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/start_queue_runners): Starts all queue runners collected in the graph. (deprecated)

[string\_input\_producer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/string_input_producer): Output strings (e.g. filenames) to a queue for an input pipeline. (deprecated)

[summary\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/summary_iterator): An iterator for reading Event protocol buffers from an event file.

[update\_checkpoint\_state(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/update_checkpoint_state): Updates the content of the 'checkpoint' file. (deprecated)

[warm\_start(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/warm_start): Warm-starts a model using the given settings.

[write\_graph(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/write_graph): Writes a graph proto to a file.

# tf.compat.v1.train.AdadeltaOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer#top_of_page)
* [Class AdadeltaOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer#class_adadeltaoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdadeltaOptimizer#apply_gradients)

## Class AdadeltaOptimizer

Optimizer that implements the Adadelta algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

See [M. D. Zeiler](http://arxiv.org/abs/1212.5701) ([pdf](http://arxiv.org/pdf/1212.5701v1.pdf))

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    rho=0.95,  
    epsilon=1e-08,  
    use\_locking=False,  
    name='Adadelta'  
)

Construct a new Adadelta optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate. To match the exact form in the original paper use 1.0.
* **rho**: A Tensor or a floating point value. The decay rate.
* **epsilon**: A Tensor or a floating point value. A constant epsilon used to better conditioning the grad update.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Adadelta".

#### Eager Compatibility

When eager execution is enabled, learning\_rate, rho, and epsilon can each be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.AdagradDAOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer#top_of_page)
* [Class AdagradDAOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer#class_adagraddaoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradDAOptimizer#apply_gradients)

## Class AdagradDAOptimizer

Adagrad Dual Averaging algorithm for sparse linear models.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/training/adagrad\_da.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/adagrad_da.py).

See this [paper](http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf).

This optimizer takes care of regularization of unseen features in a mini batch by updating them when they are seen with a closed form update rule that is equivalent to having updated them on every mini-batch.

AdagradDA is typically used when there is a need for large sparsity in the trained model. This optimizer only guarantees sparsity for linear models. Be careful when using AdagradDA for deep networks as it will require careful initialization of the gradient accumulators for it to train.

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    global\_step,  
    initial\_gradient\_squared\_accumulator\_value=0.1,  
    l1\_regularization\_strength=0.0,  
    l2\_regularization\_strength=0.0,  
    use\_locking=False,  
    name='AdagradDA'  
)

Construct a new AdagradDA optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **global\_step**: A Tensor containing the current training step number.
* **initial\_gradient\_squared\_accumulator\_value**: A floating point value. Starting value for the accumulators, must be positive.
* **l1\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **l2\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "AdagradDA".

#### Raises:

* **ValueError**: If the initial\_gradient\_squared\_accumulator\_value is invalid.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.AdagradOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer#top_of_page)
* [Class AdagradOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer#class_adagradoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdagradOptimizer#apply_gradients)

## Class AdagradOptimizer

Optimizer that implements the Adagrad algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

See this [paper](http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf) or this [intro](https://ppasupat.github.io/a9online/uploads/proximal_notes.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    initial\_accumulator\_value=0.1,  
    use\_locking=False,  
    name='Adagrad'  
)

Construct a new Adagrad optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **initial\_accumulator\_value**: A floating point value. Starting value for the accumulators, must be positive.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Adagrad".

#### Raises:

* **ValueError**: If the initial\_accumulator\_value is invalid.

#### Eager Compatibility

When eager execution is enabled, learning\_rate can be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.AdamOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer#top_of_page)
* [Class AdamOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer#class_adamoptimizer)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/AdamOptimizer#methods)

## Class AdamOptimizer

Optimizer that implements the Adam algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)

See [Kingma et al., 2014](http://arxiv.org/abs/1412.6980) ([pdf](http://arxiv.org/pdf/1412.6980.pdf)).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate=0.001,  
    beta1=0.9,  
    beta2=0.999,  
    epsilon=1e-08,  
    use\_locking=False,  
    name='Adam'  
)

Construct a new Adam optimizer.

#### Initialization:

m0:=0(Initialize initial 1st moment vector)

v0:=0(Initialize initial 2nd moment vector)

t:=0(Initialize timestep)

The update rule for variable with gradient g uses an optimization described at the end of section 2 of the paper:

t:=t+1

lrt:=learning\\_rate∗1−beta2t/(1−beta1t)

mt:=beta1∗mt−1+(1−beta1)∗g

vt:=beta2∗vt−1+(1−beta2)∗g∗g

variable:=variable−lrt∗mt/(vt+ϵ)

The default value of 1e-8 for epsilon might not be a good default in general. For example, when training an Inception network on ImageNet a current good choice is 1.0 or 0.1. Note that since AdamOptimizer uses the formulation just before Section 2.1 of the Kingma and Ba paper rather than the formulation in Algorithm 1, the "epsilon" referred to here is "epsilon hat" in the paper.

The sparse implementation of this algorithm (used when the gradient is an IndexedSlices object, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding lookup in the forward pass) does apply momentum to variable slices even if they were not used in the forward pass (meaning they have a gradient equal to zero). Momentum decay (beta1) is also applied to the entire momentum accumulator. This means that the sparse behavior is equivalent to the dense behavior (in contrast to some momentum implementations which ignore momentum unless a variable slice was actually used).

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **beta1**: A float value or a constant float tensor. The exponential decay rate for the 1st moment estimates.
* **beta2**: A float value or a constant float tensor. The exponential decay rate for the 2nd moment estimates.
* **epsilon**: A small constant for numerical stability. This epsilon is "epsilon hat" in the Kingma and Ba paper (in the formula just before Section 2.1), not the epsilon in Algorithm 1 of the paper.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name for the operations created when applying gradients. Defaults to "Adam". @compatibility(eager) When eager execution is enabled, learning\_rate, beta1, beta2, and epsilon can each be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions. @end\_compatibility

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.add\_queue\_runner

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/add_queue_runner#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/add_queue_runner#aliases)

Adds a QueueRunner to a collection in the graph. (deprecated)

### Aliases:

* tf.compat.v1.train.add\_queue\_runner
* tf.compat.v1.train.queue\_runner.add\_queue\_runner

tf.compat.v1.train.add\_queue\_runner(  
    qr,  
    collection=tf.GraphKeys.QUEUE\_RUNNERS  
)

Defined in [python/training/queue\_runner\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/queue_runner_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: To construct input pipelines, use the [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) module.

When building a complex model that uses many queues it is often difficult to gather all the queue runners that need to be run. This convenience function allows you to add a queue runner to a well known collection in the graph.

The companion method start\_queue\_runners() can be used to start threads for all the collected queue runners.

#### Args:

* **qr**: A QueueRunner.
* **collection**: A GraphKey specifying the graph collection to add the queue runner to. Defaults to GraphKeys.QUEUE\_RUNNERS.

# tf.compat.v1.train.assert\_global\_step

Asserts global\_step\_tensor is a scalar int Variable or Tensor.

tf.compat.v1.train.assert\_global\_step(global\_step\_tensor)

Defined in [python/training/training\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/training_util.py).

#### Args:

* **global\_step\_tensor**: Tensor to test.

# tf.compat.v1.train.basic\_train\_loop

Basic loop to train a model.

tf.compat.v1.train.basic\_train\_loop(  
    supervisor,  
    train\_step\_fn,  
    args=None,  
    kwargs=None,  
    master=''  
)

Defined in [python/training/basic\_loops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/basic_loops.py).

Calls train\_step\_fn in a loop to train a model. The function is called as:

train\_step\_fn(session, \*args, \*\*kwargs)

It is passed a [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) in addition to args and kwargs. The function typically runs one training step in the session.

#### Args:

* **supervisor**: [tf.compat.v1.train.Supervisor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor) to run the training services.
* **train\_step\_fn**: Callable to execute one training step. Called repeatedly astrain\_step\_fn(session, \*args \*\*kwargs).
* **args**: Optional positional arguments passed to train\_step\_fn.
* **kwargs**: Optional keyword arguments passed to train\_step\_fn.
* **master**: Master to use to create the training session. Defaults to "" which causes the session to be created in the local process.

# tf.compat.v1.train.batch

Creates batches of tensors in tensors. (deprecated)

tf.compat.v1.train.batch(  
    tensors,  
    batch\_size,  
    num\_threads=1,  
    capacity=32,  
    enqueue\_many=False,  
    shapes=None,  
    dynamic\_pad=False,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.batch(batch\_size)** (or **padded\_batch(...)** if **dynamic\_pad=True**).

The argument tensors can be a list or a dictionary of tensors. The value returned by the function will be of the same type as tensors.

This function is implemented using a queue. A QueueRunner for the queue is added to the current Graph's QUEUE\_RUNNER collection.

If enqueue\_many is False, tensors is assumed to represent a single example. An input tensor with shape [x, y, z] will be output as a tensor with shape [batch\_size, x, y, z].

If enqueue\_many is True, tensors is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of tensors should have the same size in the first dimension. If an input tensor has shape [\*, x, y, z], the output will have shape [batch\_size, x, y, z]. The capacity argument controls the how long the prefetching is allowed to grow the queues.

The returned operation is a dequeue operation and will throw [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError) if the input queue is exhausted. If this operation is feeding another input queue, its queue runner will catch this exception, however, if this operation is used in your main thread you are responsible for catching this yourself.

N.B.: If dynamic\_pad is False, you must ensure that either (i) the shapes argument is passed, or (ii) all of the tensors in tensors must have fully-defined shapes. ValueError will be raised if neither of these conditions holds.

If dynamic\_pad is True, it is sufficient that the rank of the tensors is known, but individual dimensions may have shape None. In this case, for each enqueue the dimensions with value Nonemay have a variable length; upon dequeue, the output tensors will be padded on the right to the maximum shape of the tensors in the current minibatch. For numbers, this padding takes value 0. For strings, this padding is the empty string. See PaddingFIFOQueue for more info.

If allow\_smaller\_final\_batch is True, a smaller batch value than batch\_size is returned when the queue is closed and there are not enough elements to fill the batch, otherwise the pending elements are discarded. In addition, all output tensors' static shapes, as accessed via the shapeproperty will have a first Dimension value of None, and operations that depend on fixed batch\_size would fail.

#### Args:

* **tensors**: The list or dictionary of tensors to enqueue.
* **batch\_size**: The new batch size pulled from the queue.
* **num\_threads**: The number of threads enqueuing tensors. The batching will be nondeterministic if num\_threads > 1.
* **capacity**: An integer. The maximum number of elements in the queue.
* **enqueue\_many**: Whether each tensor in tensors is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensors.
* **dynamic\_pad**: Boolean. Allow variable dimensions in input shapes. The given dimensions are padded upon dequeue so that tensors within a batch have the same shapes.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same types as tensors (except if the input is a list of one element, then it returns a tensor, not a list).

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.batch\_join

Runs a list of tensors to fill a queue to create batches of examples. (deprecated)

tf.compat.v1.train.batch\_join(  
    tensors\_list,  
    batch\_size,  
    capacity=32,  
    enqueue\_many=False,  
    shapes=None,  
    dynamic\_pad=False,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.interleave(...).batch(batch\_size)** (or **padded\_batch(...)** if **dynamic\_pad=True**).

The tensors\_list argument is a list of tuples of tensors, or a list of dictionaries of tensors. Each element in the list is treated similarly to the tensors argument of tf.compat.v1.train.batch().

WARNING: This function is nondeterministic, since it starts a separate thread for each tensor.

Enqueues a different list of tensors in different threads. Implemented using a queue -- a QueueRunnerfor the queue is added to the current Graph's QUEUE\_RUNNER collection.

len(tensors\_list) threads will be started, with thread i enqueuing the tensors fromtensors\_list[i]. tensors\_list[i1][j] must match tensors\_list[i2][j] in type and shape, except in the first dimension if enqueue\_many is true.

If enqueue\_many is False, each tensors\_list[i] is assumed to represent a single example. An input tensor x will be output as a tensor with shape [batch\_size] + x.shape.

If enqueue\_many is True, tensors\_list[i] is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of tensors\_list[i] should have the same size in the first dimension. The slices of any input tensor x are treated as examples, and the output tensors will have shape [batch\_size] + x.shape[1:].

The capacity argument controls the how long the prefetching is allowed to grow the queues.

The returned operation is a dequeue operation and will throw [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError) if the input queue is exhausted. If this operation is feeding another input queue, its queue runner will catch this exception, however, if this operation is used in your main thread you are responsible for catching this yourself.

N.B.: If dynamic\_pad is False, you must ensure that either (i) the shapes argument is passed, or (ii) all of the tensors in tensors\_list must have fully-defined shapes. ValueError will be raised if neither of these conditions holds.

If dynamic\_pad is True, it is sufficient that the rank of the tensors is known, but individual dimensions may have value None. In this case, for each enqueue the dimensions with value Nonemay have a variable length; upon dequeue, the output tensors will be padded on the right to the maximum shape of the tensors in the current minibatch. For numbers, this padding takes value 0. For strings, this padding is the empty string. See PaddingFIFOQueue for more info.

If allow\_smaller\_final\_batch is True, a smaller batch value than batch\_size is returned when the queue is closed and there are not enough elements to fill the batch, otherwise the pending elements are discarded. In addition, all output tensors' static shapes, as accessed via the shapeproperty will have a first Dimension value of None, and operations that depend on fixed batch\_size would fail.

#### Args:

* **tensors\_list**: A list of tuples or dictionaries of tensors to enqueue.
* **batch\_size**: An integer. The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensor\_list\_list[i].
* **dynamic\_pad**: Boolean. Allow variable dimensions in input shapes. The given dimensions are padded upon dequeue so that tensors within a batch have the same shapes.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional) If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same number and types as tensors\_list[i].

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensor\_list\_list.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.Checkpoint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint#top_of_page)
* [Class Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint#class_checkpoint)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint#properties)
  + [save\_counter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Checkpoint#save_counter)

## Class Checkpoint

Groups trackable objects, saving and restoring them.

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

Checkpoint's constructor accepts keyword arguments whose values are types that contain trackable state, such as [tf.compat.v1.train.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer) implementations, [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable), tf.keras.Layerimplementations, or [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) implementations. It saves these values with a checkpoint, and maintains a save\_counter for numbering checkpoints.

Example usage when graph building:

import tensorflow as tf  
import os  
  
checkpoint\_directory = "/tmp/training\_checkpoints"  
checkpoint\_prefix = os.path.join(checkpoint\_directory, "ckpt")  
  
checkpoint = tf.train.Checkpoint(optimizer=optimizer, model=model)  
status = checkpoint.restore(tf.train.latest\_checkpoint(checkpoint\_directory))  
train\_op = optimizer.minimize( ... )  
status.assert\_consumed()  # Optional sanity checks.  
with tf.compat.v1.Session() as session:  
  # Use the Session to restore variables, or initialize them if  
  # tf.train.latest\_checkpoint returned None.  
  status.initialize\_or\_restore(session)  
  for \_ in range(num\_training\_steps):  
    session.run(train\_op)  
  checkpoint.save(file\_prefix=checkpoint\_prefix)

Example usage with eager execution enabled:

import tensorflow as tf  
import os  
  
tf.compat.v1.enable\_eager\_execution()  
  
checkpoint\_directory = "/tmp/training\_checkpoints"  
checkpoint\_prefix = os.path.join(checkpoint\_directory, "ckpt")  
  
checkpoint = tf.train.Checkpoint(optimizer=optimizer, model=model)  
status = checkpoint.restore(tf.train.latest\_checkpoint(checkpoint\_directory))  
for \_ in range(num\_training\_steps):  
  optimizer.minimize( ... )  # Variables will be restored on creation.  
status.assert\_consumed()  # Optional sanity checks.  
checkpoint.save(file\_prefix=checkpoint\_prefix)

Checkpoint.save and Checkpoint.restore write and read object-based checkpoints, in contrast to [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) which writes and reads variable.name based checkpoints. Object-based checkpointing saves a graph of dependencies between Python objects (Layers, Optimizers, Variables, etc.) with named edges, and this graph is used to match variables when restoring a checkpoint. It can be more robust to changes in the Python program, and helps to support restore-on-create for variables when executing eagerly. Prefer [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) over [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) for new code.

Checkpoint objects have dependencies on the objects passed as keyword arguments to their constructors, and each dependency is given a name that is identical to the name of the keyword argument for which it was created. TensorFlow classes like Layers and Optimizers will automatically add dependencies on their variables (e.g. "kernel" and "bias" for[tf.keras.layers.Dense](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense)). Inheriting from [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) makes managing dependencies easy in user-defined classes, since Model hooks into attribute assignment. For example:

class Regress(tf.keras.Model):  
  
  def \_\_init\_\_(self):  
    super(Regress, self).\_\_init\_\_()  
    self.input\_transform = tf.keras.layers.Dense(10)  
    # ...  
  
  def call(self, inputs):  
    x = self.input\_transform(inputs)  
    # ...

This Model has a dependency named "input\_transform" on its Dense layer, which in turn depends on its variables. As a result, saving an instance of Regress using [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) will also save all the variables created by the Dense layer.

When variables are assigned to multiple workers, each worker writes its own section of the checkpoint. These sections are then merged/re-indexed to behave as a single checkpoint. This avoids copying all variables to one worker, but does require that all workers see a common filesystem.

While [tf.keras.Model.save\_weights](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#save_weights) and [tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) save in the same format, note that the root of the resulting checkpoint is the object the save method is attached to. This means saving a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) using save\_weights and loading into a [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) with a Model attached (or vice versa) will not match the Model's variables. See the [guide to training checkpoints](https://www.tensorflow.org/alpha/guide/checkpoints) for details. Prefer [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) over [tf.keras.Model.save\_weights](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#save_weights) for training checkpoints.

#### Attributes:

* **save\_counter**: Incremented when save() is called. Used to number checkpoints.

## \_\_init\_\_

\_\_init\_\_(\*\*kwargs)

Group objects into a training checkpoint.

#### Args:

* **\*\*kwargs**: Keyword arguments are set as attributes of this object, and are saved with the checkpoint. Values must be trackable objects.

#### Raises:

* **ValueError**: If objects in kwargs are not trackable.

## Properties

### save\_counter

An integer variable which starts at zero and is incremented on save.

Used to number checkpoints.

#### Returns:

The save counter variable.

## Methods

### restore

restore(save\_path)

Restore a training checkpoint.

Restores this Checkpoint and any objects it depends on.

When executing eagerly, either assigns values immediately if variables to restore have been created already, or defers restoration until the variables are created. Dependencies added after this call will be matched if they have a corresponding object in the checkpoint (the restore request will queue in any trackable object waiting for the expected dependency to be added).

When graph building, restoration ops are added to the graph but not run immediately.

To ensure that loading is complete and no more assignments will take place, use the assert\_consumed() method of the status object returned by restore:

checkpoint = tf.train.Checkpoint( ... )  
checkpoint.restore(path).assert\_consumed()

An exception will be raised if any Python objects in the dependency graph were not found in the checkpoint, or if any checkpointed values do not have a matching Python object.

When graph building, assert\_consumed() indicates that all of the restore ops that will be created for this checkpoint have been created. They can be run via the run\_restore\_ops() method of the status object:

checkpoint.restore(path).assert\_consumed().run\_restore\_ops()

If the checkpoint has not been consumed completely, then the list of restore ops will grow as more objects are added to the dependency graph.

Name-based [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) checkpoints can be loaded using this method. Names are used to match variables. No restore ops are created/run until run\_restore\_ops() or initialize\_or\_restore() are called on the returned status object when graph building, but there is restore-on-creation when executing eagerly. Re-encode name-based checkpoints using[tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) as soon as possible.

#### Args:

* **save\_path**: The path to the checkpoint, as returned by save or[tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). If None (as when there is no latest checkpoint for [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint) to return), returns an object which may run initializers for objects in the dependency graph. If the checkpoint was written by the name-based[tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver), names are used to match variables.

#### Returns:

A load status object, which can be used to make assertions about the status of a checkpoint restoration and run initialization/restore ops.

The returned status object has the following methods:

* assert\_consumed(): Raises an exception if any variables/objects are unmatched: either checkpointed values which don't have a matching Python object or Python objects in the dependency graph with no values in the checkpoint. This method returns the status object, and so may be chained with initialize\_or\_restore or run\_restore\_ops.
* assert\_existing\_objects\_matched(): Raises an exception if any existing Python objects in the dependency graph are unmatched. Unlike assert\_consumed, this assertion will pass if values in the checkpoint have no corresponding Python objects. For example a tf.keras.Layer object which has not yet been built, and so has not created any variables, will pass this assertion but fail assert\_consumed. Useful when loading part of a larger checkpoint into a new Python program, e.g. a training checkpoint with a [tf.compat.v1.train.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)was saved but only the state required for inference is being loaded. This method returns the status object, and so may be chained with initialize\_or\_restore or run\_restore\_ops.
* assert\_nontrivial\_match(): Asserts that something aside from the root object was matched. This is a very weak assertion, but is useful for sanity checking in library code where objects may exist in the checkpoint which haven't been created in Python and some Python objects may not have a checkpointed value.
* expect\_partial(): Silence warnings about incomplete checkpoint restores. Warnings are otherwise printed for unused parts of the checkpoint file or object when the Checkpoint object is deleted (often at program shutdown).
* initialize\_or\_restore(session=None): When graph building, runs variable initializers if save\_path is None, but otherwise runs restore operations. If no session is explicitly specified, the default session is used. No effect when executing eagerly (variables are initialized or restored eagerly).
* run\_restore\_ops(session=None): When graph building, runs restore operations. If no session is explicitly specified, the default session is used. No effect when executing eagerly (restore operations are run eagerly). May only be called when save\_path is not None.

### save

save(  
    file\_prefix,  
    session=None  
)

Saves a training checkpoint and provides basic checkpoint management.

The saved checkpoint includes variables created by this object and any trackable objects it depends on at the time Checkpoint.save() is called.

save is a basic convenience wrapper around the write method, sequentially numbering checkpoints using save\_counter and updating the metadata used by [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). More advanced checkpoint management, for example garbage collection and custom numbering, may be provided by other utilities which also wrap write (tf.contrib.checkpoint.CheckpointManagerfor example).

#### Args:

* **file\_prefix**: A prefix to use for the checkpoint filenames (/path/to/directory/and\_a\_prefix). Names are generated based on this prefix and Checkpoint.save\_counter.
* **session**: The session to evaluate variables in. Ignored when executing eagerly. If not provided when graph building, the default session is used.

#### Returns:

The full path to the checkpoint.

### write

write(  
    file\_prefix,  
    session=None  
)

Writes a training checkpoint.

The checkpoint includes variables created by this object and any trackable objects it depends on at the time Checkpoint.write() is called.

write does not number checkpoints, increment save\_counter, or update the metadata used by [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). It is primarily intended for use by higher level checkpoint management utilities. save provides a very basic implementation of these features.

#### Args:

* **file\_prefix**: A prefix to use for the checkpoint filenames (/path/to/directory/and\_a\_prefix).
* **session**: The session to evaluate variables in. Ignored when executing eagerly. If not provided when graph building, the default session is used.

#### Returns:

The full path to the checkpoint (i.e. file\_prefix).

# tf.compat.v1.train.checkpoint\_exists

Checks whether a V1 or V2 checkpoint exists with the specified prefix. (deprecated)

tf.compat.v1.train.checkpoint\_exists(checkpoint\_prefix)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use standard file APIs to check for files with this prefix.

This is the recommended way to check if a checkpoint exists, since it takes into account the naming difference between V1 and V2 formats.

#### Args:

* **checkpoint\_prefix**: the prefix of a V1 or V2 checkpoint, with V2 taking priority. Typically the result of Saver.save() or that of tf.train.latest\_checkpoint(), regardless of sharded/non-sharded or V1/V2.

#### Returns:

A bool, true iff a checkpoint referred to by checkpoint\_prefix exists.

# tf.compat.v1.train.ChiefSessionCreator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator#top_of_page)
* [Class ChiefSessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator#class_chiefsessioncreator)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator#methods)
  + [create\_session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ChiefSessionCreator#create_session)

## Class ChiefSessionCreator

Creates a tf.compat.v1.Session for a chief.

Inherits From: [SessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator)

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

## \_\_init\_\_

\_\_init\_\_(  
    scaffold=None,  
    master='',  
    config=None,  
    checkpoint\_dir=None,  
    checkpoint\_filename\_with\_path=None  
)

Initializes a chief session creator.

#### Args:

* **scaffold**: A Scaffold used for gathering or building supportive ops. If not specified a default one is created. It's used to finalize the graph.
* **master**: String representation of the TensorFlow master to use.
* **config**: ConfigProto proto used to configure the session.
* **checkpoint\_dir**: A string. Optional path to a directory where to restore variables.
* **checkpoint\_filename\_with\_path**: Full file name path to the checkpoint file.

## Methods

### create\_session

create\_session()

# tf.compat.v1.train.cosine\_decay

Applies cosine decay to the learning rate.

tf.compat.v1.train.cosine\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    alpha=0.0,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

See [Loshchilov & Hutter, ICLR2016], SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies a cosine decay function to a provided initial learning rate. It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

global\_step = min(global\_step, decay\_steps)  
cosine\_decay = 0.5 \* (1 + cos(pi \* global\_step / decay\_steps))  
decayed = (1 - alpha) \* cosine\_decay + alpha  
decayed\_learning\_rate = learning\_rate \* decayed

#### Example usage:

decay\_steps = 1000  
lr\_decayed = cosine\_decay(learning\_rate, global\_step, decay\_steps)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **alpha**: A scalar float32 or float64 Tensor or a Python number. Minimum learning rate value as a fraction of learning\_rate.
* **name**: String. Optional name of the operation. Defaults to 'CosineDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.cosine\_decay\_restarts

Applies cosine decay with restarts to the learning rate.

tf.compat.v1.train.cosine\_decay\_restarts(  
    learning\_rate,  
    global\_step,  
    first\_decay\_steps,  
    t\_mul=2.0,  
    m\_mul=1.0,  
    alpha=0.0,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

See [Loshchilov & Hutter, ICLR2016], SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies a cosine decay function with restarts to a provided initial learning rate. It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate while taking into account possible warm restarts. The learning rate multiplier first decays from 1 to alpha for first\_decay\_steps steps. Then, a warm restart is performed. Each new warm restart runs for t\_mul times more steps and with m\_mul times smaller initial learning rate.

#### Example usage:

first\_decay\_steps = 1000  
lr\_decayed = cosine\_decay\_restarts(learning\_rate, global\_step,  
                                   first\_decay\_steps)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation.
* **first\_decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **t\_mul**: A scalar float32 or float64 Tensor or a Python number. Used to derive the number of iterations in the i-th period
* **m\_mul**: A scalar float32 or float64 Tensor or a Python number. Used to derive the initial learning rate of the i-th period:
* **alpha**: A scalar float32 or float64 Tensor or a Python number. Minimum learning rate value as a fraction of the learning\_rate.
* **name**: String. Optional name of the operation. Defaults to 'SGDRDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.create\_global\_step

Create global step tensor in graph.

tf.compat.v1.train.create\_global\_step(graph=None)

Defined in [python/training/training\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/training_util.py).

#### Args:

* **graph**: The graph in which to create the global step tensor. If missing, use default graph.

#### Returns:

Global step tensor.

#### Raises:

* **ValueError**: if global step tensor is already defined.
* tf.compat.v1.train.do\_quantize\_training\_on\_graphdef
* A general quantization scheme is being developed in tf.contrib.quantize. (deprecated)
* tf.compat.v1.train.do\_quantize\_training\_on\_graphdef(  
      input\_graph,  
      num\_bits  
  )
* Defined in [python/pywrap\_tensorflow\_internal.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/pywrap_tensorflow_internal.py).
* **Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: GraphDef quantized training rewriter is deprecated in the long termConsider using that instead, though since it is in the tf.contrib namespace, it is notsubject to backward compatibility guarantees.

tf.compat.v1.train.exponential\_decay

Applies exponential decay to the learning rate.

tf.compat.v1.train.exponential\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    decay\_rate,  
    staircase=False,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies an exponential decay function to a provided initial learning rate. It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

decayed\_learning\_rate = learning\_rate \*  
                        decay\_rate ^ (global\_step / decay\_steps)

If the argument staircase is True, then global\_step / decay\_steps is an integer division and the decayed learning rate follows a staircase function.

Example: decay every 100000 steps with a base of 0.96:

...  
global\_step = tf.Variable(0, trainable=False)  
starter\_learning\_rate = 0.1  
learning\_rate = tf.compat.v1.train.exponential\_decay(starter\_learning\_rate,  
global\_step,  
                                           100000, 0.96, staircase=True)  
# Passing global\_step to minimize() will increment it at each step.  
learning\_step = (  
    tf.compat.v1.train.GradientDescentOptimizer(learning\_rate)  
    .minimize(...my loss..., global\_step=global\_step)  
)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation. Must not be negative.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Must be positive. See the decay computation above.
* **decay\_rate**: A scalar float32 or float64 Tensor or a Python number. The decay rate.
* **staircase**: Boolean. If True decay the learning rate at discrete intervals
* **name**: String. Optional name of the operation. Defaults to 'ExponentialDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.export\_meta\_graph

Returns MetaGraphDef proto.

tf.compat.v1.train.export\_meta\_graph(  
    filename=None,  
    meta\_info\_def=None,  
    graph\_def=None,  
    saver\_def=None,  
    collection\_list=None,  
    as\_text=False,  
    graph=None,  
    export\_scope=None,  
    clear\_devices=False,  
    clear\_extraneous\_savers=False,  
    strip\_default\_attrs=False,  
    save\_debug\_info=False,  
    \*\*kwargs  
)

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

Optionally writes it to filename.

This function exports the graph, saver, and collection objects into MetaGraphDef protocol buffer with the intention of it being imported at a later time or location to restart training, run inference, or be a subgraph.

#### Args:

* **filename**: Optional filename including the path for writing the generated MetaGraphDefprotocol buffer.
* **meta\_info\_def**: MetaInfoDef protocol buffer.
* **graph\_def**: GraphDef protocol buffer.
* **saver\_def**: SaverDef protocol buffer.
* **collection\_list**: List of string keys to collect.
* **as\_text**: If True, writes the MetaGraphDef as an ASCII proto.
* **graph**: The Graph to export. If None, use the default graph.
* **export\_scope**: Optional string. Name scope under which to extract the subgraph. The scope name will be striped from the node definitions for easy import later into new name scopes. If None, the whole graph is exported. graph\_def and export\_scope cannot both be specified.
* **clear\_devices**: Whether or not to clear the device field for an Operation or Tensor during export.
* **clear\_extraneous\_savers**: Remove any Saver-related information from the graph (both Save/Restore ops and SaverDefs) that are not associated with the provided SaverDef.
* **strip\_default\_attrs**: Boolean. If True, default-valued attributes will be removed from the NodeDefs. For a detailed guide, see [Stripping Default-Valued Attributes](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md#stripping-default-valued-attributes).
* **save\_debug\_info**: If True, save the GraphDebugInfo to a separate file, which in the same directory of filename and with \_debug added before the file extend.
* **\*\*kwargs**: Optional keyed arguments.

#### Returns:

A MetaGraphDef proto.

#### Raises:

* **ValueError**: When the GraphDef is larger than 2GB.
* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Exporting/importing meta graphs is not supported unless both graph\_def and graph are provided. No graph exists when eager execution is enabled.

# tf.compat.v1.train.FtrlOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer#top_of_page)
* [Class FtrlOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer#class_ftrloptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/FtrlOptimizer#apply_gradients)

## Class FtrlOptimizer

Optimizer that implements the FTRL algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

See this [paper](https://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf). This version has support for both online L2 (the L2 penalty given in the paper above) and shrinkage-type L2 (which is the addition of an L2 penalty to the loss function).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    learning\_rate\_power=-0.5,  
    initial\_accumulator\_value=0.1,  
    l1\_regularization\_strength=0.0,  
    l2\_regularization\_strength=0.0,  
    use\_locking=False,  
    name='Ftrl',  
    accum\_name=None,  
    linear\_name=None,  
    l2\_shrinkage\_regularization\_strength=0.0  
)

Construct a new FTRL optimizer.

#### Args:

* **learning\_rate**: A float value or a constant float Tensor.
* **learning\_rate\_power**: A float value, must be less or equal to zero. Controls how the learning rate decreases during training. Use zero for a fixed learning rate. See section 3.1 in the [paper](https://www.eecs.tufts.edu/~dsculley/papers/ad-click-prediction.pdf).
* **initial\_accumulator\_value**: The starting value for accumulators. Only zero or positive values are allowed.
* **l1\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **l2\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Ftrl".
* **accum\_name**: The suffix for the variable that keeps the gradient squared accumulator. If not present, defaults to name.
* **linear\_name**: The suffix for the variable that keeps the linear gradient accumulator. If not present, defaults to name + "\_1".
* **l2\_shrinkage\_regularization\_strength**: A float value, must be greater than or equal to zero. This differs from L2 above in that the L2 above is a stabilization penalty, whereas this L2 shrinkage is a magnitude penalty. The FTRL formulation can be written as: w\_{t+1} = argminw(\hat{g}{1:t}w + L1||w||\_1 + L2||w||\_2^2), where \hat{g} = g + (2L2\_shrinkagew), and g is the gradient of the loss function w.r.t. the weights w. Specifically, in the absence of L1 regularization, it is equivalent to the following update rule: w\_{t+1} = w\_t - lr\_t / (1 + 2L2lr\_t) \* g\_t - 2L2\_shrinkagelr\_t / (1 + 2L2lr\_t) \* w\_t where lr\_t is the learning rate at t. When input is sparse shrinkage will only happen on the active weights.

#### Raises:

* **ValueError**: If one of the arguments is invalid.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.generate\_checkpoint\_state\_proto

Generates a checkpoint state proto.

tf.compat.v1.train.generate\_checkpoint\_state\_proto(  
    save\_dir,  
    model\_checkpoint\_path,  
    all\_model\_checkpoint\_paths=None,  
    all\_model\_checkpoint\_timestamps=None,  
    last\_preserved\_timestamp=None  
)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

#### Args:

* **save\_dir**: Directory where the model was saved.
* **model\_checkpoint\_path**: The checkpoint file.
* **all\_model\_checkpoint\_paths**: List of strings. Paths to all not-yet-deleted checkpoints, sorted from oldest to newest. If this is a non-empty list, the last element must be equal to model\_checkpoint\_path. These paths are also saved in the CheckpointState proto.
* **all\_model\_checkpoint\_timestamps**: A list of floats, indicating the number of seconds since the Epoch when each checkpoint was generated.
* **last\_preserved\_timestamp**: A float, indicating the number of seconds since the Epoch when the last preserved checkpoint was written, e.g. due to a keep\_checkpoint\_every\_n\_hoursparameter (see tf.contrib.checkpoint.CheckpointManager for an implementation).

#### Returns:

CheckpointState proto with model\_checkpoint\_path and all\_model\_checkpoint\_paths updated to either absolute paths or relative paths to the current save\_dir.

#### Raises:

* **ValueError**: If all\_model\_checkpoint\_timestamps was provided but its length does not match all\_model\_checkpoint\_paths.

# tf.compat.v1.train.get\_checkpoint\_mtimes

Returns the mtimes (modification timestamps) of the checkpoints. (deprecated)

tf.compat.v1.train.get\_checkpoint\_mtimes(checkpoint\_prefixes)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use standard file utilities to get mtimes.

Globs for the checkpoints pointed to by checkpoint\_prefixes. If the files exist, collect their mtime. Both V2 and V1 checkpoints are considered, in that priority.

This is the recommended way to get the mtimes, since it takes into account the naming difference between V1 and V2 formats.

**Note:** If not all checkpoints exist, the length of the returned mtimes list will be smaller than the length of **checkpoint\_prefixes** list, so mapping checkpoints to corresponding mtimes will not be possible.

#### Args:

* **checkpoint\_prefixes**: a list of checkpoint paths, typically the results of Saver.save() or those of tf.train.latest\_checkpoint(), regardless of sharded/non-sharded or V1/V2.

#### Returns:

A list of mtimes (in microseconds) of the found checkpoints.

# tf.compat.v1.train.get\_global\_step

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_global_step#top_of_page)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_global_step#used_in_the_guide)

Get the global step tensor.

tf.compat.v1.train.get\_global\_step(graph=None)

Defined in [python/training/training\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/training_util.py).

### Used in the guide:

* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

The global step tensor must be an integer variable. We first try to find it in the collection GLOBAL\_STEP, or by name global\_step:0.

#### Args:

* **graph**: The graph to find the global step in. If missing, use default graph.

#### Returns:

The global step variable, or None if none was found.

#### Raises:

* **TypeError**: If the global step tensor has a non-integer type, or if it is not a Variable.

# tf.compat.v1.train.get\_or\_create\_global\_step

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_or_create_global_step#top_of_page)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_or_create_global_step#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/get_or_create_global_step#used_in_the_tutorials)

Returns and create (if necessary) the global step tensor.

tf.compat.v1.train.get\_or\_create\_global\_step(graph=None)

Defined in [python/training/training\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/training_util.py).

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)

### Used in the tutorials:

* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)

#### Args:

* **graph**: The graph in which to create the global step tensor. If missing, use default graph.

#### Returns:

The global step tensor.

# tf.compat.v1.train.global\_step

Small helper to get the global step.

tf.compat.v1.train.global\_step(  
    sess,  
    global\_step\_tensor  
)

Defined in [python/training/training\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/training_util.py).

# Create a variable to hold the global\_step.  
global\_step\_tensor = tf.Variable(10, trainable=False, name='global\_step')  
# Create a session.  
sess = tf.compat.v1.Session()  
# Initialize the variable  
sess.run(global\_step\_tensor.initializer)  
# Get the variable value.  
print('global\_step: %s' % tf.compat.v1.train.global\_step(sess,  
global\_step\_tensor))  
  
global\_step: 10

#### Args:

* **sess**: A TensorFlow Session object.
* **global\_step\_tensor**: Tensor or the name of the operation that contains the global step.

#### Returns:

The global step value.

# tf.compat.v1.train.GradientDescentOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer#top_of_page)
* [Class GradientDescentOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer#class_gradientdescentoptimizer)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/GradientDescentOptimizer#methods)

## Class GradientDescentOptimizer

Optimizer that implements the gradient descent algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/training/gradient\_descent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/gradient_descent.py).

### Used in the tutorials:

* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    use\_locking=False,  
    name='GradientDescent'  
)

Construct a new gradient descent optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate to use.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "GradientDescent".

#### Eager Compatibility

When eager execution is enabled, learning\_rate can be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.import\_meta\_graph

Recreates a Graph saved in a MetaGraphDef proto.

tf.compat.v1.train.import\_meta\_graph(  
    meta\_graph\_or\_file,  
    clear\_devices=False,  
    import\_scope=None,  
    \*\*kwargs  
)

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

This function takes a MetaGraphDef protocol buffer as input. If the argument is a file containing a MetaGraphDef protocol buffer , it constructs a protocol buffer from the file content. The function then adds all the nodes from the graph\_def field to the current graph, recreates all the collections, and returns a saver constructed from the saver\_def field.

In combination with export\_meta\_graph(), this function can be used to

* Serialize a graph along with other Python objects such as QueueRunner, Variable into a MetaGraphDef.
* Restart training from a saved graph and checkpoints.
* Run inference from a saved graph and checkpoints.

...  
# Create a saver.  
saver = tf.compat.v1.train.Saver(...variables...)  
# Remember the training\_op we want to run by adding it to a collection.  
tf.compat.v1.add\_to\_collection('train\_op', train\_op)  
sess = tf.compat.v1.Session()  
for step in xrange(1000000):  
    sess.run(train\_op)  
    if step % 1000 == 0:  
        # Saves checkpoint, which by default also exports a meta\_graph  
        # named 'my-model-global\_step.meta'.  
        saver.save(sess, 'my-model', global\_step=step)

Later we can continue training from this saved meta\_graph without building the model from scratch.

with tf.compat.v1.Session() as sess:  
  new\_saver =  
  tf.compat.v1.train.import\_meta\_graph('my-save-dir/my-model-10000.meta')  
  new\_saver.restore(sess, 'my-save-dir/my-model-10000')  
  # tf.compat.v1.get\_collection() returns a list. In this example we only want  
  # the first one.  
  train\_op = tf.compat.v1.get\_collection('train\_op')[0]  
  for step in xrange(1000000):  
    sess.run(train\_op)

NOTE: Restarting training from saved meta\_graph only works if the device assignments have not changed.

#### Example 2:

Variables, placeholders, and independent operations can also be stored, as shown in the following example.

# Saving contents and operations.  
v1 = tf.compat.v1.placeholder(tf.float32, name="v1")  
v2 = tf.compat.v1.placeholder(tf.float32, name="v2")  
v3 = tf.mul(v1, v2)  
vx = tf.Variable(10.0, name="vx")  
v4 = tf.add(v3, vx, name="v4")  
saver = tf.compat.v1.train.Saver([vx])  
sess = tf.compat.v1.Session()  
sess.run(tf.compat.v1.initialize\_all\_variables())  
sess.run(vx.assign(tf.add(vx, vx)))  
result = sess.run(v4, feed\_dict={v1:12.0, v2:3.3})  
print(result)  
saver.save(sess, "./model\_ex1")

Later this model can be restored and contents loaded.

# Restoring variables and running operations.  
saver = tf.compat.v1.train.import\_meta\_graph("./model\_ex1.meta")  
sess = tf.compat.v1.Session()  
saver.restore(sess, "./model\_ex1")  
result = sess.run("v4:0", feed\_dict={"v1:0": 12.0, "v2:0": 3.3})  
print(result)

#### Args:

* **meta\_graph\_or\_file**: MetaGraphDef protocol buffer or filename (including the path) containing a MetaGraphDef.
* **clear\_devices**: Whether or not to clear the device field for an Operation or Tensor during import.
* **import\_scope**: Optional string. Name scope to add. Only used when initializing from protocol buffer.
* **\*\*kwargs**: Optional keyed arguments.

#### Returns:

A saver constructed from saver\_def in MetaGraphDef or None.

A None value is returned if no variables exist in the MetaGraphDef (i.e., there are no variables to restore).

#### Raises:

* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Exporting/importing meta graphs is not supported. No graph exists when eager execution is enabled.

# tf.compat.v1.train.init\_from\_checkpoint

Replaces [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) initializers so they load from a checkpoint file.

tf.compat.v1.train.init\_from\_checkpoint(  
    ckpt\_dir\_or\_file,  
    assignment\_map  
)

Defined in [python/training/checkpoint\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_utils.py).

Values are not loaded immediately, but when the initializer is run (typically by running a [tf.compat.v1.global\_variables\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/global_variables_initializer) op).

**Note:** This overrides default initialization ops of specified variables and redefines dtype.

Assignment map supports following syntax:

* 'checkpoint\_scope\_name/': 'scope\_name/' - will load all variables in current scope\_namefrom checkpoint\_scope\_name with matching tensor names.
* 'checkpoint\_scope\_name/some\_other\_variable': 'scope\_name/variable\_name' - will initialize scope\_name/variable\_name variable from checkpoint\_scope\_name/some\_other\_variable.
* 'scope\_variable\_name': variable - will initialize given [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) object with tensor 'scope\_variable\_name' from the checkpoint.
* 'scope\_variable\_name': list(variable) - will initialize list of partitioned variables with tensor 'scope\_variable\_name' from the checkpoint.
* '/': 'scope\_name/' - will load all variables in current scope\_name from checkpoint's root (e.g. no scope).

Supports loading into partitioned variables, which are represented as '<variable>/part\_<part #>'.

#### Example:

# Say, '/tmp/model.ckpt' has the following tensors:  
#  -- name='old\_scope\_1/var1', shape=[20, 2]  
#  -- name='old\_scope\_1/var2', shape=[50, 4]  
#  -- name='old\_scope\_2/var3', shape=[100, 100]  
  
# Create new model's variables  
with tf.compat.v1.variable\_scope('new\_scope\_1'):  
  var1 = tf.compat.v1.get\_variable('var1', shape=[20, 2],  
                         initializer=tf.compat.v1.zeros\_initializer())  
with tf.compat.v1.variable\_scope('new\_scope\_2'):  
  var2 = tf.compat.v1.get\_variable('var2', shape=[50, 4],  
                         initializer=tf.compat.v1.zeros\_initializer())  
  # Partition into 5 variables along the first axis.  
  var3 = tf.compat.v1.get\_variable(name='var3', shape=[100, 100],  
                         initializer=tf.compat.v1.zeros\_initializer(),  
                         partitioner=lambda shape, dtype: [5, 1])  
  
# Initialize all variables in `new\_scope\_1` from `old\_scope\_1`.  
init\_from\_checkpoint('/tmp/model.ckpt', {'old\_scope\_1/': 'new\_scope\_1'})  
  
# Use names to specify which variables to initialize from checkpoint.  
init\_from\_checkpoint('/tmp/model.ckpt',  
                     {'old\_scope\_1/var1': 'new\_scope\_1/var1',  
                      'old\_scope\_1/var2': 'new\_scope\_2/var2'})  
  
# Or use tf.Variable objects to identify what to initialize.  
init\_from\_checkpoint('/tmp/model.ckpt',  
                     {'old\_scope\_1/var1': var1,  
                      'old\_scope\_1/var2': var2})  
  
# Initialize partitioned variables using variable's name  
init\_from\_checkpoint('/tmp/model.ckpt',  
                     {'old\_scope\_2/var3': 'new\_scope\_2/var3'})  
  
# Or specify the list of tf.Variable objects.  
init\_from\_checkpoint('/tmp/model.ckpt',  
                     {'old\_scope\_2/var3': var3.\_get\_variable\_list()})

#### Args:

* **ckpt\_dir\_or\_file**: Directory with checkpoints file or path to checkpoint.
* **assignment\_map**: Dict, where keys are names of the variables in the checkpoint and values are current variables or names of current variables (in default graph).

#### Raises:

* **ValueError**: If missing variables in current graph, or if missing checkpoints or tensors in checkpoints.

# tf.compat.v1.train.input\_producer

Output the rows of input\_tensor to a queue for an input pipeline. (deprecated)

tf.compat.v1.train.input\_producer(  
    input\_tensor,  
    element\_shape=None,  
    num\_epochs=None,  
    shuffle=True,  
    seed=None,  
    capacity=32,  
    shared\_name=None,  
    summary\_name=None,  
    name=None,  
    cancel\_op=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.from\_tensor\_slices(input\_tensor).shuffle(tf.shape(input\_tensor, out\_type=tf.int64)[0]).repeat(num\_epochs)**. If **shuffle=False**, omit the **.shuffle(...)**.**Note:** if **num\_epochs** is not **None**, this function creates local counter **epochs**. Use **local\_variables\_initializer()** to initialize local variables.

#### Args:

* **input\_tensor**: A tensor with the rows to produce. Must be at least one-dimensional. Must either have a fully-defined shape, or element\_shape must be defined.
* **element\_shape**: (Optional.) A TensorShape representing the shape of a row of input\_tensor, if it cannot be inferred.
* **num\_epochs**: (Optional.) An integer. If specified input\_producer produces each row of input\_tensor num\_epochs times before generating an OutOfRange error. If not specified, input\_producer can cycle through the rows of input\_tensor an unlimited number of times.
* **shuffle**: (Optional.) A boolean. If true, the rows are randomly shuffled within each epoch.
* **seed**: (Optional.) An integer. The seed to use if shuffle is true.
* **capacity**: (Optional.) The capacity of the queue to be used for buffering the input.
* **shared\_name**: (Optional.) If set, this queue will be shared under the given name across multiple sessions.
* **summary\_name**: (Optional.) If set, a scalar summary for the current queue size will be generated, using this name as part of the tag.
* **name**: (Optional.) A name for queue.
* **cancel\_op**: (Optional.) Cancel op for the queue

#### Returns:

A queue with the output rows. A QueueRunner for the queue is added to the current QUEUE\_RUNNERcollection of the current graph.

#### Raises:

* **ValueError**: If the shape of the input cannot be inferred from the arguments.
* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.inverse\_time\_decay

Applies inverse time decay to the initial learning rate.

tf.compat.v1.train.inverse\_time\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    decay\_rate,  
    staircase=False,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies an inverse decay function to a provided initial learning rate. It requires an global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

decayed\_learning\_rate = learning\_rate / (1 + decay\_rate \* global\_step /  
decay\_step)

or, if staircase is True, as:

decayed\_learning\_rate = learning\_rate / (1 + decay\_rate \* floor(global\_step /  
decay\_step))

Example: decay 1/t with a rate of 0.5:

...  
global\_step = tf.Variable(0, trainable=False)  
learning\_rate = 0.1  
decay\_steps = 1.0  
decay\_rate = 0.5  
learning\_rate = tf.compat.v1.train.inverse\_time\_decay(learning\_rate,  
global\_step,  
decay\_steps, decay\_rate)  
  
# Passing global\_step to minimize() will increment it at each step.  
learning\_step = (  
    tf.compat.v1.train.GradientDescentOptimizer(learning\_rate)  
    .minimize(...my loss..., global\_step=global\_step)  
)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A Python number. Global step to use for the decay computation. Must not be negative.
* **decay\_steps**: How often to apply decay.
* **decay\_rate**: A Python number. The decay rate.
* **staircase**: Whether to apply decay in a discrete staircase, as opposed to continuous, fashion.
* **name**: String. Optional name of the operation. Defaults to 'InverseTimeDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.limit\_epochs

Returns tensor num\_epochs times and then raises an OutOfRange error. (deprecated)

tf.compat.v1.train.limit\_epochs(  
    tensor,  
    num\_epochs=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.from\_tensors(tensor).repeat(num\_epochs)**.**Note:** creates local counter **epochs**. Use **local\_variables\_initializer()** to initialize local variables.

#### Args:

* **tensor**: Any Tensor.
* **num\_epochs**: A positive integer (optional). If specified, limits the number of steps the output tensor may be evaluated.
* **name**: A name for the operations (optional).

#### Returns:

tensor or OutOfRange.

#### Raises:

* **ValueError**: if num\_epochs is invalid.

# tf.compat.v1.train.linear\_cosine\_decay

Applies linear cosine decay to the learning rate.

tf.compat.v1.train.linear\_cosine\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    num\_periods=0.5,  
    alpha=0.0,  
    beta=0.001,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

See [Bello et al., ICML2017] Neural Optimizer Search with RL. https://arxiv.org/abs/1709.07417

For the idea of warm starts here controlled by num\_periods, see [Loshchilov & Hutter, ICLR2016] SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

Note that linear cosine decay is more aggressive than cosine decay and larger initial learning rates can typically be used.

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies a linear cosine decay function to a provided initial learning rate. It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

global\_step = min(global\_step, decay\_steps)  
linear\_decay = (decay\_steps - global\_step) / decay\_steps)  
cosine\_decay = 0.5 \* (  
    1 + cos(pi \* 2 \* num\_periods \* global\_step / decay\_steps))  
decayed = (alpha + linear\_decay) \* cosine\_decay + beta  
decayed\_learning\_rate = learning\_rate \* decayed

#### Example usage:

decay\_steps = 1000  
lr\_decayed = linear\_cosine\_decay(learning\_rate, global\_step, decay\_steps)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **num\_periods**: Number of periods in the cosine part of the decay. See computation above.
* **alpha**: See computation above.
* **beta**: See computation above.
* **name**: String. Optional name of the operation. Defaults to 'LinearCosineDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.LooperThread

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread#top_of_page)
* [Class LooperThread](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread#class_looperthread)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread#properties)
  + [daemon](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/LooperThread#daemon)

## Class LooperThread

A thread that runs code repeatedly, optionally on a timer.

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

This thread class is intended to be used with a Coordinator. It repeatedly runs code specified either as target and args or by the run\_loop() method.

Before each run the thread checks if the coordinator has requested stop. In that case the looper thread terminates immediately.

If the code being run raises an exception, that exception is reported to the coordinator and the thread terminates. The coordinator will then request all the other threads it coordinates to stop.

You typically pass looper threads to the supervisor Join() method.

## \_\_init\_\_

\_\_init\_\_(  
    coord,  
    timer\_interval\_secs,  
    target=None,  
    args=None,  
    kwargs=None  
)

Create a LooperThread.

#### Args:

* **coord**: A Coordinator.
* **timer\_interval\_secs**: Time boundaries at which to call Run(), or None if it should be called back to back.
* **target**: Optional callable object that will be executed in the thread.
* **args**: Optional arguments to pass to target when calling it.
* **kwargs**: Optional keyword arguments to pass to target when calling it.

#### Raises:

* **ValueError**: If one of the arguments is invalid.

## Properties

### daemon

A boolean value indicating whether this thread is a daemon thread.

This must be set before start() is called, otherwise RuntimeError is raised. Its initial value is inherited from the creating thread; the main thread is not a daemon thread and therefore all threads created in the main thread default to daemon = False.

The entire Python program exits when no alive non-daemon threads are left.

### ident

Thread identifier of this thread or None if it has not been started.

This is a nonzero integer. See the thread.get\_ident() function. Thread identifiers may be recycled when a thread exits and another thread is created. The identifier is available even after the thread has exited.

### name

A string used for identification purposes only.

It has no semantics. Multiple threads may be given the same name. The initial name is set by the constructor.

## Methods

### getName

getName()

### isAlive

isAlive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. The module function enumerate() returns a list of all alive threads.

### isDaemon

isDaemon()

### is\_alive

is\_alive()

Return whether the thread is alive.

This method returns True just before the run() method starts until just after the run() method terminates. The module function enumerate() returns a list of all alive threads.

### join

join(timeout=None)

Wait until the thread terminates.

This blocks the calling thread until the thread whose join() method is called terminates -- either normally or through an unhandled exception or until the optional timeout occurs.

When the timeout argument is present and not None, it should be a floating point number specifying a timeout for the operation in seconds (or fractions thereof). As join() always returns None, you must call isAlive() after join() to decide whether a timeout happened -- if the thread is still alive, the join() call timed out.

When the timeout argument is not present or None, the operation will block until the thread terminates.

A thread can be join()ed many times.

join() raises a RuntimeError if an attempt is made to join the current thread as that would cause a deadlock. It is also an error to join() a thread before it has been started and attempts to do so raises the same exception.

### loop

@staticmethod  
loop(  
    coord,  
    timer\_interval\_secs,  
    target,  
    args=None,  
    kwargs=None  
)

Start a LooperThread that calls a function periodically.

If timer\_interval\_secs is None the thread calls target(args) repeatedly. Otherwise target(args) is called every timer\_interval\_secs seconds. The thread terminates when a stop of the coordinator is requested.

#### Args:

* **coord**: A Coordinator.
* **timer\_interval\_secs**: Number. Time boundaries at which to call target.
* **target**: A callable object.
* **args**: Optional arguments to pass to target when calling it.
* **kwargs**: Optional keyword arguments to pass to target when calling it.

#### Returns:

The started thread.

### run

run()

### run\_loop

run\_loop()

Called at 'timer\_interval\_secs' boundaries.

### setDaemon

setDaemon(daemonic)

### setName

setName(name)

### start

start()

Start the thread's activity.

It must be called at most once per thread object. It arranges for the object's run() method to be invoked in a separate thread of control.

This method will raise a RuntimeError if called more than once on the same thread object.

### start\_loop

start\_loop()

Called when the thread starts.

### stop\_loop

stop\_loop()

Called when the thread stops.

# tf.compat.v1.train.maybe\_batch

Conditionally creates batches of tensors based on keep\_input. (deprecated)

tf.compat.v1.train.maybe\_batch(  
    tensors,  
    keep\_input,  
    batch\_size,  
    num\_threads=1,  
    capacity=32,  
    enqueue\_many=False,  
    shapes=None,  
    dynamic\_pad=False,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.filter(...).batch(batch\_size)** (or **padded\_batch(...)** if **dynamic\_pad=True**).

See docstring in batch for more details.

#### Args:

* **tensors**: The list or dictionary of tensors to enqueue.
* **keep\_input**: A bool Tensor. This tensor controls whether the input is added to the queue or not. If it is a scalar and evaluates True, then tensors are all added to the queue. If it is a vector and enqueue\_many is True, then each example is added to the queue only if the corresponding value in keep\_input is True. This tensor essentially acts as a filtering mechanism.
* **batch\_size**: The new batch size pulled from the queue.
* **num\_threads**: The number of threads enqueuing tensors. The batching will be nondeterministic if num\_threads > 1.
* **capacity**: An integer. The maximum number of elements in the queue.
* **enqueue\_many**: Whether each tensor in tensors is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensors.
* **dynamic\_pad**: Boolean. Allow variable dimensions in input shapes. The given dimensions are padded upon dequeue so that tensors within a batch have the same shapes.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same types as tensors.

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors.

# tf.compat.v1.train.maybe\_batch\_join

Runs a list of tensors to conditionally fill a queue to create batches. (deprecated)

tf.compat.v1.train.maybe\_batch\_join(  
    tensors\_list,  
    keep\_input,  
    batch\_size,  
    capacity=32,  
    enqueue\_many=False,  
    shapes=None,  
    dynamic\_pad=False,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.interleave(...).filter(...).batch(batch\_size)** (or **padded\_batch(...)** if **dynamic\_pad=True**).

See docstring in batch\_join for more details.

#### Args:

* **tensors\_list**: A list of tuples or dictionaries of tensors to enqueue.
* **keep\_input**: A bool Tensor. This tensor controls whether the input is added to the queue or not. If it is a scalar and evaluates True, then tensors are all added to the queue. If it is a vector and enqueue\_many is True, then each example is added to the queue only if the corresponding value in keep\_input is True. This tensor essentially acts as a filtering mechanism.
* **batch\_size**: An integer. The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensor\_list\_list[i].
* **dynamic\_pad**: Boolean. Allow variable dimensions in input shapes. The given dimensions are padded upon dequeue so that tensors within a batch have the same shapes.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional) If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same number and types as tensors\_list[i].

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensor\_list\_list.

# tf.compat.v1.train.maybe\_shuffle\_batch

Creates batches by randomly shuffling conditionally-enqueued tensors. (deprecated)

tf.compat.v1.train.maybe\_shuffle\_batch(  
    tensors,  
    batch\_size,  
    capacity,  
    min\_after\_dequeue,  
    keep\_input,  
    num\_threads=1,  
    seed=None,  
    enqueue\_many=False,  
    shapes=None,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.filter(...).shuffle(min\_after\_dequeue).batch(batch\_size)**.

See docstring in shuffle\_batch for more details.

#### Args:

* **tensors**: The list or dictionary of tensors to enqueue.
* **batch\_size**: The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **min\_after\_dequeue**: Minimum number elements in the queue after a dequeue, used to ensure a level of mixing of elements.
* **keep\_input**: A bool Tensor. This tensor controls whether the input is added to the queue or not. If it is a scalar and evaluates True, then tensors are all added to the queue. If it is a vector and enqueue\_many is True, then each example is added to the queue only if the corresponding value in keep\_input is True. This tensor essentially acts as a filtering mechanism.
* **num\_threads**: The number of threads enqueuing tensor\_list.
* **seed**: Seed for the random shuffling within the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensor\_list.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional) If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the types as tensors.

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.maybe\_shuffle\_batch\_join

Create batches by randomly shuffling conditionally-enqueued tensors. (deprecated)

tf.compat.v1.train.maybe\_shuffle\_batch\_join(  
    tensors\_list,  
    batch\_size,  
    capacity,  
    min\_after\_dequeue,  
    keep\_input,  
    seed=None,  
    enqueue\_many=False,  
    shapes=None,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.interleave(...).filter(...).shuffle(min\_after\_dequeue).batch(batch\_size)**.

See docstring in shuffle\_batch\_join for more details.

#### Args:

* **tensors\_list**: A list of tuples or dictionaries of tensors to enqueue.
* **batch\_size**: An integer. The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **min\_after\_dequeue**: Minimum number elements in the queue after a dequeue, used to ensure a level of mixing of elements.
* **keep\_input**: A bool Tensor. This tensor controls whether the input is added to the queue or not. If it is a scalar and evaluates True, then tensors are all added to the queue. If it is a vector and enqueue\_many is True, then each example is added to the queue only if the corresponding value in keep\_input is True. This tensor essentially acts as a filtering mechanism.
* **seed**: Seed for the random shuffling within the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensors\_list[i].
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same number and types as tensors\_list[i].

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors\_list.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.MomentumOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer#top_of_page)
* [Class MomentumOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer#class_momentumoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MomentumOptimizer#apply_gradients)

## Class MomentumOptimizer

Optimizer that implements the Momentum algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

Computes (if use\_nesterov = False):

accumulation = momentum \* accumulation + gradient  
variable -= learning\_rate \* accumulation

Note that in the dense version of this algorithm, accumulation is updated and applied regardless of a gradient's value, whereas the sparse version (when the gradient is an IndexedSlices, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding) only updates variable slices and corresponding accumulation terms when that part of the variable was used in the forward pass.

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    momentum,  
    use\_locking=False,  
    name='Momentum',  
    use\_nesterov=False  
)

Construct a new Momentum optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **momentum**: A Tensor or a floating point value. The momentum.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Momentum".
* **use\_nesterov**: If True use Nesterov Momentum. See [Sutskever et al., 2013](http://jmlr.org/proceedings/papers/v28/sutskever13.pdf). This implementation always computes gradients at the value of the variable(s) passed to the optimizer. Using Nesterov Momentum makes the variable(s) track the values called theta\_t + mu\*v\_t in the paper. This implementation is an approximation of the original formula, valid for high values of momentum. It will compute the "adjusted gradient" in NAG by assuming that the new gradient will be estimated by the current average gradient plus the product of momentum and the change in the average gradient.

#### Eager Compatibility

When eager execution is enabled, learning\_rate and momentum can each be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.MonitoredSession

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession#top_of_page)
* [Class MonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession#class_monitoredsession)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession#__init__)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession#child_classes)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession#properties)

## Class MonitoredSession

Session-like object that handles initialization, recovery and hooks.

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

#### Example usage:

saver\_hook = CheckpointSaverHook(...)  
summary\_hook = SummarySaverHook(...)  
with MonitoredSession(session\_creator=ChiefSessionCreator(...),  
                      hooks=[saver\_hook, summary\_hook]) as sess:  
  while not sess.should\_stop():  
    sess.run(train\_op)

Initialization: At creation time the monitored session does following things in given order:

* calls hook.begin() for each given hook
* finalizes the graph via scaffold.finalize()
* create session
* initializes the model via initialization ops provided by Scaffold
* restores variables if a checkpoint exists
* launches queue runners
* calls hook.after\_create\_session()

Run: When run() is called, the monitored session does following things:

* calls hook.before\_run()
* calls TensorFlow session.run() with merged fetches and feed\_dict
* calls hook.after\_run()
* returns result of session.run() asked by user
* if AbortedError or UnavailableError occurs, it recovers or reinitializes the session before executing the run() call again

Exit: At the close(), the monitored session does following things in order:

* calls hook.end()
* closes the queue runners and the session
* suppresses OutOfRange error which indicates that all inputs have been processed if the monitored\_session is used as a context

How to set [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) arguments:

* In most cases you can set session arguments as follows:

MonitoredSession(  
  session\_creator=ChiefSessionCreator(master=..., config=...))

* In distributed setting for a non-chief worker, you can use following:

MonitoredSession(  
  session\_creator=WorkerSessionCreator(master=..., config=...))

See MonitoredTrainingSession for an example usage based on chief or worker.

**Note:** This is not a [**tf.compat.v1.Session**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session). For example, it cannot do following:

* it cannot be set as default session.
* it cannot be sent to saver.save.
* it cannot be sent to tf.train.start\_queue\_runners.

#### Args:

* **session\_creator**: A factory object to create session. Typically a ChiefSessionCreatorwhich is the default one.
* **hooks**: An iterable of `SessionRunHook' objects.

#### Returns:

A MonitoredSession object.

## \_\_init\_\_

\_\_init\_\_(  
    session\_creator=None,  
    hooks=None,  
    stop\_grace\_period\_secs=120  
)

## Child Classes

[class StepContext](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext)

## Properties

### graph

The graph that was launched in this session.

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

### \_\_exit\_\_

\_\_exit\_\_(  
    exception\_type,  
    exception\_value,  
    traceback  
)

### close

close()

### run

run(  
    fetches,  
    feed\_dict=None,  
    options=None,  
    run\_metadata=None  
)

Run ops in the monitored session.

This method is completely compatible with the tf.Session.run() method.

#### Args:

* **fetches**: Same as tf.Session.run().
* **feed\_dict**: Same as tf.Session.run().
* **options**: Same as tf.Session.run().
* **run\_metadata**: Same as tf.Session.run().

#### Returns:

Same as tf.Session.run().

### run\_step\_fn

run\_step\_fn(step\_fn)

Run ops using a step function.

#### Args:

* **step\_fn**: A function or a method with a single argument of type StepContext. The function may use methods of the argument to perform computations with access to a raw session. The returned value of the step\_fn will be returned from run\_step\_fn, unless a stop is requested. In that case, the next should\_stop call will return True. Example usage: ```python with tf.Graph().as\_default(): c = tf.compat.v1.placeholder(dtypes.float32) v = tf.add(c, 4.0) w = tf.add(c, 0.5) def step\_fn(step\_context): a = step\_context.session.run(fetches=v, feed\_dict={c: 0.5}) if a <= 4.5: step\_context.request\_stop() return step\_context.run\_with\_hooks(fetches=w, feed\_dict={c: 0.1}) with tf.MonitoredSession() as session: while not session.should\_stop(): a = session.run\_step\_fn(step\_fn)

   ```  Hooks interact with the `run\_with\_hooks()` call inside the  
   `step\_fn` as they do with a `MonitoredSession.run` call.

#### Returns:

Returns the returned value of step\_fn.

#### Raises:

* **StopIteration**: if step\_fn has called request\_stop(). It may be caught by with tf.MonitoredSession() to close the session.
* **ValueError**: if step\_fn doesn't have a single argument called step\_context. It may also optionally have self for cases when it belongs to an object.

### should\_stop

should\_stop()

# tf.compat.v1.train.MonitoredSession.StepContext

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext#top_of_page)
* [Class StepContext](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext#class_stepcontext)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext#properties)

## Class StepContext

Control flow instrument for the step\_fn from run\_step\_fn().

### Aliases:

* Class tf.compat.v1.train.MonitoredSession.StepContext
* Class tf.compat.v1.train.SingularMonitoredSession.StepContext

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

Users of step\_fn may perform run() calls without running hooks by accessing the session. A run() call with hooks may be performed using run\_with\_hooks(). Computation flow can be interrupted using request\_stop().

## \_\_init\_\_

\_\_init\_\_(  
    session,  
    run\_with\_hooks\_fn  
)

Initializes the step\_context argument for a step\_fn invocation.

#### Args:

* **session**: An instance of [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session).
* **run\_with\_hooks\_fn**: A function for running fetches and hooks.

## Properties

### session

## Methods

### request\_stop

request\_stop()

Exit the training loop by causing should\_stop() to return True.

Causes step\_fn to exit by raising an exception.

#### Raises:

StopIteration

### run\_with\_hooks

run\_with\_hooks(  
    \*args,  
    \*\*kwargs  
)

Same as MonitoredSession.run. Accepts the same arguments.

# tf.compat.v1.train.MonitoredTrainingSession

Creates a MonitoredSession for training.

tf.compat.v1.train.MonitoredTrainingSession(  
    master='',  
    is\_chief=True,  
    checkpoint\_dir=None,  
    scaffold=None,  
    hooks=None,  
    chief\_only\_hooks=None,  
    save\_checkpoint\_secs=USE\_DEFAULT,  
    save\_summaries\_steps=USE\_DEFAULT,  
    save\_summaries\_secs=USE\_DEFAULT,  
    config=None,  
    stop\_grace\_period\_secs=120,  
    log\_step\_count\_steps=100,  
    max\_wait\_secs=7200,  
    save\_checkpoint\_steps=USE\_DEFAULT,  
    summary\_dir=None  
)

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

For a chief, this utility sets proper session initializer/restorer. It also creates hooks related to checkpoint and summary saving. For workers, this utility sets proper session creator which waits for the chief to initialize/restore. Please check [tf.compat.v1.train.MonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession) for more information.

#### Args:

* **master**: String the TensorFlow master to use.
* **is\_chief**: If True, it will take care of initialization and recovery the underlying TensorFlow session. If False, it will wait on a chief to initialize or recover the TensorFlow session.
* **checkpoint\_dir**: A string. Optional path to a directory where to restore variables.
* **scaffold**: A Scaffold used for gathering or building supportive ops. If not specified, a default one is created. It's used to finalize the graph.
* **hooks**: Optional list of SessionRunHook objects.
* **chief\_only\_hooks**: list of SessionRunHook objects. Activate these hooks ifis\_chief==True, ignore otherwise.
* **save\_checkpoint\_secs**: The frequency, in seconds, that a checkpoint is saved using a default checkpoint saver. If both save\_checkpoint\_steps and save\_checkpoint\_secs are set to None, then the default checkpoint saver isn't used. If both are provided, then only save\_checkpoint\_secs is used. Default 600.
* **save\_summaries\_steps**: The frequency, in number of global steps, that the summaries are written to disk using a default summary saver. If both save\_summaries\_steps and save\_summaries\_secs are set to None, then the default summary saver isn't used. Default 100.
* **save\_summaries\_secs**: The frequency, in secs, that the summaries are written to disk using a default summary saver. If both save\_summaries\_steps and save\_summaries\_secs are set to None, then the default summary saver isn't used. Default not enabled.
* **config**: an instance of [tf.compat.v1.ConfigProto](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ConfigProto) proto used to configure the session. It's the config argument of constructor of [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session).
* **stop\_grace\_period\_secs**: Number of seconds given to threads to stop after close() has been called.
* **log\_step\_count\_steps**: The frequency, in number of global steps, that the global step/sec is logged.
* **max\_wait\_secs**: Maximum time workers should wait for the session to become available. This should be kept relatively short to help detect incorrect code, but sometimes may need to be increased if the chief takes a while to start up.
* **save\_checkpoint\_steps**: The frequency, in number of global steps, that a checkpoint is saved using a default checkpoint saver. If both save\_checkpoint\_steps and save\_checkpoint\_secs are set to None, then the default checkpoint saver isn't used. If both are provided, then only save\_checkpoint\_secs is used. Default not enabled.
* **summary\_dir**: A string. Optional path to a directory where to save summaries. If None, checkpoint\_dir is used instead.

#### Returns:

A MonitoredSession object.

# tf.compat.v1.train.natural\_exp\_decay

Applies natural exponential decay to the initial learning rate.

tf.compat.v1.train.natural\_exp\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    decay\_rate,  
    staircase=False,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies an exponential decay function to a provided initial learning rate. It requires an global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

decayed\_learning\_rate = learning\_rate \* exp(-decay\_rate \* global\_step /  
decay\_step)

or, if staircase is True, as:

decayed\_learning\_rate = learning\_rate \* exp(-decay\_rate \* floor(global\_step /  
decay\_step))

Example: decay exponentially with a base of 0.96:

...  
global\_step = tf.Variable(0, trainable=False)  
learning\_rate = 0.1  
decay\_steps = 5  
k = 0.5  
learning\_rate = tf.compat.v1.train.natural\_exp\_decay(learning\_rate,  
global\_step,  
                                           decay\_steps, k)  
  
# Passing global\_step to minimize() will increment it at each step.  
learning\_step = (  
    tf.compat.v1.train.GradientDescentOptimizer(learning\_rate)  
    .minimize(...my loss..., global\_step=global\_step)  
)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A Python number. Global step to use for the decay computation. Must not be negative.
* **decay\_steps**: How often to apply decay.
* **decay\_rate**: A Python number. The decay rate.
* **staircase**: Whether to apply decay in a discrete staircase, as opposed to continuous, fashion.
* **name**: String. Optional name of the operation. Defaults to 'ExponentialTimeDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.NewCheckpointReader

tf.compat.v1.train.NewCheckpointReader(filepattern)

Defined in [python/pywrap\_tensorflow\_internal.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/pywrap_tensorflow_internal.py).

# tf.compat.v1.train.noisy\_linear\_cosine\_decay

Applies noisy linear cosine decay to the learning rate.

tf.compat.v1.train.noisy\_linear\_cosine\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    initial\_variance=1.0,  
    variance\_decay=0.55,  
    num\_periods=0.5,  
    alpha=0.0,  
    beta=0.001,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

See [Bello et al., ICML2017] Neural Optimizer Search with RL. https://arxiv.org/abs/1709.07417

For the idea of warm starts here controlled by num\_periods, see [Loshchilov & Hutter, ICLR2016] SGDR: Stochastic Gradient Descent with Warm Restarts. https://arxiv.org/abs/1608.03983

Note that linear cosine decay is more aggressive than cosine decay and larger initial learning rates can typically be used.

When training a model, it is often recommended to lower the learning rate as the training progresses. This function applies a noisy linear cosine decay function to a provided initial learning rate. It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

global\_step = min(global\_step, decay\_steps)  
linear\_decay = (decay\_steps - global\_step) / decay\_steps)  
cosine\_decay = 0.5 \* (  
    1 + cos(pi \* 2 \* num\_periods \* global\_step / decay\_steps))  
decayed = (alpha + linear\_decay + eps\_t) \* cosine\_decay + beta  
decayed\_learning\_rate = learning\_rate \* decayed

where eps\_t is 0-centered gaussian noise with variance initial\_variance / (1 + global\_step) \*\* variance\_decay

#### Example usage:

decay\_steps = 1000  
lr\_decayed = noisy\_linear\_cosine\_decay(  
  learning\_rate, global\_step, decay\_steps)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Number of steps to decay over.
* **initial\_variance**: initial variance for the noise. See computation above.
* **variance\_decay**: decay for the noise's variance. See computation above.
* **num\_periods**: Number of periods in the cosine part of the decay. See computation above.
* **alpha**: See computation above.
* **beta**: See computation above.
* **name**: String. Optional name of the operation. Defaults to 'NoisyLinearCosineDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.Optimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer#top_of_page)
* [Class Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer#class_optimizer)
  + [Usage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer#usage)
  + [Processing gradients before applying them.](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer#processing_gradients_before_applying_them)
  + [Gating Gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer#gating_gradients)

## Class Optimizer

Base class for optimizers.

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

This class defines the API to add Ops to train a model. You never use this class directly, but instead instantiate one of its subclasses such as GradientDescentOptimizer, AdagradOptimizer, or MomentumOptimizer.

### Usage

# Create an optimizer with the desired parameters.  
opt = GradientDescentOptimizer(learning\_rate=0.1)  
# Add Ops to the graph to minimize a cost by updating a list of variables.  
# "cost" is a Tensor, and the list of variables contains tf.Variable  
# objects.  
opt\_op = opt.minimize(cost, var\_list=<list of variables>)

In the training program you will just have to run the returned Op.

# Execute opt\_op to do one step of training:  
opt\_op.run()

### Processing gradients before applying them.

Calling minimize() takes care of both computing the gradients and applying them to the variables. If you want to process the gradients before applying them you can instead use the optimizer in three steps:

1. Compute the gradients with compute\_gradients().
2. Process the gradients as you wish.
3. Apply the processed gradients with apply\_gradients().

#### Example:

# Create an optimizer.  
opt = GradientDescentOptimizer(learning\_rate=0.1)  
  
# Compute the gradients for a list of variables.  
grads\_and\_vars = opt.compute\_gradients(loss, <list of variables>)  
  
# grads\_and\_vars is a list of tuples (gradient, variable).  Do whatever you  
# need to the 'gradient' part, for example cap them, etc.  
capped\_grads\_and\_vars = [(MyCapper(gv[0]), gv[1]) for gv in grads\_and\_vars]  
  
# Ask the optimizer to apply the capped gradients.  
opt.apply\_gradients(capped\_grads\_and\_vars)

### Gating Gradients

Both minimize() and compute\_gradients() accept a gate\_gradients argument that controls the degree of parallelism during the application of the gradients.

The possible values are: GATE\_NONE, GATE\_OP, and GATE\_GRAPH.

**GATE\_NONE**: Compute and apply gradients in parallel. This provides the maximum parallelism in execution, at the cost of some non-reproducibility in the results. For example the two gradients of matmul depend on the input values: With GATE\_NONE one of the gradients could be applied to one of the inputs before the other gradient is computed resulting in non-reproducible results.

**GATE\_OP**: For each Op, make sure all gradients are computed before they are used. This prevents race conditions for Ops that generate gradients for multiple inputs where the gradients depend on the inputs.

**GATE\_GRAPH**: Make sure all gradients for all variables are computed before any one of them is used. This provides the least parallelism but can be useful if you want to process all gradients before applying any of them.

### Slots

Some optimizer subclasses, such as MomentumOptimizer and AdagradOptimizer allocate and manage additional variables associated with the variables to train. These are called *Slots*. Slots have names and you can ask the optimizer for the names of the slots that it uses. Once you have a slot name you can ask the optimizer for the variable it created to hold the slot value.

This can be useful if you want to log debug a training algorithm, report stats about the slots, etc.

## \_\_init\_\_

\_\_init\_\_(  
    use\_locking,  
    name  
)

Create a new Optimizer.

This must be called by the constructors of subclasses.

#### Args:

* **use\_locking**: Bool. If True apply use locks to prevent concurrent updates to variables.
* **name**: A non-empty string. The name to use for accumulators created for the optimizer.

#### Raises:

* **ValueError**: If name is malformed.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.piecewise\_constant

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/piecewise_constant#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/piecewise_constant#aliases)

Piecewise constant from boundaries and interval values.

### Aliases:

* tf.compat.v1.train.piecewise\_constant
* tf.compat.v1.train.piecewise\_constant\_decay

tf.compat.v1.train.piecewise\_constant(  
    x,  
    boundaries,  
    values,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

Example: use a learning rate that's 1.0 for the first 100001 steps, 0.5 for the next 10000 steps, and 0.1 for any additional steps.

global\_step = tf.Variable(0, trainable=False)  
boundaries = [100000, 110000]  
values = [1.0, 0.5, 0.1]  
learning\_rate = tf.compat.v1.train.piecewise\_constant(global\_step, boundaries,  
values)  
  
# Later, whenever we perform an optimization step, we increment global\_step.

#### Args:

* **x**: A 0-D scalar Tensor. Must be one of the following types: float32, float64, uint8, int8, int16, int32, int64.
* **boundaries**: A list of Tensors or ints or floats with strictly increasing entries, and with all elements having the same type as x.
* **values**: A list of Tensors or floats or ints that specifies the values for the intervals defined by boundaries. It should have one more element than boundaries, and all elements should have the same type.
* **name**: A string. Optional name of the operation. Defaults to 'PiecewiseConstant'.

#### Returns:

A 0-D Tensor. Its value is values[0] when x <= boundaries[0], values[1] when x > boundaries[0] and x <= boundaries[1], ..., and values[-1] when x > boundaries[-1].

#### Raises:

* **ValueError**: if types of x and boundaries do not match, or types of all values do not match or the number of elements in the lists does not match.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.polynomial\_decay

Applies a polynomial decay to the learning rate.

tf.compat.v1.train.polynomial\_decay(  
    learning\_rate,  
    global\_step,  
    decay\_steps,  
    end\_learning\_rate=0.0001,  
    power=1.0,  
    cycle=False,  
    name=None  
)

Defined in [python/training/learning\_rate\_decay.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/learning_rate_decay.py).

It is commonly observed that a monotonically decreasing learning rate, whose degree of change is carefully chosen, results in a better performing model. This function applies a polynomial decay function to a provided initial learning\_rate to reach an end\_learning\_rate in the given decay\_steps.

It requires a global\_step value to compute the decayed learning rate. You can just pass a TensorFlow variable that you increment at each training step.

The function returns the decayed learning rate. It is computed as:

global\_step = min(global\_step, decay\_steps)  
decayed\_learning\_rate = (learning\_rate - end\_learning\_rate) \*  
                        (1 - global\_step / decay\_steps) ^ (power) +  
                        end\_learning\_rate

If cycle is True then a multiple of decay\_steps is used, the first one that is bigger than global\_steps.

decay\_steps = decay\_steps \* ceil(global\_step / decay\_steps)  
decayed\_learning\_rate = (learning\_rate - end\_learning\_rate) \*  
                        (1 - global\_step / decay\_steps) ^ (power) +  
                        end\_learning\_rate

Example: decay from 0.1 to 0.01 in 10000 steps using sqrt (i.e. power=0.5):

...  
global\_step = tf.Variable(0, trainable=False)  
starter\_learning\_rate = 0.1  
end\_learning\_rate = 0.01  
decay\_steps = 10000  
learning\_rate = tf.compat.v1.train.polynomial\_decay(starter\_learning\_rate,  
global\_step,  
                                          decay\_steps, end\_learning\_rate,  
                                          power=0.5)  
# Passing global\_step to minimize() will increment it at each step.  
learning\_step = (  
    tf.compat.v1.train.GradientDescentOptimizer(learning\_rate)  
    .minimize(...my loss..., global\_step=global\_step)  
)

#### Args:

* **learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The initial learning rate.
* **global\_step**: A scalar int32 or int64 Tensor or a Python number. Global step to use for the decay computation. Must not be negative.
* **decay\_steps**: A scalar int32 or int64 Tensor or a Python number. Must be positive. See the decay computation above.
* **end\_learning\_rate**: A scalar float32 or float64 Tensor or a Python number. The minimal end learning rate.
* **power**: A scalar float32 or float64 Tensor or a Python number. The power of the polynomial. Defaults to linear, 1.0.
* **cycle**: A boolean, whether or not it should cycle beyond decay\_steps.
* **name**: String. Optional name of the operation. Defaults to 'PolynomialDecay'.

#### Returns:

A scalar Tensor of the same type as learning\_rate. The decayed learning rate.

#### Raises:

* **ValueError**: if global\_step is not supplied.

#### Eager Compatibility

When eager execution is enabled, this function returns a function which in turn returns the decayed learning rate Tensor. This can be useful for changing the learning rate value across different invocations of optimizer functions.

# tf.compat.v1.train.ProximalAdagradOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer#top_of_page)
* [Class ProximalAdagradOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer#class_proximaladagradoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalAdagradOptimizer#apply_gradients)

## Class ProximalAdagradOptimizer

Optimizer that implements the Proximal Adagrad algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/training/proximal\_adagrad.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/proximal_adagrad.py).

See this [paper](http://papers.nips.cc/paper/3793-efficient-learning-using-forward-backward-splitting.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    initial\_accumulator\_value=0.1,  
    l1\_regularization\_strength=0.0,  
    l2\_regularization\_strength=0.0,  
    use\_locking=False,  
    name='ProximalAdagrad'  
)

Construct a new ProximalAdagrad optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **initial\_accumulator\_value**: A floating point value. Starting value for the accumulators, must be positive.
* **l1\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **l2\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "Adagrad".

#### Raises:

* **ValueError**: If the initial\_accumulator\_value is invalid.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.ProximalGradientDescentOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer#top_of_page)
* [Class ProximalGradientDescentOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer#class_proximalgradientdescentoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/ProximalGradientDescentOptimizer#apply_gradients)

## Class ProximalGradientDescentOptimizer

Optimizer that implements the proximal gradient descent algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/training/proximal\_gradient\_descent.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/proximal_gradient_descent.py).

See this [paper](http://papers.nips.cc/paper/3793-efficient-learning-using-forward-backward-splitting.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    l1\_regularization\_strength=0.0,  
    l2\_regularization\_strength=0.0,  
    use\_locking=False,  
    name='ProximalGradientDescent'  
)

Construct a new proximal gradient descent optimizer.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate to use.
* **l1\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **l2\_regularization\_strength**: A float value, must be greater than or equal to zero.
* **use\_locking**: If True use locks for update operations.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "GradientDescent".

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.QueueRunner

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner#top_of_page)
* [Class QueueRunner](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner#class_queuerunner)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/QueueRunner#properties)

## Class QueueRunner

Holds a list of enqueue operations for a queue, each to be run in a thread.

### Aliases:

* Class tf.compat.v1.train.QueueRunner
* Class tf.compat.v1.train.queue\_runner.QueueRunner

Defined in [python/training/queue\_runner\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/queue_runner_impl.py).

Queues are a convenient TensorFlow mechanism to compute tensors asynchronously using multiple threads. For example in the canonical 'Input Reader' setup one set of threads generates filenames in a queue; a second set of threads read records from the files, processes them, and enqueues tensors on a second queue; a third set of threads dequeues these input records to construct batches and runs them through training operations.

There are several delicate issues when running multiple threads that way: closing the queues in sequence as the input is exhausted, correctly catching and reporting exceptions, etc.

The QueueRunner, combined with the Coordinator, helps handle these issues.

#### Eager Compatibility

QueueRunners are not compatible with eager execution. Instead, please use [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) to get data into your model.

## \_\_init\_\_

\_\_init\_\_(  
    queue=None,  
    enqueue\_ops=None,  
    close\_op=None,  
    cancel\_op=None,  
    queue\_closed\_exception\_types=None,  
    queue\_runner\_def=None,  
    import\_scope=None  
)

Create a QueueRunner. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: To construct input pipelines, use the [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) module.

On construction the QueueRunner adds an op to close the queue. That op will be run if the enqueue ops raise exceptions.

When you later call the create\_threads() method, the QueueRunner will create one thread for each op in enqueue\_ops. Each thread will run its enqueue op in parallel with the other threads. The enqueue ops do not have to all be the same op, but it is expected that they all enqueue tensors inqueue.

#### Args:

* **queue**: A Queue.
* **enqueue\_ops**: List of enqueue ops to run in threads later.
* **close\_op**: Op to close the queue. Pending enqueue ops are preserved.
* **cancel\_op**: Op to close the queue and cancel pending enqueue ops.
* **queue\_closed\_exception\_types**: Optional tuple of Exception types that indicate that the queue has been closed when raised during an enqueue operation. Defaults to (tf.errors.OutOfRangeError,). Another common case includes (tf.errors.OutOfRangeError, tf.errors.CancelledError), when some of the enqueue ops may dequeue from other Queues.
* **queue\_runner\_def**: Optional QueueRunnerDef protocol buffer. If specified, recreates the QueueRunner from its contents. queue\_runner\_def and the other arguments are mutually exclusive.
* **import\_scope**: Optional string. Name scope to add. Only used when initializing from protocol buffer.

#### Raises:

* **ValueError**: If both queue\_runner\_def and queue are both specified.
* **ValueError**: If queue or enqueue\_ops are not provided when not restoring from queue\_runner\_def.
* **RuntimeError**: If eager execution is enabled.

## Properties

### cancel\_op

### close\_op

### enqueue\_ops

### exceptions\_raised

Exceptions raised but not handled by the QueueRunner threads.

Exceptions raised in queue runner threads are handled in one of two ways depending on whether or not a Coordinator was passed to create\_threads():

* With a Coordinator, exceptions are reported to the coordinator and forgotten by the QueueRunner.
* Without a Coordinator, exceptions are captured by the QueueRunner and made available in this exceptions\_raised property.

#### Returns:

A list of Python Exception objects. The list is empty if no exception was captured. (No exceptions are captured when using a Coordinator.)

### name

The string name of the underlying Queue.

### queue

### queue\_closed\_exception\_types

## Methods

### create\_threads

create\_threads(  
    sess,  
    coord=None,  
    daemon=False,  
    start=False  
)

Create threads to run the enqueue ops for the given session.

This method requires a session in which the graph was launched. It creates a list of threads, optionally starting them. There is one thread for each op passed in enqueue\_ops.

The coord argument is an optional coordinator that the threads will use to terminate together and report exceptions. If a coordinator is given, this method starts an additional thread to close the queue when the coordinator requests a stop.

If previously created threads for the given session are still running, no new threads will be created.

#### Args:

* **sess**: A Session.
* **coord**: Optional Coordinator object for reporting errors and checking stop conditions.
* **daemon**: Boolean. If True make the threads daemon threads.
* **start**: Boolean. If True starts the threads. If False the caller must call the start() method of the returned threads.

#### Returns:

A list of threads.

### from\_proto

@staticmethod  
from\_proto(  
    queue\_runner\_def,  
    import\_scope=None  
)

Returns a QueueRunner object created from queue\_runner\_def.

### to\_proto

to\_proto(export\_scope=None)

Converts this QueueRunner to a QueueRunnerDef protocol buffer.

#### Args:

* **export\_scope**: Optional string. Name scope to remove.

#### Returns:

A QueueRunnerDef protocol buffer, or None if the Variable is not in the specified name scope.

# tf.compat.v1.train.range\_input\_producer

Produces the integers from 0 to limit-1 in a queue. (deprecated)

tf.compat.v1.train.range\_input\_producer(  
    limit,  
    num\_epochs=None,  
    shuffle=True,  
    seed=None,  
    capacity=32,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.range(limit).shuffle(limit).repeat(num\_epochs)**. If **shuffle=False**, omit the **.shuffle(...)**.**Note:** if **num\_epochs** is not **None**, this function creates local counter **epochs**. Use **local\_variables\_initializer()** to initialize local variables.

#### Args:

* **limit**: An int32 scalar tensor.
* **num\_epochs**: An integer (optional). If specified, range\_input\_producer produces each integer num\_epochs times before generating an OutOfRange error. If not specified, range\_input\_producer can cycle through the integers an unlimited number of times.
* **shuffle**: Boolean. If true, the integers are randomly shuffled within each epoch.
* **seed**: An integer (optional). Seed used if shuffle == True.
* **capacity**: An integer. Sets the queue capacity.
* **shared\_name**: (optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: A name for the operations (optional).

#### Returns:

A Queue with the output integers. A QueueRunner for the Queue is added to the current Graph's QUEUE\_RUNNER collection.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.remove\_checkpoint

Removes a checkpoint given by checkpoint\_prefix. (deprecated)

tf.compat.v1.train.remove\_checkpoint(  
    checkpoint\_prefix,  
    checkpoint\_format\_version=tf.train.SaverDef.V2,  
    meta\_graph\_suffix='meta'  
)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use standard file APIs to delete files with this prefix.

#### Args:

* **checkpoint\_prefix**: The prefix of a V1 or V2 checkpoint. Typically the result of Saver.save()or that of tf.train.latest\_checkpoint(), regardless of sharded/non-sharded or V1/V2.
* **checkpoint\_format\_version**: SaverDef.CheckpointFormatVersion, defaults toSaverDef.V2.
* **meta\_graph\_suffix**: Suffix for MetaGraphDef file. Defaults to 'meta'.

# tf.compat.v1.train.replica\_device\_setter

Return a device function to use when building a Graph for replicas.

tf.compat.v1.train.replica\_device\_setter(  
    ps\_tasks=0,  
    ps\_device='/job:ps',  
    worker\_device='/job:worker',  
    merge\_devices=True,  
    cluster=None,  
    ps\_ops=None,  
    ps\_strategy=None  
)

Defined in [python/training/device\_setter.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/device_setter.py).

Device Functions are used in with tf.device(device\_function): statement to automatically assign devices to Operation objects as they are constructed, Device constraints are added from the inner-most context first, working outwards. The merging behavior adds constraints to fields that are yet unset by a more inner context. Currently the fields are (job, task, cpu/gpu).

If cluster is None, and ps\_tasks is 0, the returned function is a no-op. Otherwise, the value of ps\_tasks is derived from cluster.

By default, only Variable ops are placed on ps tasks, and the placement strategy is round-robin over all ps tasks. A custom ps\_strategy may be used to do more intelligent placement, such astf.contrib.training.GreedyLoadBalancingStrategy.

For example,

# To build a cluster with two ps jobs on hosts ps0 and ps1, and 3 worker  
# jobs on hosts worker0, worker1 and worker2.  
cluster\_spec = {  
    "ps": ["ps0:2222", "ps1:2222"],  
    "worker": ["worker0:2222", "worker1:2222", "worker2:2222"]}  
with  
tf.device(tf.compat.v1.train.replica\_device\_setter(cluster=cluster\_spec)):  
  # Build your graph  
  v1 = tf.Variable(...)  # assigned to /job:ps/task:0  
  v2 = tf.Variable(...)  # assigned to /job:ps/task:1  
  v3 = tf.Variable(...)  # assigned to /job:ps/task:0  
# Run compute

#### Args:

* **ps\_tasks**: Number of tasks in the ps job. Ignored if cluster is provided.
* **ps\_device**: String. Device of the ps job. If empty no ps job is used. Defaults to ps.
* **worker\_device**: String. Device of the worker job. If empty no worker job is used.
* **merge\_devices**: Boolean. If True, merges or only sets a device if the device constraint is completely unset. merges device specification rather than overriding them.
* **cluster**: ClusterDef proto or ClusterSpec.
* **ps\_ops**: List of strings representing Operation types that need to be placed on ps devices. If None, defaults to STANDARD\_PS\_OPS.
* **ps\_strategy**: A callable invoked for every ps Operation (i.e. matched by ps\_ops), that takes the Operation and returns the ps task index to use. If None, defaults to a round-robin strategy across all ps devices.

#### Returns:

A function to pass to tf.device().

#### Raises:

TypeError if cluster is not a dictionary or ClusterDef protocol buffer, or if ps\_strategy is provided but not a callable.

# tf.compat.v1.train.RMSPropOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer#top_of_page)
* [Class RMSPropOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer#class_rmspropoptimizer)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer#methods)
  + [apply\_gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/RMSPropOptimizer#apply_gradients)

## Class RMSPropOptimizer

Optimizer that implements the RMSProp algorithm.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

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

See the [paper](http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf).

## \_\_init\_\_

\_\_init\_\_(  
    learning\_rate,  
    decay=0.9,  
    momentum=0.0,  
    epsilon=1e-10,  
    use\_locking=False,  
    centered=False,  
    name='RMSProp'  
)

Construct a new RMSProp optimizer.

Note that in the dense implementation of this algorithm, variables and their corresponding accumulators (momentum, gradient moving average, square gradient moving average) will be updated even if the gradient is zero (i.e. accumulators will decay, momentum will be applied). The sparse implementation (used when the gradient is an IndexedSlices object, typically because of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) or an embedding lookup in the forward pass) will not update variable slices or their accumulators unless those slices were used in the forward pass (nor is there an "eventual" correction to account for these omitted updates). This leads to more efficient updates for large embedding lookup tables (where most of the slices are not accessed in a particular graph execution), but differs from the published algorithm.

#### Args:

* **learning\_rate**: A Tensor or a floating point value. The learning rate.
* **decay**: Discounting factor for the history/coming gradient
* **momentum**: A scalar tensor.
* **epsilon**: Small value to avoid zero denominator.
* **use\_locking**: If True use locks for update operation.
* **centered**: If True, gradients are normalized by the estimated variance of the gradient; if False, by the uncentered second moment. Setting this to True may help with training, but is slightly more expensive in terms of computation and memory. Defaults to False.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "RMSProp".

#### Eager Compatibility

When eager execution is enabled, learning\_rate, decay, momentum, and epsilon can each be a callable that takes no arguments and returns the actual value to use. This can be useful for changing these values across different invocations of optimizer functions.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This is the second part of minimize(). It returns an Operation that applies gradients.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizerconstructor.

#### Returns:

An Operation that applies the specified gradients. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **TypeError**: If grads\_and\_vars is malformed.
* **ValueError**: If none of the variables have gradients.
* **RuntimeError**: If you should use \_distributed\_apply() instead.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    grad\_loss=None  
)

Compute gradients of loss for the variables in var\_list.

This is the first part of minimize(). It returns a list of (gradient, variable) pairs where "gradient" is the gradient for "variable". Note that "gradient" can be a Tensor, an IndexedSlices, or None if there is no gradient for the given variable.

#### Args:

* **loss**: A Tensor containing the value to minimize or a callable taking no arguments which returns the value to minimize. When eager execution is enabled it must be a callable.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, or GATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

A list of (gradient, variable) pairs. Variable is always present, but gradient can be None.

#### Raises:

* **TypeError**: If var\_list contains anything else than Variable objects.
* **ValueError**: If some arguments are invalid.
* **RuntimeError**: If called with eager execution enabled and loss is not callable.

#### Eager Compatibility

When eager execution is enabled, gate\_gradients, aggregation\_method, and colocate\_gradients\_with\_ops are ignored.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    var,  
    name  
)

Return a slot named name created for var by the Optimizer.

Some Optimizer subclasses use additional variables. For example Momentum and Adagrad use variables to accumulate updates. This method gives access to these Variable objects if for some reason you need them.

Use get\_slot\_names() to get the list of slot names created by the Optimizer.

#### Args:

* **var**: A variable passed to minimize() or apply\_gradients().
* **name**: A string.

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names()

Return a list of the names of slots created by the Optimizer.

See get\_slot().

#### Returns:

A list of strings.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

A list of variables which encode the current state of Optimizer.

Includes slot variables and additional global variables created by the optimizer in the current default graph.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.Saver

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver#top_of_page)
* [Class Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver#class_saver)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver#properties)
  + [last\_checkpoints](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver#last_checkpoints)

## Class Saver

Saves and restores variables.

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

See [Variables](https://tensorflow.org/guide/variables) for an overview of variables, saving and restoring.

The Saver class adds ops to save and restore variables to and from checkpoints. It also provides convenience methods to run these ops.

Checkpoints are binary files in a proprietary format which map variable names to tensor values. The best way to examine the contents of a checkpoint is to load it using a Saver.

Savers can automatically number checkpoint filenames with a provided counter. This lets you keep multiple checkpoints at different steps while training a model. For example you can number the checkpoint filenames with the training step number. To avoid filling up disks, savers manage checkpoint files automatically. For example, they can keep only the N most recent files, or one checkpoint for every N hours of training.

You number checkpoint filenames by passing a value to the optional global\_step argument to save():

saver.save(sess, 'my-model', global\_step=0) ==> filename: 'my-model-0'  
...  
saver.save(sess, 'my-model', global\_step=1000) ==> filename: 'my-model-1000'

Additionally, optional arguments to the Saver() constructor let you control the proliferation of checkpoint files on disk:

* max\_to\_keep indicates the maximum number of recent checkpoint files to keep. As new files are created, older files are deleted. If None or 0, no checkpoints are deleted from the filesystem but only the last one is kept in the checkpoint file. Defaults to 5 (that is, the 5 most recent checkpoint files are kept.)
* keep\_checkpoint\_every\_n\_hours: In addition to keeping the most recent max\_to\_keepcheckpoint files, you might want to keep one checkpoint file for every N hours of training. This can be useful if you want to later analyze how a model progressed during a long training session. For example, passing keep\_checkpoint\_every\_n\_hours=2 ensures that you keep one checkpoint file for every 2 hours of training. The default value of 10,000 hours effectively disables the feature.

Note that you still have to call the save() method to save the model. Passing these arguments to the constructor will not save variables automatically for you.

A training program that saves regularly looks like:

...  
# Create a saver.  
saver = tf.compat.v1.train.Saver(...variables...)  
# Launch the graph and train, saving the model every 1,000 steps.  
sess = tf.compat.v1.Session()  
for step in xrange(1000000):  
    sess.run(..training\_op..)  
    if step % 1000 == 0:  
        # Append the step number to the checkpoint name:  
        saver.save(sess, 'my-model', global\_step=step)

In addition to checkpoint files, savers keep a protocol buffer on disk with the list of recent checkpoints. This is used to manage numbered checkpoint files and by latest\_checkpoint(), which makes it easy to discover the path to the most recent checkpoint. That protocol buffer is stored in a file named 'checkpoint' next to the checkpoint files.

If you create several savers, you can specify a different filename for the protocol buffer file in the call to save().

## \_\_init\_\_

\_\_init\_\_(  
    var\_list=None,  
    reshape=False,  
    sharded=False,  
    max\_to\_keep=5,  
    keep\_checkpoint\_every\_n\_hours=10000.0,  
    name=None,  
    restore\_sequentially=False,  
    saver\_def=None,  
    builder=None,  
    defer\_build=False,  
    allow\_empty=False,  
    write\_version=tf.train.SaverDef.V2,  
    pad\_step\_number=False,  
    save\_relative\_paths=False,  
    filename=None  
)

Creates a Saver.

The constructor adds ops to save and restore variables.

var\_list specifies the variables that will be saved and restored. It can be passed as a dict or a list:

* A dict of names to variables: The keys are the names that will be used to save or restore the variables in the checkpoint files.
* A list of variables: The variables will be keyed with their op name in the checkpoint files.

#### For example:

v1 = tf.Variable(..., name='v1')  
v2 = tf.Variable(..., name='v2')  
  
# Pass the variables as a dict:  
saver = tf.compat.v1.train.Saver({'v1': v1, 'v2': v2})  
  
# Or pass them as a list.  
saver = tf.compat.v1.train.Saver([v1, v2])  
# Passing a list is equivalent to passing a dict with the variable op names  
# as keys:  
saver = tf.compat.v1.train.Saver({v.op.name: v for v in [v1, v2]})

The optional reshape argument, if True, allows restoring a variable from a save file where the variable had a different shape, but the same number of elements and type. This is useful if you have reshaped a variable and want to reload it from an older checkpoint.

The optional sharded argument, if True, instructs the saver to shard checkpoints per device.

#### Args:

* **var\_list**: A list of Variable/SaveableObject, or a dictionary mapping names to SaveableObjects. If None, defaults to the list of all saveable objects.
* **reshape**: If True, allows restoring parameters from a checkpoint where the variables have a different shape.
* **sharded**: If True, shard the checkpoints, one per device.
* **max\_to\_keep**: Maximum number of recent checkpoints to keep. Defaults to 5.
* **keep\_checkpoint\_every\_n\_hours**: How often to keep checkpoints. Defaults to 10,000 hours.
* **name**: String. Optional name to use as a prefix when adding operations.
* **restore\_sequentially**: A Bool, which if true, causes restore of different variables to happen sequentially within each device. This can lower memory usage when restoring very large models.
* **saver\_def**: Optional SaverDef proto to use instead of running the builder. This is only useful for specialty code that wants to recreate a Saver object for a previously built Graph that had a Saver. The saver\_def proto should be the one returned by the as\_saver\_def() call of the Saver that was created for that Graph.
* **builder**: Optional SaverBuilder to use if a saver\_def was not provided. Defaults to BulkSaverBuilder().
* **defer\_build**: If True, defer adding the save and restore ops to the build() call. In that case build() should be called before finalizing the graph or using the saver.
* **allow\_empty**: If False (default) raise an error if there are no variables in the graph. Otherwise, construct the saver anyway and make it a no-op.
* **write\_version**: controls what format to use when saving checkpoints. It also affects certain filepath matching logic. The V2 format is the recommended choice: it is much more optimized than V1 in terms of memory required and latency incurred during restore. Regardless of this flag, the Saver is able to restore from both V2 and V1 checkpoints.
* **pad\_step\_number**: if True, pads the global step number in the checkpoint filepaths to some fixed width (8 by default). This is turned off by default.
* **save\_relative\_paths**: If True, will write relative paths to the checkpoint state file. This is needed if the user wants to copy the checkpoint directory and reload from the copied directory.
* **filename**: If known at graph construction time, filename used for variable loading/saving.

#### Raises:

* **TypeError**: If var\_list is invalid.
* **ValueError**: If any of the keys or values in var\_list are not unique.
* **RuntimeError**: If eager execution is enabled andvar\_list does not specify a list of varialbes to save.

#### Eager Compatibility

When eager execution is enabled, var\_list must specify a list or dict of variables to save. Otherwise, a RuntimeError will be raised.

Although Saver works in some cases when executing eagerly, it is fragile. Please switch to [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) or [tf.keras.Model.save\_weights](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#save_weights), which perform a more robust object-based saving. These APIs will load checkpoints written by Saver.

## Properties

### last\_checkpoints

List of not-yet-deleted checkpoint filenames.

You can pass any of the returned values to restore().

#### Returns:

A list of checkpoint filenames, sorted from oldest to newest.

## Methods

### as\_saver\_def

as\_saver\_def()

Generates a SaverDef representation of this saver.

#### Returns:

A SaverDef proto.

### build

build()

### export\_meta\_graph

export\_meta\_graph(  
    filename=None,  
    collection\_list=None,  
    as\_text=False,  
    export\_scope=None,  
    clear\_devices=False,  
    clear\_extraneous\_savers=False,  
    strip\_default\_attrs=False,  
    save\_debug\_info=False  
)

Writes MetaGraphDef to save\_path/filename.

#### Args:

* **filename**: Optional meta\_graph filename including the path.
* **collection\_list**: List of string keys to collect.
* **as\_text**: If True, writes the meta\_graph as an ASCII proto.
* **export\_scope**: Optional string. Name scope to remove.
* **clear\_devices**: Whether or not to clear the device field for an Operation or Tensor during export.
* **clear\_extraneous\_savers**: Remove any Saver-related information from the graph (both Save/Restore ops and SaverDefs) that are not associated with this Saver.
* **strip\_default\_attrs**: Boolean. If True, default-valued attributes will be removed from the NodeDefs. For a detailed guide, see [Stripping Default-Valued Attributes](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md#stripping-default-valued-attributes).
* **save\_debug\_info**: If True, save the GraphDebugInfo to a separate file, which in the same directory of filename and with \_debug added before the file extension.

#### Returns:

A MetaGraphDef proto.

### from\_proto

@staticmethod  
from\_proto(  
    saver\_def,  
    import\_scope=None  
)

Returns a Saver object created from saver\_def.

#### Args:

* **saver\_def**: a SaverDef protocol buffer.
* **import\_scope**: Optional string. Name scope to use.

#### Returns:

A Saver built from saver\_def.

### recover\_last\_checkpoints

recover\_last\_checkpoints(checkpoint\_paths)

Recovers the internal saver state after a crash.

This method is useful for recovering the "self.\_last\_checkpoints" state.

Globs for the checkpoints pointed to by checkpoint\_paths. If the files exist, use their mtime as the checkpoint timestamp.

#### Args:

* **checkpoint\_paths**: a list of checkpoint paths.

### restore

restore(  
    sess,  
    save\_path  
)

Restores previously saved variables.

This method runs the ops added by the constructor for restoring variables. It requires a session in which the graph was launched. The variables to restore do not have to have been initialized, as restoring is itself a way to initialize variables.

The save\_path argument is typically a value previously returned from a save() call, or a call to latest\_checkpoint().

#### Args:

* **sess**: A Session to use to restore the parameters. None in eager mode.
* **save\_path**: Path where parameters were previously saved.

#### Raises:

* **ValueError**: If save\_path is None or not a valid checkpoint.

### save

save(  
    sess,  
    save\_path,  
    global\_step=None,  
    latest\_filename=None,  
    meta\_graph\_suffix='meta',  
    write\_meta\_graph=True,  
    write\_state=True,  
    strip\_default\_attrs=False,  
    save\_debug\_info=False  
)

Saves variables.

This method runs the ops added by the constructor for saving variables. It requires a session in which the graph was launched. The variables to save must also have been initialized.

The method returns the path prefix of the newly created checkpoint files. This string can be passed directly to a call to restore().

#### Args:

* **sess**: A Session to use to save the variables.
* **save\_path**: String. Prefix of filenames created for the checkpoint.
* **global\_step**: If provided the global step number is appended to save\_path to create the checkpoint filenames. The optional argument can be a Tensor, a Tensor name or an integer.
* **latest\_filename**: Optional name for the protocol buffer file that will contains the list of most recent checkpoints. That file, kept in the same directory as the checkpoint files, is automatically managed by the saver to keep track of recent checkpoints. Defaults to 'checkpoint'.
* **meta\_graph\_suffix**: Suffix for MetaGraphDef file. Defaults to 'meta'.
* **write\_meta\_graph**: Boolean indicating whether or not to write the meta graph file.
* **write\_state**: Boolean indicating whether or not to write the CheckpointStateProto.
* **strip\_default\_attrs**: Boolean. If True, default-valued attributes will be removed from the NodeDefs. For a detailed guide, see [Stripping Default-Valued Attributes](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md#stripping-default-valued-attributes).
* **save\_debug\_info**: If True, save the GraphDebugInfo to a separate file, which in the same directory of save\_path and with \_debug added before the file extension. This is only enabled when write\_meta\_graph is True

#### Returns:

A string: path prefix used for the checkpoint files. If the saver is sharded, this string ends with: '-?????-of-nnnnn' where 'nnnnn' is the number of shards created. If the saver is empty, returns None.

#### Raises:

* **TypeError**: If sess is not a Session.
* **ValueError**: If latest\_filename contains path components, or if it collides with save\_path.
* **RuntimeError**: If save and restore ops weren't built.

### set\_last\_checkpoints

set\_last\_checkpoints(last\_checkpoints)

DEPRECATED: Use set\_last\_checkpoints\_with\_time.

Sets the list of old checkpoint filenames.

#### Args:

* **last\_checkpoints**: A list of checkpoint filenames.

#### Raises:

* **AssertionError**: If last\_checkpoints is not a list.

### set\_last\_checkpoints\_with\_time

set\_last\_checkpoints\_with\_time(last\_checkpoints\_with\_time)

Sets the list of old checkpoint filenames and timestamps.

#### Args:

* **last\_checkpoints\_with\_time**: A list of tuples of checkpoint filenames and timestamps.

#### Raises:

* **AssertionError**: If last\_checkpoints\_with\_time is not a list.

### to\_proto

to\_proto(export\_scope=None)

Converts this Saver to a SaverDef protocol buffer.

#### Args:

* **export\_scope**: Optional string. Name scope to remove.

#### Returns:

A SaverDef protocol buffer.

# tf.compat.v1.train.SaverDef

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SaverDef#top_of_page)
* [Class SaverDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SaverDef#class_saverdef)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SaverDef#properties)
  + [filename\_tensor\_name](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SaverDef#filename_tensor_name)
  + [keep\_checkpoint\_every\_n\_hours](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SaverDef#keep_checkpoint_every_n_hours)

## Class SaverDef

Defined in [core/protobuf/saver.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/protobuf/saver.proto).

## Properties

### filename\_tensor\_name

string filename\_tensor\_name

### keep\_checkpoint\_every\_n\_hours

float keep\_checkpoint\_every\_n\_hours

### max\_to\_keep

int32 max\_to\_keep

### restore\_op\_name

string restore\_op\_name

### save\_tensor\_name

string save\_tensor\_name

### sharded

bool sharded

### version

CheckpointFormatVersion version

## Class Members

* CheckpointFormatVersion
* LEGACY = 0
* V1 = 1
* V2 = 2

# tf.compat.v1.train.Scaffold

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold#top_of_page)
* [Class Scaffold](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold#class_scaffold)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Scaffold#properties)

## Class Scaffold

Structure to create or gather pieces commonly needed to train a model.

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

### Used in the guide:

* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

When you build a model for training you usually need ops to initialize variables, a Saver to checkpoint them, an op to collect summaries for the visualizer, and so on.

Various libraries built on top of the core TensorFlow library take care of creating some or all of these pieces and storing them in well known collections in the graph. The Scaffold class helps pick these pieces from the graph collections, creating and adding them to the collections if needed.

If you call the scaffold constructor without any arguments, it will pick pieces from the collections, creating default ones if needed when scaffold.finalize() is called. You can pass arguments to the constructor to provide your own pieces. Pieces that you pass to the constructor are not added to the graph collections.

The following pieces are directly accessible as attributes of the Scaffold object:

* saver: A [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) object taking care of saving the variables. Picked from and stored into the SAVERS collection in the graph by default.
* init\_op: An op to run to initialize the variables. Picked from and stored into the INIT\_OPcollection in the graph by default.
* ready\_op: An op to verify that the variables are initialized. Picked from and stored into the READY\_OP collection in the graph by default.
* ready\_for\_local\_init\_op: An op to verify that global state has been initialized and it is alright to run local\_init\_op. Picked from and stored into the READY\_FOR\_LOCAL\_INIT\_OP collection in the graph by default. This is needed when the initialization of local variables depends on the values of global variables.
* local\_init\_op: An op to initialize the local variables. Picked from and stored into the LOCAL\_INIT\_OP collection in the graph by default.
* summary\_op: An op to run and merge the summaries in the graph. Picked from and stored into the SUMMARY\_OP collection in the graph by default.

You can also pass the following additional pieces to the constructor:

* init\_feed\_dict: A session feed dictionary that should be used when running the init op.
* init\_fn: A callable to run after the init op to perform additional initializations. The callable will be called as init\_fn(scaffold, session).

## \_\_init\_\_

\_\_init\_\_(  
    init\_op=None,  
    init\_feed\_dict=None,  
    init\_fn=None,  
    ready\_op=None,  
    ready\_for\_local\_init\_op=None,  
    local\_init\_op=None,  
    summary\_op=None,  
    saver=None,  
    copy\_from\_scaffold=None  
)

Create a scaffold.

#### Args:

* **init\_op**: Optional op for initializing variables.
* **init\_feed\_dict**: Optional session feed dictionary to use when running the init\_op.
* **init\_fn**: Optional function to use to initialize the model after running the init\_op. Will be called as init\_fn(scaffold, session).
* **ready\_op**: Optional op to verify that the variables are initialized. Must return an empty 1D string tensor when the variables are initialized, or a non-empty 1D string tensor listing the names of the non-initialized variables.
* **ready\_for\_local\_init\_op**: Optional op to verify that the global variables are initialized and local\_init\_op can be run. Must return an empty 1D string tensor when the global variables are initialized, or a non-empty 1D string tensor listing the names of the non-initialized global variables.
* **local\_init\_op**: Optional op to initialize local variables.
* **summary\_op**: Optional op to gather all summaries. Must return a scalar string tensor containing a serialized Summary proto.
* **saver**: Optional [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) object to use to save and restore variables. May also be a [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) object, in which case object-based checkpoints are saved. This will also load some object-based checkpoints saved from elsewhere, but that loading may be fragile since it uses fixed keys rather than performing a full graph-based match. For example if a variable has two paths from the Checkpoint object because two Model objects share the Layer object that owns it, removing one Model may change the keys and break checkpoint loading through this API, whereas a graph-based match would match the variable through the other Model.
* **copy\_from\_scaffold**: Optional scaffold object to copy fields from. Its fields will be overwritten by the provided fields in this function.

## Properties

### init\_feed\_dict

### init\_fn

### init\_op

### local\_init\_op

### ready\_for\_local\_init\_op

### ready\_op

### saver

### summary\_op

## Methods

### default\_local\_init\_op

@staticmethod  
default\_local\_init\_op()

Returns an op that groups the default local init ops.

This op is used during session initialization when a Scaffold is initialized without specifying the local\_init\_op arg. It includes [tf.compat.v1.local\_variables\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/local_variables_initializer),[tf.compat.v1.tables\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tables_initializer), and also initializes local session resources.

#### Returns:

The default Scaffold local init op.

### finalize

finalize()

Creates operations if needed and finalizes the graph.

### get\_or\_default

@staticmethod  
get\_or\_default(  
    arg\_name,  
    collection\_key,  
    default\_constructor  
)

Get from cache or create a default operation.

# tf.compat.v1.train.sdca\_fprint

Computes fingerprints of the input strings.

tf.compat.v1.train.sdca\_fprint(  
    input,  
    name=None  
)

Defined in generated file: python/ops/gen\_sdca\_ops.py.

#### Args:

* **input**: A Tensor of type string. vector of strings to compute fingerprints on.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.compat.v1.train.sdca\_optimizer

Distributed version of Stochastic Dual Coordinate Ascent (SDCA) optimizer for

tf.compat.v1.train.sdca\_optimizer(  
    sparse\_example\_indices,  
    sparse\_feature\_indices,  
    sparse\_feature\_values,  
    dense\_features,  
    example\_weights,  
    example\_labels,  
    sparse\_indices,  
    sparse\_weights,  
    dense\_weights,  
    example\_state\_data,  
    loss\_type,  
    l1,  
    l2,  
    num\_loss\_partitions,  
    num\_inner\_iterations,  
    adaptative=True,  
    name=None  
)

Defined in generated file: python/ops/gen\_sdca\_ops.py.

linear models with L1 + L2 regularization. As global optimization objective is strongly-convex, the optimizer optimizes the dual objective at each step. The optimizer applies each update one example at a time. Examples are sampled uniformly, and the optimizer is learning rate free and enjoys linear convergence rate.

[Proximal Stochastic Dual Coordinate Ascent](http://arxiv.org/pdf/1211.2717v1.pdf).  
Shai Shalev-Shwartz, Tong Zhang. 2012

LossObjective=∑fi(wxi)+(l2/2)∗|w|2+l1∗|w|

[Adding vs. Averaging in Distributed Primal-Dual Optimization](http://arxiv.org/abs/1502.03508).  
Chenxin Ma, Virginia Smith, Martin Jaggi, Michael I. Jordan, Peter Richtarik, Martin Takac. 2015

[Stochastic Dual Coordinate Ascent with Adaptive Probabilities](https://arxiv.org/abs/1502.08053).  
Dominik Csiba, Zheng Qu, Peter Richtarik. 2015

#### Args:

* **sparse\_example\_indices**: A list of Tensor objects with type int64. a list of vectors which contain example indices.
* **sparse\_feature\_indices**: A list with the same length as sparse\_example\_indices of Tensor objects with type int64. a list of vectors which contain feature indices.
* **sparse\_feature\_values**: A list of Tensor objects with type float32. a list of vectors which contains feature value associated with each feature group.
* **dense\_features**: A list of Tensor objects with type float32. a list of matrices which contains the dense feature values.
* **example\_weights**: A Tensor of type float32. a vector which contains the weight associated with each example.
* **example\_labels**: A Tensor of type float32. a vector which contains the label/target associated with each example.
* **sparse\_indices**: A list with the same length as sparse\_example\_indices of Tensor objects with type int64. a list of vectors where each value is the indices which has corresponding weights in sparse\_weights. This field maybe omitted for the dense approach.
* **sparse\_weights**: A list with the same length as sparse\_example\_indices of Tensor objects with type float32. a list of vectors where each value is the weight associated with a sparse feature group.
* **dense\_weights**: A list with the same length as dense\_features of Tensor objects with type float32. a list of vectors where the values are the weights associated with a dense feature group.
* **example\_state\_data**: A Tensor of type float32. a list of vectors containing the example state data.
* **loss\_type**: A string from: "logistic\_loss", "squared\_loss", "hinge\_loss", "smooth\_hinge\_loss", "poisson\_loss". Type of the primal loss. Currently SdcaSolver supports logistic, squared and hinge losses.
* **l1**: A float. Symmetric l1 regularization strength.
* **l2**: A float. Symmetric l2 regularization strength.
* **num\_loss\_partitions**: An int that is >= 1. Number of partitions of the global loss function.
* **num\_inner\_iterations**: An int that is >= 1. Number of iterations per mini-batch.
* **adaptative**: An optional bool. Defaults to True. Whether to use Adaptive SDCA for the inner loop.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (out\_example\_state\_data, out\_delta\_sparse\_weights, out\_delta\_dense\_weights).

* **out\_example\_state\_data**: A Tensor of type float32.
* **out\_delta\_sparse\_weights**: A list with the same length as sparse\_example\_indices of Tensor objects with type float32.
* **out\_delta\_dense\_weights**: A list with the same length as dense\_features of Tensorobjects with type float32.

# tf.compat.v1.train.sdca\_shrink\_l1

Applies L1 regularization shrink step on the parameters.

tf.compat.v1.train.sdca\_shrink\_l1(  
    weights,  
    l1,  
    l2,  
    name=None  
)

Defined in generated file: python/ops/gen\_sdca\_ops.py.

#### Args:

* **weights**: A list of Tensor objects with type mutable float32. a list of vectors where each value is the weight associated with a feature group.
* **l1**: A float. Symmetric l1 regularization strength.
* **l2**: A float. Symmetric l2 regularization strength. Should be a positive float.
* **name**: A name for the operation (optional).

#### Returns:

The created Operation.

# tf.compat.v1.train.SessionCreator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator#top_of_page)
* [Class SessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator#class_sessioncreator)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator#methods)
  + [create\_session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator#create_session)

## Class SessionCreator

A factory for tf.Session.

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

## Methods

### create\_session

create\_session()

# tf.compat.v1.train.SessionManager

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager#top_of_page)
* [Class SessionManager](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager#class_sessionmanager)
  + [Usage:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager#usage)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionManager#methods)

## Class SessionManager

Training helper that restores from checkpoint and creates session.

Defined in [python/training/session\_manager.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/session_manager.py).

This class is a small wrapper that takes care of session creation and checkpoint recovery. It also provides functions that to facilitate coordination among multiple training threads or processes.

* Checkpointing trained variables as the training progresses.
* Initializing variables on startup, restoring them from the most recent checkpoint after a crash, or wait for checkpoints to become available.

### Usage:

with tf.Graph().as\_default():  
   ...add operations to the graph...  
  # Create a SessionManager that will checkpoint the model in '/tmp/mydir'.  
  sm = SessionManager()  
  sess = sm.prepare\_session(master, init\_op, saver, checkpoint\_dir)  
  # Use the session to train the graph.  
  while True:  
    sess.run(<my\_train\_op>)

prepare\_session() initializes or restores a model. It requires init\_op and saver as an argument.

A second process could wait for the model to be ready by doing the following:

with tf.Graph().as\_default():  
   ...add operations to the graph...  
  # Create a SessionManager that will wait for the model to become ready.  
  sm = SessionManager()  
  sess = sm.wait\_for\_session(master)  
  # Use the session to train the graph.  
  while True:  
    sess.run(<my\_train\_op>)

wait\_for\_session() waits for a model to be initialized by other processes.

## \_\_init\_\_

\_\_init\_\_(  
    local\_init\_op=None,  
    ready\_op=None,  
    ready\_for\_local\_init\_op=None,  
    graph=None,  
    recovery\_wait\_secs=30,  
    local\_init\_run\_options=None  
)

Creates a SessionManager.

The local\_init\_op is an Operation that is run always after a new session was created. If None, this step is skipped.

The ready\_op is an Operation used to check if the model is ready. The model is considered ready if that operation returns an empty 1D string tensor. If the operation returns a non empty 1D string tensor, the elements are concatenated and used to indicate to the user why the model is not ready.

The ready\_for\_local\_init\_op is an Operation used to check if the model is ready to run local\_init\_op. The model is considered ready if that operation returns an empty 1D string tensor. If the operation returns a non empty 1D string tensor, the elements are concatenated and used to indicate to the user why the model is not ready.

If ready\_op is None, the model is not checked for readiness.

recovery\_wait\_secs is the number of seconds between checks that the model is ready. It is used by processes to wait for a model to be initialized or restored. Defaults to 30 seconds.

#### Args:

* **local\_init\_op**: An Operation run immediately after session creation. Usually used to initialize tables and local variables.
* **ready\_op**: An Operation to check if the model is initialized.
* **ready\_for\_local\_init\_op**: An Operation to check if the model is ready to run local\_init\_op.
* **graph**: The Graph that the model will use.
* **recovery\_wait\_secs**: Seconds between checks for the model to be ready.
* **local\_init\_run\_options**: RunOptions to be passed to session.run when executing the local\_init\_op.

#### Raises:

* **ValueError**: If ready\_for\_local\_init\_op is not None but local\_init\_op is None

## Methods

### prepare\_session

prepare\_session(  
    master,  
    init\_op=None,  
    saver=None,  
    checkpoint\_dir=None,  
    checkpoint\_filename\_with\_path=None,  
    wait\_for\_checkpoint=False,  
    max\_wait\_secs=7200,  
    config=None,  
    init\_feed\_dict=None,  
    init\_fn=None  
)

Creates a Session. Makes sure the model is ready to be used.

Creates a Session on 'master'. If a saver object is passed in, and checkpoint\_dir points to a directory containing valid checkpoint files, then it will try to recover the model from checkpoint. If no checkpoint files are available, and wait\_for\_checkpoint is True, then the process would check every recovery\_wait\_secs, up to max\_wait\_secs, for recovery to succeed.

If the model cannot be recovered successfully then it is initialized by running the init\_op and calling init\_fn if they are provided. The local\_init\_op is also run after init\_op and init\_fn, regardless of whether the model was recovered successfully, but only if ready\_for\_local\_init\_op passes.

If the model is recovered from a checkpoint it is assumed that all global variables have been initialized, in particular neither init\_op nor init\_fn will be executed.

It is an error if the model cannot be recovered and no init\_op or init\_fn or local\_init\_op are passed.

#### Args:

* **master**: String representation of the TensorFlow master to use.
* **init\_op**: Optional Operation used to initialize the model.
* **saver**: A Saver object used to restore a model.
* **checkpoint\_dir**: Path to the checkpoint files. The latest checkpoint in the dir will be used to restore.
* **checkpoint\_filename\_with\_path**: Full file name path to the checkpoint file.
* **wait\_for\_checkpoint**: Whether to wait for checkpoint to become available.
* **max\_wait\_secs**: Maximum time to wait for checkpoints to become available.
* **config**: Optional ConfigProto proto used to configure the session.
* **init\_feed\_dict**: Optional dictionary that maps Tensor objects to feed values. This feed dictionary is passed to the session run() call when running the init op.
* **init\_fn**: Optional callable used to initialize the model. Called after the optional init\_op is called. The callable must accept one argument, the session being initialized.

#### Returns:

A Session object that can be used to drive the model.

#### Raises:

* **RuntimeError**: If the model cannot be initialized or recovered.
* **ValueError**: If both checkpoint\_dir and checkpoint\_filename\_with\_path are set.

### recover\_session

recover\_session(  
    master,  
    saver=None,  
    checkpoint\_dir=None,  
    checkpoint\_filename\_with\_path=None,  
    wait\_for\_checkpoint=False,  
    max\_wait\_secs=7200,  
    config=None  
)

Creates a Session, recovering if possible.

Creates a new session on 'master'. If the session is not initialized and can be recovered from a checkpoint, recover it.

#### Args:

* **master**: String representation of the TensorFlow master to use.
* **saver**: A Saver object used to restore a model.
* **checkpoint\_dir**: Path to the checkpoint files. The latest checkpoint in the dir will be used to restore.
* **checkpoint\_filename\_with\_path**: Full file name path to the checkpoint file.
* **wait\_for\_checkpoint**: Whether to wait for checkpoint to become available.
* **max\_wait\_secs**: Maximum time to wait for checkpoints to become available.
* **config**: Optional ConfigProto proto used to configure the session.

#### Returns:

A pair (sess, initialized) where 'initialized' is True if the session could be recovered and initialized, False otherwise.

#### Raises:

* **ValueError**: If both checkpoint\_dir and checkpoint\_filename\_with\_path are set.

### wait\_for\_session

wait\_for\_session(  
    master,  
    config=None,  
    max\_wait\_secs=float('Inf')  
)

Creates a new Session and waits for model to be ready.

Creates a new Session on 'master'. Waits for the model to be initialized or recovered from a checkpoint. It's expected that another thread or process will make the model ready, and that this is intended to be used by threads/processes that participate in a distributed training configuration where a different thread/process is responsible for initializing or recovering the model being trained.

NB: The amount of time this method waits for the session is bounded by max\_wait\_secs. By default, this function will wait indefinitely.

#### Args:

* **master**: String representation of the TensorFlow master to use.
* **config**: Optional ConfigProto proto used to configure the session.
* **max\_wait\_secs**: Maximum time to wait for the session to become available.

#### Returns:

A Session. May be None if the operation exceeds the timeout specified by config.operation\_timeout\_in\_ms.

#### Raises:

* **tf.DeadlineExceededError**: if the session is not available after max\_wait\_secs.

# tf.compat.v1.train.shuffle\_batch

Creates batches by randomly shuffling tensors. (deprecated)

tf.compat.v1.train.shuffle\_batch(  
    tensors,  
    batch\_size,  
    capacity,  
    min\_after\_dequeue,  
    num\_threads=1,  
    seed=None,  
    enqueue\_many=False,  
    shapes=None,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.shuffle(min\_after\_dequeue).batch(batch\_size)**.

This function adds the following to the current Graph:

* A shuffling queue into which tensors from tensors are enqueued.
* A dequeue\_many operation to create batches from the queue.
* A QueueRunner to QUEUE\_RUNNER collection, to enqueue the tensors from tensors.

If enqueue\_many is False, tensors is assumed to represent a single example. An input tensor with shape [x, y, z] will be output as a tensor with shape [batch\_size, x, y, z].

If enqueue\_many is True, tensors is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of tensors should have the same size in the first dimension. If an input tensor has shape [\*, x, y, z], the output will have shape [batch\_size, x, y, z].

The capacity argument controls the how long the prefetching is allowed to grow the queues.

The returned operation is a dequeue operation and will throw [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError) if the input queue is exhausted. If this operation is feeding another input queue, its queue runner will catch this exception, however, if this operation is used in your main thread you are responsible for catching this yourself.

#### For example:

# Creates batches of 32 images and 32 labels.  
image\_batch, label\_batch = tf.compat.v1.train.shuffle\_batch(  
      [single\_image, single\_label],  
      batch\_size=32,  
      num\_threads=4,  
      capacity=50000,  
      min\_after\_dequeue=10000)

N.B.: You must ensure that either (i) the shapes argument is passed, or (ii) all of the tensors in tensors must have fully-defined shapes. ValueError will be raised if neither of these conditions holds.

If allow\_smaller\_final\_batch is True, a smaller batch value than batch\_size is returned when the queue is closed and there are not enough elements to fill the batch, otherwise the pending elements are discarded. In addition, all output tensors' static shapes, as accessed via the shapeproperty will have a first Dimension value of None, and operations that depend on fixed batch\_size would fail.

#### Args:

* **tensors**: The list or dictionary of tensors to enqueue.
* **batch\_size**: The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **min\_after\_dequeue**: Minimum number elements in the queue after a dequeue, used to ensure a level of mixing of elements.
* **num\_threads**: The number of threads enqueuing tensor\_list.
* **seed**: Seed for the random shuffling within the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensor\_list.
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (Optional) If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the types as tensors.

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.shuffle\_batch\_join

Create batches by randomly shuffling tensors. (deprecated)

tf.compat.v1.train.shuffle\_batch\_join(  
    tensors\_list,  
    batch\_size,  
    capacity,  
    min\_after\_dequeue,  
    seed=None,  
    enqueue\_many=False,  
    shapes=None,  
    allow\_smaller\_final\_batch=False,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.interleave(...).shuffle(min\_after\_dequeue).batch(batch\_size)**.

The tensors\_list argument is a list of tuples of tensors, or a list of dictionaries of tensors. Each element in the list is treated similarly to the tensors argument of tf.compat.v1.train.shuffle\_batch().

This version enqueues a different list of tensors in different threads. It adds the following to the current Graph:

* A shuffling queue into which tensors from tensors\_list are enqueued.
* A dequeue\_many operation to create batches from the queue.
* A QueueRunner to QUEUE\_RUNNER collection, to enqueue the tensors from tensors\_list.

len(tensors\_list) threads will be started, with thread i enqueuing the tensors from tensors\_list[i]. tensors\_list[i1][j] must match tensors\_list[i2][j] in type and shape, except in the first dimension if enqueue\_many is true.

If enqueue\_many is False, each tensors\_list[i] is assumed to represent a single example. An input tensor with shape [x, y, z] will be output as a tensor with shape [batch\_size, x, y, z].

If enqueue\_many is True, tensors\_list[i] is assumed to represent a batch of examples, where the first dimension is indexed by example, and all members of tensors\_list[i] should have the same size in the first dimension. If an input tensor has shape [\*, x, y, z], the output will have shape [batch\_size, x, y, z].

The capacity argument controls the how long the prefetching is allowed to grow the queues.

The returned operation is a dequeue operation and will throw [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError) if the input queue is exhausted. If this operation is feeding another input queue, its queue runner will catch this exception, however, if this operation is used in your main thread you are responsible for catching this yourself.

If allow\_smaller\_final\_batch is True, a smaller batch value than batch\_size is returned when the queue is closed and there are not enough elements to fill the batch, otherwise the pending elements are discarded. In addition, all output tensors' static shapes, as accessed via the shapeproperty will have a first Dimension value of None, and operations that depend on fixed batch\_size would fail.

#### Args:

* **tensors\_list**: A list of tuples or dictionaries of tensors to enqueue.
* **batch\_size**: An integer. The new batch size pulled from the queue.
* **capacity**: An integer. The maximum number of elements in the queue.
* **min\_after\_dequeue**: Minimum number elements in the queue after a dequeue, used to ensure a level of mixing of elements.
* **seed**: Seed for the random shuffling within the queue.
* **enqueue\_many**: Whether each tensor in tensor\_list\_list is a single example.
* **shapes**: (Optional) The shapes for each example. Defaults to the inferred shapes for tensors\_list[i].
* **allow\_smaller\_final\_batch**: (Optional) Boolean. If True, allow the final batch to be smaller if there are insufficient items left in the queue.
* **shared\_name**: (optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: (Optional) A name for the operations.

#### Returns:

A list or dictionary of tensors with the same number and types as tensors\_list[i].

#### Raises:

* **ValueError**: If the shapes are not specified, and cannot be inferred from the elements of tensors\_list.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.SingularMonitoredSession

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession#top_of_page)
* [Class SingularMonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession#class_singularmonitoredsession)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession#__init__)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession#child_classes)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SingularMonitoredSession#properties)

## Class SingularMonitoredSession

Session-like object that handles initialization, restoring, and hooks.

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

Please note that this utility is not recommended for distributed settings. For distributed settings, please use [tf.compat.v1.train.MonitoredSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession). The differences between MonitoredSessionand SingularMonitoredSession are:

* MonitoredSession handles AbortedError and UnavailableError for distributed settings, but SingularMonitoredSession does not.
* MonitoredSession can be created in chief or worker modes. SingularMonitoredSessionis always created as chief.
* You can access the raw [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) object used by SingularMonitoredSession, whereas in MonitoredSession the raw session is private. This can be used:
  + To run without hooks.
  + To save and restore.
* All other functionality is identical.

#### Example usage:

saver\_hook = CheckpointSaverHook(...)  
summary\_hook = SummarySaverHook(...)  
with SingularMonitoredSession(hooks=[saver\_hook, summary\_hook]) as sess:  
  while not sess.should\_stop():  
    sess.run(train\_op)

Initialization: At creation time the hooked session does following things in given order:

* calls hook.begin() for each given hook
* finalizes the graph via scaffold.finalize()
* create session
* initializes the model via initialization ops provided by Scaffold
* restores variables if a checkpoint exists
* launches queue runners

Run: When run() is called, the hooked session does following things:

* calls hook.before\_run()
* calls TensorFlow session.run() with merged fetches and feed\_dict
* calls hook.after\_run()
* returns result of session.run() asked by user

Exit: At the close(), the hooked session does following things in order:

* calls hook.end()
* closes the queue runners and the session
* suppresses OutOfRange error which indicates that all inputs have been processed if the SingularMonitoredSession is used as a context.

## \_\_init\_\_

\_\_init\_\_(  
    hooks=None,  
    scaffold=None,  
    master='',  
    config=None,  
    checkpoint\_dir=None,  
    stop\_grace\_period\_secs=120,  
    checkpoint\_filename\_with\_path=None  
)

Creates a SingularMonitoredSession.

#### Args:

* **hooks**: An iterable of `SessionRunHook' objects.
* **scaffold**: A Scaffold used for gathering or building supportive ops. If not specified a default one is created. It's used to finalize the graph.
* **master**: String representation of the TensorFlow master to use.
* **config**: ConfigProto proto used to configure the session.
* **checkpoint\_dir**: A string. Optional path to a directory where to restore variables.
* **stop\_grace\_period\_secs**: Number of seconds given to threads to stop after close() has been called.
* **checkpoint\_filename\_with\_path**: A string. Optional path to a checkpoint file from which to restore variables.

## Child Classes

[class StepContext](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredSession/StepContext)

## Properties

### graph

The graph that was launched in this session.

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

### \_\_exit\_\_

\_\_exit\_\_(  
    exception\_type,  
    exception\_value,  
    traceback  
)

### close

close()

### raw\_session

raw\_session()

Returns underlying TensorFlow.Session object.

### run

run(  
    fetches,  
    feed\_dict=None,  
    options=None,  
    run\_metadata=None  
)

Run ops in the monitored session.

This method is completely compatible with the tf.Session.run() method.

#### Args:

* **fetches**: Same as tf.Session.run().
* **feed\_dict**: Same as tf.Session.run().
* **options**: Same as tf.Session.run().
* **run\_metadata**: Same as tf.Session.run().

#### Returns:

Same as tf.Session.run().

### run\_step\_fn

run\_step\_fn(step\_fn)

Run ops using a step function.

#### Args:

* **step\_fn**: A function or a method with a single argument of type StepContext. The function may use methods of the argument to perform computations with access to a raw session. The returned value of the step\_fn will be returned from run\_step\_fn, unless a stop is requested. In that case, the next should\_stop call will return True. Example usage: ```python with tf.Graph().as\_default(): c = tf.compat.v1.placeholder(dtypes.float32) v = tf.add(c, 4.0) w = tf.add(c, 0.5) def step\_fn(step\_context): a = step\_context.session.run(fetches=v, feed\_dict={c: 0.5}) if a <= 4.5: step\_context.request\_stop() return step\_context.run\_with\_hooks(fetches=w, feed\_dict={c: 0.1}) with tf.MonitoredSession() as session: while not session.should\_stop(): a = session.run\_step\_fn(step\_fn)

   ```  Hooks interact with the `run\_with\_hooks()` call inside the  
   `step\_fn` as they do with a `MonitoredSession.run` call.

#### Returns:

Returns the returned value of step\_fn.

#### Raises:

* **StopIteration**: if step\_fn has called request\_stop(). It may be caught by with tf.MonitoredSession() to close the session.
* **ValueError**: if step\_fn doesn't have a single argument called step\_context. It may also optionally have self for cases when it belongs to an object.

### should\_stop

should\_stop()

# tf.compat.v1.train.slice\_input\_producer

Produces a slice of each Tensor in tensor\_list. (deprecated)

tf.compat.v1.train.slice\_input\_producer(  
    tensor\_list,  
    num\_epochs=None,  
    shuffle=True,  
    seed=None,  
    capacity=32,  
    shared\_name=None,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.from\_tensor\_slices(tuple(tensor\_list)).shuffle(tf.shape(input\_tensor, out\_type=tf.int64)[0]).repeat(num\_epochs)**. If **shuffle=False**, omit the **.shuffle(...)**.

Implemented using a Queue -- a QueueRunner for the Queue is added to the current Graph's QUEUE\_RUNNER collection.

#### Args:

* **tensor\_list**: A list of Tensor objects. Every Tensor in tensor\_list must have the same size in the first dimension.
* **num\_epochs**: An integer (optional). If specified, slice\_input\_producer produces each slice num\_epochs times before generating an OutOfRange error. If not specified, slice\_input\_producer can cycle through the slices an unlimited number of times.
* **shuffle**: Boolean. If true, the integers are randomly shuffled within each epoch.
* **seed**: An integer (optional). Seed used if shuffle == True.
* **capacity**: An integer. Sets the queue capacity.
* **shared\_name**: (optional). If set, this queue will be shared under the given name across multiple sessions.
* **name**: A name for the operations (optional).

#### Returns:

A list of tensors, one for each element of tensor\_list. If the tensor in tensor\_list has shape [N, a, b, .., z], then the corresponding output tensor will have shape [a, b, ..., z].

#### Raises:

* **ValueError**: if slice\_input\_producer produces nothing from tensor\_list.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.start\_queue\_runners

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/start_queue_runners#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/start_queue_runners#aliases)

Starts all queue runners collected in the graph. (deprecated)

### Aliases:

* tf.compat.v1.train.queue\_runner.start\_queue\_runners
* tf.compat.v1.train.start\_queue\_runners

tf.compat.v1.train.start\_queue\_runners(  
    sess=None,  
    coord=None,  
    daemon=True,  
    start=True,  
    collection=tf.GraphKeys.QUEUE\_RUNNERS  
)

Defined in [python/training/queue\_runner\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/queue_runner_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: To construct input pipelines, use the [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) module.

This is a companion method to add\_queue\_runner(). It just starts threads for all queue runners collected in the graph. It returns the list of all threads.

#### Args:

* **sess**: Session used to run the queue ops. Defaults to the default session.
* **coord**: Optional Coordinator for coordinating the started threads.
* **daemon**: Whether the threads should be marked as daemons, meaning they don't block program exit.
* **start**: Set to False to only create the threads, not start them.
* **collection**: A GraphKey specifying the graph collection to get the queue runners from. Defaults to GraphKeys.QUEUE\_RUNNERS.

#### Raises:

* **ValueError**: if sess is None and there isn't any default session.
* **TypeError**: if sess is not a [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) object.

#### Returns:

A list of threads.

#### Raises:

* **RuntimeError**: If called with eager execution enabled.
* **ValueError**: If called without a default [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) registered.

#### Eager Compatibility

Not compatible with eager execution. To ingest data under eager execution, use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API instead.

# tf.compat.v1.train.string\_input\_producer

Output strings (e.g. filenames) to a queue for an input pipeline. (deprecated)

tf.compat.v1.train.string\_input\_producer(  
    string\_tensor,  
    num\_epochs=None,  
    shuffle=True,  
    seed=None,  
    capacity=32,  
    shared\_name=None,  
    name=None,  
    cancel\_op=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Queue-based input pipelines have been replaced by [**tf.data**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data). Use **tf.data.Dataset.from\_tensor\_slices(string\_tensor).shuffle(tf.shape(input\_tensor, out\_type=tf.int64)[0]).repeat(num\_epochs)**. If **shuffle=False**, omit the **.shuffle(...)**.**Note:** if **num\_epochs** is not **None**, this function creates local counter **epochs**. Use **local\_variables\_initializer()** to initialize local variables.

#### Args:

* **string\_tensor**: A 1-D string tensor with the strings to produce.
* **num\_epochs**: An integer (optional). If specified, string\_input\_producer produces each string from string\_tensor num\_epochs times before generating an OutOfRange error. If not specified, string\_input\_producer can cycle through the strings in string\_tensor an unlimited number of times.
* **shuffle**: Boolean. If true, the strings are randomly shuffled within each epoch.
* **seed**: An integer (optional). Seed used if shuffle == True.
* **capacity**: An integer. Sets the queue capacity.
* **shared\_name**: (optional). If set, this queue will be shared under the given name across multiple sessions. All sessions open to the device which has this queue will be able to access it via the shared\_name. Using this in a distributed setting means each name will only be seen by one of the sessions which has access to this operation.
* **name**: A name for the operations (optional).
* **cancel\_op**: Cancel op for the queue (optional).

#### Returns:

A queue with the output strings. A QueueRunner for the Queue is added to the current Graph's QUEUE\_RUNNER collection.

#### Raises:

* **ValueError**: If the string\_tensor is a null Python list. At runtime, will fail with an assertion if string\_tensor becomes a null tensor.

#### Eager Compatibility

Input pipelines based on Queues are not supported when eager execution is enabled. Please use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API to ingest data under eager execution.

# tf.compat.v1.train.summary\_iterator

An iterator for reading Event protocol buffers from an event file.

tf.compat.v1.train.summary\_iterator(path)

Defined in [python/summary/summary\_iterator.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/summary/summary_iterator.py).

You can use this function to read events written to an event file. It returns a Python iterator that yields Event protocol buffers.

Example: Print the contents of an events file.

for e in tf.compat.v1.train.summary\_iterator(path to events file):  
    print(e)

Example: Print selected summary values.

# This example supposes that the events file contains summaries with a  
# summary value tag 'loss'.  These could have been added by calling  
# `add\_summary()`, passing the output of a scalar summary op created with  
# with: `tf.compat.v1.summary.scalar('loss', loss\_tensor)`.  
for e in tf.compat.v1.train.summary\_iterator(path to events file):  
    for v in e.summary.value:  
        if v.tag == 'loss':  
            print(v.simple\_value)

See the protocol buffer definitions of [Event](https://www.tensorflow.org/code/tensorflow/core/util/event.proto) and [Summary](https://www.tensorflow.org/code/tensorflow/core/framework/summary.proto) for more information about their attributes.

#### Args:

* **path**: The path to an event file created by a SummaryWriter.

#### Yields:

Event protocol buffers.

# tf.compat.v1.train.Supervisor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor#top_of_page)
* [Class Supervisor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor#class_supervisor)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor#properties)
  + [coord](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Supervisor#coord)

## Class Supervisor

A training helper that checkpoints models and computes summaries.

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

This class is deprecated. Please use [tf.compat.v1.train.MonitoredTrainingSession](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/MonitoredTrainingSession) instead.

The Supervisor is a small wrapper around a Coordinator, a Saver, and a SessionManager that takes care of common needs of TensorFlow training programs.

#### Use for a single program

with tf.Graph().as\_default():  
  ...add operations to the graph...  
  # Create a Supervisor that will checkpoint the model in '/tmp/mydir'.  
  sv = Supervisor(logdir='/tmp/mydir')  
  # Get a TensorFlow session managed by the supervisor.  
  with sv.managed\_session(FLAGS.master) as sess:  
    # Use the session to train the graph.  
    while not sv.should\_stop():  
      sess.run(<my\_train\_op>)

Within the with sv.managed\_session() block all variables in the graph have been initialized. In addition, a few services have been started to checkpoint the model and add summaries to the event log.

If the program crashes and is restarted, the managed session automatically reinitialize variables from the most recent checkpoint.

The supervisor is notified of any exception raised by one of the services. After an exception is raised, should\_stop() returns True. In that case the training loop should also stop. This is why the training loop has to check for sv.should\_stop().

Exceptions that indicate that the training inputs have been exhausted, [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError), also cause sv.should\_stop() to return True but are not re-raised from the with block: they indicate a normal termination.

#### Use for multiple replicas

To train with replicas you deploy the same program in a Cluster. One of the tasks must be identified as the chief: the task that handles initialization, checkpoints, summaries, and recovery. The other tasks depend on the chief for these services.

The only change you have to do to the single program code is to indicate if the program is running as the chief.

# Choose a task as the chief. This could be based on server\_def.task\_index,  
# or job\_def.name, or job\_def.tasks. It's entirely up to the end user.  
# But there can be only one \*chief\*.  
is\_chief = (server\_def.task\_index == 0)  
server = tf.distribute.Server(server\_def)  
  
with tf.Graph().as\_default():  
  ...add operations to the graph...  
  # Create a Supervisor that uses log directory on a shared file system.  
  # Indicate if you are the 'chief'  
  sv = Supervisor(logdir='/shared\_directory/...', is\_chief=is\_chief)  
  # Get a Session in a TensorFlow server on the cluster.  
  with sv.managed\_session(server.target) as sess:  
    # Use the session to train the graph.  
    while not sv.should\_stop():  
      sess.run(<my\_train\_op>)

In the chief task, the Supervisor works exactly as in the first example above. In the other tasks sv.managed\_session() waits for the Model to have been initialized before returning a session to the training code. The non-chief tasks depend on the chief task for initializing the model.

If one of the tasks crashes and restarts, managed\_session() checks if the Model is initialized. If yes, it just creates a session and returns it to the training code that proceeds normally. If the model needs to be initialized, the chief task takes care of reinitializing it; the other tasks just wait for the model to have been initialized.

NOTE: This modified program still works fine as a single program. The single program marks itself as the chief.

#### What master string to use

Whether you are running on your machine or in the cluster you can use the following values for the --master flag:

* Specifying '' requests an in-process session that does not use RPC.
* Specifying 'local' requests a session that uses the RPC-based "Master interface" to run TensorFlow programs. See tf.train.Server.create\_local\_server for details.
* Specifying 'grpc://hostname:port' requests a session that uses the RPC interface to a specific host, and also allows the in-process master to access remote tensorflow workers. Often, it is appropriate to pass server.target (for some [tf.distribute.Server](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Server) named `server).

#### Advanced use

##### Launching additional services

managed\_session() launches the Checkpoint and Summary services (threads). If you need more services to run you can simply launch them in the block controlled by managed\_session().

Example: Start a thread to print losses. We want this thread to run every 60 seconds, so we launch it with sv.loop().

...  
sv = Supervisor(logdir='/tmp/mydir')  
with sv.managed\_session(FLAGS.master) as sess:  
  sv.loop(60, print\_loss, (sess, ))  
  while not sv.should\_stop():  
    sess.run(my\_train\_op)

##### Launching fewer services

managed\_session() launches the "summary" and "checkpoint" threads which use either the optionally summary\_op and saver passed to the constructor, or default ones created automatically by the supervisor. If you want to run your own summary and checkpointing logic, disable these services by passing None to the summary\_op and saver parameters.

Example: Create summaries manually every 100 steps in the chief.

# Create a Supervisor with no automatic summaries.  
sv = Supervisor(logdir='/tmp/mydir', is\_chief=is\_chief, summary\_op=None)  
# As summary\_op was None, managed\_session() does not start the  
# summary thread.  
with sv.managed\_session(FLAGS.master) as sess:  
  for step in xrange(1000000):  
    if sv.should\_stop():  
      break  
    if is\_chief and step % 100 == 0:  
      # Create the summary every 100 chief steps.  
      sv.summary\_computed(sess, sess.run(my\_summary\_op))  
    else:  
      # Train normally  
      sess.run(my\_train\_op)

##### Custom model initialization

managed\_session() only supports initializing the model by running an init\_op or restoring from the latest checkpoint. If you have special initialization needs, see how to specify a local\_init\_opwhen creating the supervisor. You can also use the SessionManager directly to create a session and check if it could be initialized automatically.

## \_\_init\_\_

\_\_init\_\_(  
    graph=None,  
    ready\_op=USE\_DEFAULT,  
    ready\_for\_local\_init\_op=USE\_DEFAULT,  
    is\_chief=True,  
    init\_op=USE\_DEFAULT,  
    init\_feed\_dict=None,  
    local\_init\_op=USE\_DEFAULT,  
    logdir=None,  
    summary\_op=USE\_DEFAULT,  
    saver=USE\_DEFAULT,  
    global\_step=USE\_DEFAULT,  
    save\_summaries\_secs=120,  
    save\_model\_secs=600,  
    recovery\_wait\_secs=30,  
    stop\_grace\_secs=120,  
    checkpoint\_basename='model.ckpt',  
    session\_manager=None,  
    summary\_writer=USE\_DEFAULT,  
    init\_fn=None,  
    local\_init\_run\_options=None  
)

Create a Supervisor. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Please switch to tf.train.MonitoredTrainingSession

#### Args:

* **graph**: A Graph. The graph that the model will use. Defaults to the default Graph. The supervisor may add operations to the graph before creating a session, but the graph should not be modified by the caller after passing it to the supervisor.
* **ready\_op**: 1-D string Tensor. This tensor is evaluated by supervisors inprepare\_or\_wait\_for\_session() to check if the model is ready to use. The model is considered ready if it returns an empty array. Defaults to the tensor returned from tf.compat.v1.report\_uninitialized\_variables() If None, the model is not checked for readiness.
* **ready\_for\_local\_init\_op**: 1-D string Tensor. This tensor is evaluated by supervisors in prepare\_or\_wait\_for\_session() to check if the model is ready to run the local\_init\_op. The model is considered ready if it returns an empty array. Defaults to None. If None, the model is not checked for readiness before running local\_init\_op.
* **is\_chief**: If True, create a chief supervisor in charge of initializing and restoring the model. If False, create a supervisor that relies on a chief supervisor for inits and restore.
* **init\_op**: Operation. Used by chief supervisors to initialize the model when it can not be recovered. Defaults to an Operation that initializes all global variables. If None, no initialization is done automatically unless you pass a value for init\_fn, see below.
* **init\_feed\_dict**: A dictionary that maps Tensor objects to feed values. This feed dictionary will be used when init\_op is evaluated.
* **local\_init\_op**: Operation. Used by all supervisors to run initializations that should run for every new supervisor instance. By default these are table initializers and initializers for local variables. If None, no further per supervisor-instance initialization is done automatically.
* **logdir**: A string. Optional path to a directory where to checkpoint the model and log events for the visualizer. Used by chief supervisors. The directory will be created if it does not exist.
* **summary\_op**: An Operation that returns a Summary for the event logs. Used by chief supervisors if a logdir was specified. Defaults to the operation returned from summary.merge\_all(). If None, summaries are not computed automatically.
* **saver**: A Saver object. Used by chief supervisors if a logdir was specified. Defaults to the saved returned by Saver(). If None, the model is not saved automatically.
* **global\_step**: An integer Tensor of size 1 that counts steps. The value from 'global\_step' is used in summaries and checkpoint filenames. Default to the op named 'global\_step' in the graph if it exists, is of rank 1, size 1, and of type tf.int32 or tf.int64. If None the global step is not recorded in summaries and checkpoint files. Used by chief supervisors if a logdir was specified.
* **save\_summaries\_secs**: Number of seconds between the computation of summaries for the event log. Defaults to 120 seconds. Pass 0 to disable summaries.
* **save\_model\_secs**: Number of seconds between the creation of model checkpoints. Defaults to 600 seconds. Pass 0 to disable checkpoints.
* **recovery\_wait\_secs**: Number of seconds between checks that the model is ready. Used by supervisors when waiting for a chief supervisor to initialize or restore the model. Defaults to 30 seconds.
* **stop\_grace\_secs**: Grace period, in seconds, given to running threads to stop when stop() is called. Defaults to 120 seconds.
* **checkpoint\_basename**: The basename for checkpoint saving.
* **session\_manager**: SessionManager, which manages Session creation and recovery. If it is None, a default SessionManager will be created with the set of arguments passed in for backwards compatibility.
* **summary\_writer**: SummaryWriter to use or USE\_DEFAULT. Can be None to indicate that no summaries should be written.
* **init\_fn**: Optional callable used to initialize the model. Called after the optional init\_op is called. The callable must accept one argument, the session being initialized.
* **local\_init\_run\_options**: RunOptions to be passed as the SessionManager local\_init\_run\_options parameter.

#### Returns:

A Supervisor.

#### Raises:

* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Supervisors are not supported when eager execution is enabled.

## Properties

### coord

Return the Coordinator used by the Supervisor.

The Coordinator can be useful if you want to run multiple threads during your training.

#### Returns:

A Coordinator object.

### global\_step

Return the global\_step Tensor used by the supervisor.

#### Returns:

An integer Tensor for the global\_step.

### init\_feed\_dict

Return the feed dictionary used when evaluating the init\_op.

#### Returns:

A feed dictionary or None.

### init\_op

Return the Init Op used by the supervisor.

#### Returns:

An Op or None.

### is\_chief

Return True if this is a chief supervisor.

#### Returns:

A bool.

### ready\_for\_local\_init\_op

### ready\_op

Return the Ready Op used by the supervisor.

#### Returns:

An Op or None.

### save\_model\_secs

Return the delay between checkpoints.

#### Returns:

A timestamp.

### save\_path

Return the save path used by the supervisor.

#### Returns:

A string.

### save\_summaries\_secs

Return the delay between summary computations.

#### Returns:

A timestamp.

### saver

Return the Saver used by the supervisor.

#### Returns:

A Saver object.

### session\_manager

Return the SessionManager used by the Supervisor.

#### Returns:

A SessionManager object.

### summary\_op

Return the Summary Tensor used by the chief supervisor.

#### Returns:

A string Tensor for the summary or None.

### summary\_writer

Return the SummaryWriter used by the chief supervisor.

#### Returns:

A SummaryWriter.

## Methods

### Loop

Loop(  
    timer\_interval\_secs,  
    target,  
    args=None,  
    kwargs=None  
)

Start a LooperThread that calls a function periodically.

If timer\_interval\_secs is None the thread calls target(\*args, \*\*kwargs) repeatedly. Otherwise it calls it every timer\_interval\_secs seconds. The thread terminates when a stop is requested.

The started thread is added to the list of threads managed by the supervisor so it does not need to be passed to the stop() method.

#### Args:

* **timer\_interval\_secs**: Number. Time boundaries at which to call target.
* **target**: A callable object.
* **args**: Optional arguments to pass to target when calling it.
* **kwargs**: Optional keyword arguments to pass to target when calling it.

#### Returns:

The started thread.

### PrepareSession

PrepareSession(  
    master='',  
    config=None,  
    wait\_for\_checkpoint=False,  
    max\_wait\_secs=7200,  
    start\_standard\_services=True  
)

Make sure the model is ready to be used.

Create a session on 'master', recovering or initializing the model as needed, or wait for a session to be ready. If running as the chief and start\_standard\_service is set to True, also call the session manager to start the standard services.

#### Args:

* **master**: name of the TensorFlow master to use. See the [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) constructor for how this is interpreted.
* **config**: Optional ConfigProto proto used to configure the session, which is passed as-is to create the session.
* **wait\_for\_checkpoint**: Whether we should wait for the availability of a checkpoint before creating Session. Defaults to False.
* **max\_wait\_secs**: Maximum time to wait for the session to become available.
* **start\_standard\_services**: Whether to start the standard services and the queue runners.

#### Returns:

A Session object that can be used to drive the model.

### RequestStop

RequestStop(ex=None)

Request that the coordinator stop the threads.

See Coordinator.request\_stop().

#### Args:

* **ex**: Optional Exception, or Python exc\_info tuple as returned by sys.exc\_info(). If this is the first call to request\_stop() the corresponding exception is recorded and re-raised from join().

### ShouldStop

ShouldStop()

Check if the coordinator was told to stop.

See Coordinator.should\_stop().

#### Returns:

True if the coordinator was told to stop, False otherwise.

### StartQueueRunners

StartQueueRunners(  
    sess,  
    queue\_runners=None  
)

Start threads for QueueRunners.

Note that the queue runners collected in the graph key QUEUE\_RUNNERS are already started automatically when you create a session with the supervisor, so unless you have non-collected queue runners to start you do not need to call this explicitly.

#### Args:

* **sess**: A Session.
* **queue\_runners**: A list of QueueRunners. If not specified, we'll use the list of queue runners gathered in the graph under the key GraphKeys.QUEUE\_RUNNERS.

#### Returns:

The list of threads started for the QueueRunners.

#### Raises:

* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Queues are not compatible with eager execution. To ingest data when eager execution is enabled, use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API.

### StartStandardServices

StartStandardServices(sess)

Start the standard services for 'sess'.

This starts services in the background. The services started depend on the parameters to the constructor and may include:

* A Summary thread computing summaries every save\_summaries\_secs.
* A Checkpoint thread saving the model every save\_model\_secs.
* A StepCounter thread measure step time.

#### Args:

* **sess**: A Session.

#### Returns:

A list of threads that are running the standard services. You can use the Supervisor's Coordinator to join these threads with: sv.coord.Join()

#### Raises:

* **RuntimeError**: If called with a non-chief Supervisor.
* **ValueError**: If not logdir was passed to the constructor as the services need a log directory.

### Stop

Stop(  
    threads=None,  
    close\_summary\_writer=True,  
    ignore\_live\_threads=False  
)

Stop the services and the coordinator.

This does not close the session.

#### Args:

* **threads**: Optional list of threads to join with the coordinator. If None, defaults to the threads running the standard services, the threads started for QueueRunners, and the threads started by the loop() method. To wait on additional threads, pass the list in this parameter.
* **close\_summary\_writer**: Whether to close the summary\_writer. Defaults to True if the summary writer was created by the supervisor, False otherwise.
* **ignore\_live\_threads**: If True ignores threads that remain running after a grace period when joining threads via the coordinator, instead of raising a RuntimeError.

### StopOnException

StopOnException()

Context handler to stop the supervisor when an exception is raised.

See Coordinator.stop\_on\_exception().

#### Returns:

A context handler.

### SummaryComputed

SummaryComputed(  
    sess,  
    summary,  
    global\_step=None  
)

Indicate that a summary was computed.

#### Args:

* **sess**: A Session object.
* **summary**: A Summary proto, or a string holding a serialized summary proto.
* **global\_step**: Int. global step this summary is associated with. If None, it will try to fetch the current step.

#### Raises:

* **TypeError**: if 'summary' is not a Summary proto or a string.
* **RuntimeError**: if the Supervisor was created without a logdir.

### WaitForStop

WaitForStop()

Block waiting for the coordinator to stop.

### loop

loop(  
    timer\_interval\_secs,  
    target,  
    args=None,  
    kwargs=None  
)

Start a LooperThread that calls a function periodically.

If timer\_interval\_secs is None the thread calls target(\*args, \*\*kwargs) repeatedly. Otherwise it calls it every timer\_interval\_secs seconds. The thread terminates when a stop is requested.

The started thread is added to the list of threads managed by the supervisor so it does not need to be passed to the stop() method.

#### Args:

* **timer\_interval\_secs**: Number. Time boundaries at which to call target.
* **target**: A callable object.
* **args**: Optional arguments to pass to target when calling it.
* **kwargs**: Optional keyword arguments to pass to target when calling it.

#### Returns:

The started thread.

### managed\_session

managed\_session(  
    \*args,  
    \*\*kwds  
)

Returns a context manager for a managed session.

This context manager creates and automatically recovers a session. It optionally starts the standard services that handle checkpoints and summaries. It monitors exceptions raised from the with block or from the services and stops the supervisor as needed.

The context manager is typically used as follows:

def train():  
  sv = tf.compat.v1.train.Supervisor(...)  
  with sv.managed\_session(<master>) as sess:  
    for step in xrange(..):  
      if sv.should\_stop():  
        break  
      sess.run(<my training op>)  
      ...do other things needed at each training step...

An exception raised from the with block or one of the service threads is raised again when the block exits. This is done after stopping all threads and closing the session. For example, an AbortedErrorexception, raised in case of preemption of one of the workers in a distributed model, is raised again when the block exits.

If you want to retry the training loop in case of preemption you can do it as follows:

def main(...):  
  while True  
    try:  
      train()  
    except tf.errors.Aborted:  
      pass

As a special case, exceptions used for control flow, such as OutOfRangeError which reports that input queues are exhausted, are not raised again from the with block: they indicate a clean termination of the training loop and are considered normal termination.

#### Args:

* **master**: name of the TensorFlow master to use. See the [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) constructor for how this is interpreted.
* **config**: Optional ConfigProto proto used to configure the session. Passed as-is to create the session.
* **start\_standard\_services**: Whether to start the standard services, such as checkpoint, summary and step counter.
* **close\_summary\_writer**: Whether to close the summary writer when closing the session. Defaults to True.

#### Returns:

A context manager that yields a Session restored from the latest checkpoint or initialized from scratch if not checkpoint exists. The session is closed when the with block exits.

### prepare\_or\_wait\_for\_session

prepare\_or\_wait\_for\_session(  
    master='',  
    config=None,  
    wait\_for\_checkpoint=False,  
    max\_wait\_secs=7200,  
    start\_standard\_services=True  
)

Make sure the model is ready to be used.

Create a session on 'master', recovering or initializing the model as needed, or wait for a session to be ready. If running as the chief and start\_standard\_service is set to True, also call the session manager to start the standard services.

#### Args:

* **master**: name of the TensorFlow master to use. See the [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) constructor for how this is interpreted.
* **config**: Optional ConfigProto proto used to configure the session, which is passed as-is to create the session.
* **wait\_for\_checkpoint**: Whether we should wait for the availability of a checkpoint before creating Session. Defaults to False.
* **max\_wait\_secs**: Maximum time to wait for the session to become available.
* **start\_standard\_services**: Whether to start the standard services and the queue runners.

#### Returns:

A Session object that can be used to drive the model.

### request\_stop

request\_stop(ex=None)

Request that the coordinator stop the threads.

See Coordinator.request\_stop().

#### Args:

* **ex**: Optional Exception, or Python exc\_info tuple as returned by sys.exc\_info(). If this is the first call to request\_stop() the corresponding exception is recorded and re-raised from join().

### should\_stop

should\_stop()

Check if the coordinator was told to stop.

See Coordinator.should\_stop().

#### Returns:

True if the coordinator was told to stop, False otherwise.

### start\_queue\_runners

start\_queue\_runners(  
    sess,  
    queue\_runners=None  
)

Start threads for QueueRunners.

Note that the queue runners collected in the graph key QUEUE\_RUNNERS are already started automatically when you create a session with the supervisor, so unless you have non-collected queue runners to start you do not need to call this explicitly.

#### Args:

* **sess**: A Session.
* **queue\_runners**: A list of QueueRunners. If not specified, we'll use the list of queue runners gathered in the graph under the key GraphKeys.QUEUE\_RUNNERS.

#### Returns:

The list of threads started for the QueueRunners.

#### Raises:

* **RuntimeError**: If called with eager execution enabled.

#### Eager Compatibility

Queues are not compatible with eager execution. To ingest data when eager execution is enabled, use the [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data) API.

### start\_standard\_services

start\_standard\_services(sess)

Start the standard services for 'sess'.

This starts services in the background. The services started depend on the parameters to the constructor and may include:

* A Summary thread computing summaries every save\_summaries\_secs.
* A Checkpoint thread saving the model every save\_model\_secs.
* A StepCounter thread measure step time.

#### Args:

* **sess**: A Session.

#### Returns:

A list of threads that are running the standard services. You can use the Supervisor's Coordinator to join these threads with: sv.coord.Join()

#### Raises:

* **RuntimeError**: If called with a non-chief Supervisor.
* **ValueError**: If not logdir was passed to the constructor as the services need a log directory.

### stop

stop(  
    threads=None,  
    close\_summary\_writer=True,  
    ignore\_live\_threads=False  
)

Stop the services and the coordinator.

This does not close the session.

#### Args:

* **threads**: Optional list of threads to join with the coordinator. If None, defaults to the threads running the standard services, the threads started for QueueRunners, and the threads started by the loop() method. To wait on additional threads, pass the list in this parameter.
* **close\_summary\_writer**: Whether to close the summary\_writer. Defaults to True if the summary writer was created by the supervisor, False otherwise.
* **ignore\_live\_threads**: If True ignores threads that remain running after a grace period when joining threads via the coordinator, instead of raising a RuntimeError.

### stop\_on\_exception

stop\_on\_exception()

Context handler to stop the supervisor when an exception is raised.

See Coordinator.stop\_on\_exception().

#### Returns:

A context handler.

### summary\_computed

summary\_computed(  
    sess,  
    summary,  
    global\_step=None  
)

Indicate that a summary was computed.

#### Args:

* **sess**: A Session object.
* **summary**: A Summary proto, or a string holding a serialized summary proto.
* **global\_step**: Int. global step this summary is associated with. If None, it will try to fetch the current step.

#### Raises:

* **TypeError**: if 'summary' is not a Summary proto or a string.
* **RuntimeError**: if the Supervisor was created without a logdir.

### wait\_for\_stop

wait\_for\_stop()

Block waiting for the coordinator to stop.

## Class Members

* USE\_DEFAULT = 0

# tf.compat.v1.train.SyncReplicasOptimizer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer#top_of_page)
* [Class SyncReplicasOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer#class_syncreplicasoptimizer)
  + [Usage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer#usage)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SyncReplicasOptimizer#methods)

## Class SyncReplicasOptimizer

Class to synchronize, aggregate gradients and pass them to the optimizer.

Inherits From: [Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)

Defined in [python/training/sync\_replicas\_optimizer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/sync_replicas_optimizer.py).

This class is deprecated. For synchrononous training, please use [Distribution Strategies](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute).

In a typical asynchronous training environment, it's common to have some stale gradients. For example, with a N-replica asynchronous training, gradients will be applied to the variables N times independently. Depending on each replica's training speed, some gradients might be calculated from copies of the variable from several steps back (N-1 steps on average). This optimizer avoids stale gradients by collecting gradients from all replicas, averaging them, then applying them to the variables in one shot, after which replicas can fetch the new variables and continue.

The following accumulators/queue are created:

* N gradient accumulators, one per variable to train. Gradients are pushed to them and the chief worker will wait until enough gradients are collected and then average them before applying to variables. The accumulator will drop all stale gradients (more details in the accumulator op).
* 1 token queue where the optimizer pushes the new global\_step value after all variables are updated.

The following local variable is created: \* sync\_rep\_local\_step, one per replica. Compared against the global\_step in each accumulator to check for staleness of the gradients.

The optimizer adds nodes to the graph to collect gradients and pause the trainers until variables are updated. For the Parameter Server job:

1. An accumulator is created for each variable, and each replica pushes the gradients into the accumulators instead of directly applying them to the variables.
2. Each accumulator averages once enough gradients (replicas\_to\_aggregate) have been accumulated.
3. Apply the averaged gradients to the variables.
4. Only after all variables have been updated, increment the global step.
5. Only after step 4, pushes global\_step in the token\_queue, once for each worker replica. The workers can now fetch the global step, use it to update its local\_step variable and start the next batch. Please note that some workers can consume multiple minibatches, while some may not consume even one. This is because each worker fetches minibatches as long as a token exists. If one worker is stuck for some reason and does not consume a token, another worker can use it.

#### For the replicas:

1. Start a step: fetch variables and compute gradients.
2. Once the gradients have been computed, push them into gradient accumulators. Each accumulator will check the staleness and drop the stale.
3. After pushing all the gradients, dequeue an updated value of global\_step from the token queue and record that step to its local\_step variable. Note that this is effectively a barrier.
4. Start the next batch.

### Usage

# Create any optimizer to update the variables, say a simple SGD:  
opt = GradientDescentOptimizer(learning\_rate=0.1)  
  
# Wrap the optimizer with sync\_replicas\_optimizer with 50 replicas: at each  
# step the optimizer collects 50 gradients before applying to variables.  
# Note that if you want to have 2 backup replicas, you can change  
# total\_num\_replicas=52 and make sure this number matches how many physical  
# replicas you started in your job.  
opt = tf.compat.v1.train.SyncReplicasOptimizer(opt, replicas\_to\_aggregate=50,  
                               total\_num\_replicas=50)  
  
# Some models have startup\_delays to help stabilize the model but when using  
# sync\_replicas training, set it to 0.  
  
# Now you can call `minimize()` or `compute\_gradients()` and  
# `apply\_gradients()` normally  
training\_op = opt.minimize(total\_loss, global\_step=self.global\_step)  
  
  
# You can create the hook which handles initialization and queues.  
sync\_replicas\_hook = opt.make\_session\_run\_hook(is\_chief)

In the training program, every worker will run the train\_op as if not synchronized.

with training.MonitoredTrainingSession(  
    master=workers[worker\_id].target, is\_chief=is\_chief,  
    hooks=[sync\_replicas\_hook]) as mon\_sess:  
  while not mon\_sess.should\_stop():  
    mon\_sess.run(training\_op)

To use SyncReplicasOptimizer with an Estimator, you need to send sync\_replicas\_hook while calling the fit.

my\_estimator = DNNClassifier(..., optimizer=opt)  
my\_estimator.fit(..., hooks=[sync\_replicas\_hook])

## \_\_init\_\_

\_\_init\_\_(  
    opt,  
    replicas\_to\_aggregate,  
    total\_num\_replicas=None,  
    variable\_averages=None,  
    variables\_to\_average=None,  
    use\_locking=False,  
    name='sync\_replicas'  
)

Construct a sync\_replicas optimizer. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: The **SyncReplicaOptimizer** class is deprecated. For synchrononous training, please use [Distribution Strategies](https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute).

#### Args:

* **opt**: The actual optimizer that will be used to compute and apply the gradients. Must be one of the Optimizer classes.
* **replicas\_to\_aggregate**: number of replicas to aggregate for each variable update.
* **total\_num\_replicas**: Total number of tasks/workers/replicas, could be different from replicas\_to\_aggregate. If total\_num\_replicas > replicas\_to\_aggregate: it is backup\_replicas + replicas\_to\_aggregate. If total\_num\_replicas < replicas\_to\_aggregate: Replicas compute multiple batches per update to variables.
* **variable\_averages**: Optional ExponentialMovingAverage object, used to maintain moving averages for the variables passed in variables\_to\_average.
* **variables\_to\_average**: a list of variables that need to be averaged. Only needed if variable\_averages is passed in.
* **use\_locking**: If True use locks for update operation.
* **name**: string. Optional name of the returned operation.

## Methods

### apply\_gradients

apply\_gradients(  
    grads\_and\_vars,  
    global\_step=None,  
    name=None  
)

Apply gradients to variables.

This contains most of the synchronization implementation and also wraps the apply\_gradients() from the real optimizer.

#### Args:

* **grads\_and\_vars**: List of (gradient, variable) pairs as returned by compute\_gradients().
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **name**: Optional name for the returned operation. Default to the name passed to the Optimizer constructor.

#### Returns:

* **train\_op**: The op to dequeue a token so the replicas can exit this batch and start the next one. This is executed by each replica.

#### Raises:

* **ValueError**: If the grads\_and\_vars is empty.
* **ValueError**: If global step is not provided, the staleness cannot be checked.

### compute\_gradients

compute\_gradients(  
    \*args,  
    \*\*kwargs  
)

Compute gradients of "loss" for the variables in "var\_list".

This simply wraps the compute\_gradients() from the real optimizer. The gradients will be aggregated in the apply\_gradients() so that user can modify the gradients like clipping with per replica global norm if needed. The global norm with aggregated gradients can be bad as one replica's huge gradients can hurt the gradients from other replicas.

#### Args:

* **\*args**: Arguments for compute\_gradients().
* **\*\*kwargs**: Keyword arguments for compute\_gradients().

#### Returns:

A list of (gradient, variable) pairs.

### get\_chief\_queue\_runner

get\_chief\_queue\_runner()

Returns the QueueRunner for the chief to execute.

This includes the operations to synchronize replicas: aggregate gradients, apply to variables, increment global step, insert tokens to token queue.

Note that this can only be called after calling apply\_gradients() which actually generates this queuerunner.

#### Returns:

A QueueRunner for chief to execute.

#### Raises:

* **ValueError**: If this is called before apply\_gradients().

### get\_init\_tokens\_op

get\_init\_tokens\_op(num\_tokens=-1)

Returns the op to fill the sync\_token\_queue with the tokens.

This is supposed to be executed in the beginning of the chief/sync thread so that even if the total\_num\_replicas is less than replicas\_to\_aggregate, the model can still proceed as the replicas can compute multiple steps per variable update. Make sure: num\_tokens >= replicas\_to\_aggregate - total\_num\_replicas.

#### Args:

* **num\_tokens**: Number of tokens to add to the queue.

#### Returns:

An op for the chief/sync replica to fill the token queue.

#### Raises:

* **ValueError**: If this is called before apply\_gradients().
* **ValueError**: If num\_tokens are smaller than replicas\_to\_aggregate - total\_num\_replicas.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    \*args,  
    \*\*kwargs  
)

Return a slot named "name" created for "var" by the Optimizer.

This simply wraps the get\_slot() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

The Variable for the slot if it was created, None otherwise.

### get\_slot\_names

get\_slot\_names(  
    \*args,  
    \*\*kwargs  
)

Return a list of the names of slots created by the Optimizer.

This simply wraps the get\_slot\_names() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

A list of strings.

### make\_session\_run\_hook

make\_session\_run\_hook(  
    is\_chief,  
    num\_tokens=-1  
)

Creates a hook to handle SyncReplicasHook ops such as initialization.

### minimize

minimize(  
    loss,  
    global\_step=None,  
    var\_list=None,  
    gate\_gradients=GATE\_OP,  
    aggregation\_method=None,  
    colocate\_gradients\_with\_ops=False,  
    name=None,  
    grad\_loss=None  
)

Add operations to minimize loss by updating var\_list.

This method simply combines calls compute\_gradients() and apply\_gradients(). If you want to process the gradient before applying them call compute\_gradients() and apply\_gradients()explicitly instead of using this function.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **global\_step**: Optional Variable to increment by one after the variables have been updated.
* **var\_list**: Optional list or tuple of Variable objects to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKeys.TRAINABLE\_VARIABLES.
* **gate\_gradients**: How to gate the computation of gradients. Can be GATE\_NONE, GATE\_OP, orGATE\_GRAPH.
* **aggregation\_method**: Specifies the method used to combine gradient terms. Valid values are defined in the class AggregationMethod.
* **colocate\_gradients\_with\_ops**: If True, try colocating gradients with the corresponding op.
* **name**: Optional name for the returned operation.
* **grad\_loss**: Optional. A Tensor holding the gradient computed for loss.

#### Returns:

An Operation that updates the variables in var\_list. If global\_step was not None, that operation also increments global\_step.

#### Raises:

* **ValueError**: If some of the variables are not Variable objects.

#### Eager Compatibility

When eager execution is enabled, loss should be a Python function that takes no arguments and computes the value to be minimized. Minimization (and gradient computation) is done with respect to the elements of var\_list if not None, else with respect to any trainable variables created during the execution of the loss function. gate\_gradients, aggregation\_method,colocate\_gradients\_with\_ops and grad\_loss are ignored when eager execution is enabled.

### variables

variables()

Fetches a list of optimizer variables in the default graph.

This wraps variables() from the actual optimizer. It does not include the SyncReplicasOptimizer's local step.

#### Returns:

A list of variables.

## Class Members

* GATE\_GRAPH = 2
* GATE\_NONE = 0
* GATE\_OP = 1

# tf.compat.v1.train.update\_checkpoint\_state

Updates the content of the 'checkpoint' file. (deprecated)

tf.compat.v1.train.update\_checkpoint\_state(  
    save\_dir,  
    model\_checkpoint\_path,  
    all\_model\_checkpoint\_paths=None,  
    latest\_filename=None,  
    all\_model\_checkpoint\_timestamps=None,  
    last\_preserved\_timestamp=None  
)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use [**tf.train.CheckpointManager**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager) to manage checkpoints rather than manually editing the Checkpoint proto.

This updates the checkpoint file containing a CheckpointState proto.

#### Args:

* **save\_dir**: Directory where the model was saved.
* **model\_checkpoint\_path**: The checkpoint file.
* **all\_model\_checkpoint\_paths**: List of strings. Paths to all not-yet-deleted checkpoints, sorted from oldest to newest. If this is a non-empty list, the last element must be equal to model\_checkpoint\_path. These paths are also saved in the CheckpointState proto.
* **latest\_filename**: Optional name of the checkpoint file. Default to 'checkpoint'.
* **all\_model\_checkpoint\_timestamps**: Optional list of timestamps (floats, seconds since the Epoch) indicating when the checkpoints in all\_model\_checkpoint\_paths were created.
* **last\_preserved\_timestamp**: A float, indicating the number of seconds since the Epoch when the last preserved checkpoint was written, e.g. due to a keep\_checkpoint\_every\_n\_hoursparameter (see tf.contrib.checkpoint.CheckpointManager for an implementation).

#### Raises:

* **RuntimeError**: If any of the model checkpoint paths conflict with the file containing CheckpointSate.

# tf.compat.v1.train.warm\_start

Warm-starts a model using the given settings.

tf.compat.v1.train.warm\_start(  
    ckpt\_to\_initialize\_from,  
    vars\_to\_warm\_start='.\*',  
    var\_name\_to\_vocab\_info=None,  
    var\_name\_to\_prev\_var\_name=None  
)

Defined in [python/training/warm\_starting\_util.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/warm_starting_util.py).

If you are using a tf.estimator.Estimator, this will automatically be called during training.

#### Args:

* **ckpt\_to\_initialize\_from**: [Required] A string specifying the directory with checkpoint file(s) or path to checkpoint from which to warm-start the model parameters.
* **vars\_to\_warm\_start**: [Optional] One of the following:
  + A regular expression (string) that captures which variables to warm-start (see tf.compat.v1.get\_collection). This expression will only consider variables in the TRAINABLE\_VARIABLES collection -- if you need to warm-start non\_TRAINABLE vars (such as optimizer accumulators or batch norm statistics), please use the below option.
  + A list of strings, each a regex scope provided to tf.compat.v1.get\_collection with GLOBAL\_VARIABLES (please see tf.compat.v1.get\_collection). For backwards compatibility reasons, this is separate from the single-string argument type.
  + A list of Variables to warm-start. If you do not have access to the Variable objects at the call site, please use the above option.
  + None, in which case only TRAINABLE variables specified in var\_name\_to\_vocab\_infowill be warm-started.

Defaults to '.\*', which warm-starts all variables in the TRAINABLE\_VARIABLES collection. Note that this excludes variables such as accumulators and moving statistics from batch norm.

* **var\_name\_to\_vocab\_info**: [Optional] Dict of variable names (strings) to[tf.estimator.VocabInfo](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/VocabInfo). The variable names should be "full" variables, not the names of the partitions. If not explicitly provided, the variable is assumed to have no (changes to) vocabulary.
* **var\_name\_to\_prev\_var\_name**: [Optional] Dict of variable names (strings) to name of the previously-trained variable in ckpt\_to\_initialize\_from. If not explicitly provided, the name of the variable is assumed to be same between previous checkpoint and current model. Note that this has no effect on the set of variables that is warm-started, and only controls name mapping (use vars\_to\_warm\_start for controlling what variables to warm-start).

#### Raises:

* **ValueError**: If the WarmStartSettings contains prev\_var\_name or VocabInfo configuration for variable names that are not used. This is to ensure a stronger check for variable configuration than relying on users to examine the logs.

# tf.compat.v1.train.WorkerSessionCreator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator#top_of_page)
* [Class WorkerSessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator#class_workersessioncreator)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator#methods)
  + [create\_session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/WorkerSessionCreator#create_session)

## Class WorkerSessionCreator

Creates a tf.compat.v1.Session for a worker.

Inherits From: [SessionCreator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/SessionCreator)

Defined in [python/training/monitored\_session.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/monitored_session.py).

## \_\_init\_\_

\_\_init\_\_(  
    scaffold=None,  
    master='',  
    config=None,  
    max\_wait\_secs=(30 \* 60)  
)

Initializes a worker session creator.

#### Args:

* **scaffold**: A Scaffold used for gathering or building supportive ops. If not specified a default one is created. It's used to finalize the graph.
* **master**: String representation of the TensorFlow master to use.
* **config**: ConfigProto proto used to configure the session.
* **max\_wait\_secs**: Maximum time to wait for the session to become available.

## Methods

### create\_session

create\_session()

Module: tf.compat.v1.train.experimental

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

Public API for tf.train.experimental namespace.

Classes

[class DynamicLossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale): Loss scale that dynamically adjusts itself.

[class FixedLossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/FixedLossScale): Loss scale with a fixed value.

[class LossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/LossScale): Loss scale base class.

[class MixedPrecisionLossScaleOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/experimental/MixedPrecisionLossScaleOptimizer): An optimizer that applies loss scaling.

[class PythonState](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState): A mixin for putting Python state in an object-based checkpoint.

Functions

[disable\_mixed\_precision\_graph\_rewrite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/experimental/disable_mixed_precision_graph_rewrite): Disables the mixed precision graph rewrite.

[enable\_mixed\_precision\_graph\_rewrite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/experimental/enable_mixed_precision_graph_rewrite): Enable mixed precision via a graph rewrite.

# tf.compat.v1.train.experimental.disable\_mixed\_precision\_graph\_rewrite

Disables the mixed precision graph rewrite.

tf.compat.v1.train.experimental.disable\_mixed\_precision\_graph\_rewrite()

Defined in [python/training/experimental/mixed\_precision.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/experimental/mixed_precision.py).

After this is called, the mixed precision graph rewrite will no longer run for new Sessions, and so float32 operations will no longer be converted to float16 in such Sessions. However, any existing Sessions will continue to have the graph rewrite enabled if they were created afterenable\_mixed\_precision\_graph\_rewrite was called but beforedisable\_mixed\_precision\_graph\_rewrite was called.

This does not undo the effects of loss scaling. Any optimizers wrapped with a LossScaleOptimizer will continue to do loss scaling, although this loss scaling will no longer be useful if the optimizer is used in new Sessions, as the graph rewrite no longer converts the graph to use float16.

This function is useful for unit testing. A unit tests can test using the mixed precision graph rewrite, then disable it so future unit tests continue using float32. If this is done, unit tests should not share a single session, as enable\_mixed\_precision\_graph\_rewrite anddisable\_mixed\_precision\_graph\_rewrite have no effect on existing sessions.

# tf.compat.v1.train.experimental.enable\_mixed\_precision\_graph\_rewrite

Enable mixed precision via a graph rewrite.

tf.compat.v1.train.experimental.enable\_mixed\_precision\_graph\_rewrite(  
    opt,  
    loss\_scale='dynamic'  
)

Defined in [python/training/experimental/mixed\_precision.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/experimental/mixed_precision.py).

Mixed precision is the use of both float16 and float32 when training a model, and is used to make the model run faster. This function will use mixed precision to speed up the execution time of your model when run on a GPU. It does this by changing the dtype of certain operations in the graph from float32 to float16.

This function additionally wraps an Optimizer with a LossScaleOptimizer, which is required to prevent underflow in the float16 tensors during the backwards pass. An optimizer must be passed to this function, which will then be wrapped to use loss scaling.

When this function is used, gradients should only be computed and applied with the returned optimizer, either by calling opt.minimize() or opt.compute\_gradients() followed by opt.apply\_gradients(). Gradients should not be computed with [tf.gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gradients) or [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape). This is because the returned optimizer will apply loss scaling, and[tf.gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gradients)/[tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape) will not. If you do directly use [tf.gradients](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gradients) or [tf.GradientTape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/GradientTape), your model may train to a worse quality.

When eager execution is enabled, the mixed precision graph rewrite is only enabled within [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function)s, as outside [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function)s, there is no graph.

When enabled, mixed precision is only used on Volta GPUs and above. The parts of the graph on CPUs and TPUs are untouched by the graph rewrite.

#### Args:

* **opt**: An instance of a [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) or a tf.train.Optimizer.
* **loss\_scale**: Either an int/float, the string "dynamic", or an instance of a[tf.train.experimental.LossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/LossScale). The loss scale to use. It is recommended to keep this as its default value of "dynamic".

#### Returns:

A version of opt that will use loss scaling to prevent underflow.

Module: tf.train

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

Support for training models.

See the [Training](https://tensorflow.org/api_guides/python/train) guide.

Modules

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

Classes

[class BytesList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList)

[class Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint): Groups trackable objects, saving and restoring them.

[class CheckpointManager](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager): Deletes old checkpoints.

[class ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef)

[class ClusterSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec): Represents a cluster as a set of "tasks", organized into "jobs".

[class Coordinator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator): A coordinator for threads.

[class Example](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example)

[class ExponentialMovingAverage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage): Maintains moving averages of variables by employing an exponential decay.

[class Feature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature)

[class FeatureList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList)

[class FeatureLists](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists)

[class Features](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features)

[class FloatList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList)

[class Int64List](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List)

[class JobDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef)

[class SequenceExample](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample)

[class ServerDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef)

Functions

[checkpoints\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/checkpoints_iterator): Continuously yield new checkpoint files as they appear.

[get\_checkpoint\_state(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/get_checkpoint_state): Returns CheckpointState proto from the "checkpoint" file.

[latest\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint): Finds the filename of latest saved checkpoint file.

[list\_variables(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/list_variables): Returns list of all variables in the checkpoint.

[load\_checkpoint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/load_checkpoint): Returns CheckpointReader for checkpoint found in ckpt\_dir\_or\_file.

[load\_variable(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/load_variable): Returns the tensor value of the given variable in the checkpoint.

# tf.train.BytesList

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#top_of_page)
* [Class BytesList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#class_byteslist)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#properties)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/BytesList#value)

## Class BytesList

### Aliases:

* Class tf.compat.v1.train.BytesList
* Class tf.compat.v2.train.BytesList
* Class tf.train.BytesList

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Properties

### value

repeated bytes value

# tf.train.Checkpoint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#top_of_page)
* [Class Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#class_checkpoint)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#used_in_the_tutorials)

## Class Checkpoint

Groups trackable objects, saving and restoring them.

### Aliases:

* Class tf.compat.v2.train.Checkpoint
* Class tf.train.Checkpoint

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

### Used in the tutorials:

* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Checkpoint's constructor accepts keyword arguments whose values are types that contain trackable state, such as [tf.keras.optimizers.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/optimizers/Optimizer) implementations, [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable), tf.keras.Layer implementations, or [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) implementations. It saves these values with a checkpoint, and maintains a save\_counter for numbering checkpoints.

#### Example usage:

import tensorflow as tf  
import os  
  
checkpoint\_directory = "/tmp/training\_checkpoints"  
checkpoint\_prefix = os.path.join(checkpoint\_directory, "ckpt")  
  
checkpoint = tf.train.Checkpoint(optimizer=optimizer, model=model)  
status = checkpoint.restore(tf.train.latest\_checkpoint(checkpoint\_directory))  
for \_ in range(num\_training\_steps):  
  optimizer.minimize( ... )  # Variables will be restored on creation.  
status.assert\_consumed()  # Optional sanity checks.  
checkpoint.save(file\_prefix=checkpoint\_prefix)

Checkpoint.save and Checkpoint.restore write and read object-based checkpoints, in contrast to TensorFlow 1.x's [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) which writes and reads variable.name based checkpoints. Object-based checkpointing saves a graph of dependencies between Python objects (Layers, Optimizers, Variables, etc.) with named edges, and this graph is used to match variables when restoring a checkpoint. It can be more robust to changes in the Python program, and helps to support restore-on-create for variables.

Checkpoint objects have dependencies on the objects passed as keyword arguments to their constructors, and each dependency is given a name that is identical to the name of the keyword argument for which it was created. TensorFlow classes like Layers and Optimizers will automatically add dependencies on their variables (e.g. "kernel" and "bias" for[tf.keras.layers.Dense](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers/Dense)). Inheriting from [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) makes managing dependencies easy in user-defined classes, since Model hooks into attribute assignment. For example:

class Regress(tf.keras.Model):  
  
  def \_\_init\_\_(self):  
    super(Regress, self).\_\_init\_\_()  
    self.input\_transform = tf.keras.layers.Dense(10)  
    # ...  
  
  def call(self, inputs):  
    x = self.input\_transform(inputs)  
    # ...

This Model has a dependency named "input\_transform" on its Dense layer, which in turn depends on its variables. As a result, saving an instance of Regress using [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) will also save all the variables created by the Dense layer.

When variables are assigned to multiple workers, each worker writes its own section of the checkpoint. These sections are then merged/re-indexed to behave as a single checkpoint. This avoids copying all variables to one worker, but does require that all workers see a common filesystem.

While [tf.keras.Model.save\_weights](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#save_weights) and [tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) save in the same format, note that the root of the resulting checkpoint is the object the save method is attached to. This means saving a [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) using save\_weights and loading into a [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) with a Model attached (or vice versa) will not match the Model's variables. See the [guide to training checkpoints](https://www.tensorflow.org/alpha/guide/checkpoints) for details. Prefer [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) over [tf.keras.Model.save\_weights](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model#save_weights) for training checkpoints.

#### Attributes:

* **save\_counter**: Incremented when save() is called. Used to number checkpoints.

## \_\_init\_\_

\_\_init\_\_(\*\*kwargs)

Group objects into a training checkpoint.

#### Args:

* **\*\*kwargs**: Keyword arguments are set as attributes of this object, and are saved with the checkpoint. Values must be trackable objects.

#### Raises:

* **ValueError**: If objects in kwargs are not trackable.

## Properties

### save\_counter

An integer variable which starts at zero and is incremented on save.

Used to number checkpoints.

#### Returns:

The save counter variable.

## Methods

### restore

restore(save\_path)

Restore a training checkpoint.

Restores this Checkpoint and any objects it depends on.

Either assigns values immediately if variables to restore have been created already, or defers restoration until the variables are created. Dependencies added after this call will be matched if they have a corresponding object in the checkpoint (the restore request will queue in any trackable object waiting for the expected dependency to be added).

To ensure that loading is complete and no more assignments will take place, use the assert\_consumed() method of the status object returned by restore:

checkpoint = tf.train.Checkpoint( ... )  
checkpoint.restore(path).assert\_consumed()

An exception will be raised if any Python objects in the dependency graph were not found in the checkpoint, or if any checkpointed values do not have a matching Python object.

Name-based [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) checkpoints from TensorFlow 1.x can be loaded using this method. Names are used to match variables. Re-encode name-based checkpoints using [tf.train.Checkpoint.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#save) as soon as possible.

#### Args:

* **save\_path**: The path to the checkpoint, as returned by save or[tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). If None (as when there is no latest checkpoint for [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint) to return), returns an object which may run initializers for objects in the dependency graph. If the checkpoint was written by the name-based[tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver), names are used to match variables.

#### Returns:

A load status object, which can be used to make assertions about the status of a checkpoint restoration.

The returned status object has the following methods:

* assert\_consumed(): Raises an exception if any variables/objects are unmatched: either checkpointed values which don't have a matching Python object or Python objects in the dependency graph with no values in the checkpoint. This method returns the status object, and so may be chained with other assertions.
* assert\_existing\_objects\_matched(): Raises an exception if any existing Python objects in the dependency graph are unmatched. Unlike assert\_consumed, this assertion will pass if values in the checkpoint have no corresponding Python objects. For example a tf.keras.Layer object which has not yet been built, and so has not created any variables, will pass this assertion but fail assert\_consumed. Useful when loading part of a larger checkpoint into a new Python program, e.g. a training checkpoint with a [tf.compat.v1.train.Optimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Optimizer)was saved but only the state required for inference is being loaded. This method returns the status object, and so may be chained with other assertions.
* assert\_nontrivial\_match(): Asserts that something aside from the root object was matched. This is a very weak assertion, but is useful for sanity checking in library code where objects may exist in the checkpoint which haven't been created in Python and some Python objects may not have a checkpointed value.
* expect\_partial(): Silence warnings about incomplete checkpoint restores. Warnings are otherwise printed for unused parts of the checkpoint file or object when the Checkpoint object is deleted (often at program shutdown).

### save

save(file\_prefix)

Saves a training checkpoint and provides basic checkpoint management.

The saved checkpoint includes variables created by this object and any trackable objects it depends on at the time Checkpoint.save() is called.

save is a basic convenience wrapper around the write method, sequentially numbering checkpoints using save\_counter and updating the metadata used by [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). More advanced checkpoint management, for example garbage collection and custom numbering, may be provided by other utilities which also wrap write (tf.contrib.checkpoint.CheckpointManagerfor example).

#### Args:

* **file\_prefix**: A prefix to use for the checkpoint filenames (/path/to/directory/and\_a\_prefix). Names are generated based on this prefix and Checkpoint.save\_counter.

#### Returns:

The full path to the checkpoint.

### write

write(file\_prefix)

Writes a training checkpoint.

The checkpoint includes variables created by this object and any trackable objects it depends on at the time Checkpoint.write() is called.

write does not number checkpoints, increment save\_counter, or update the metadata used by [tf.train.latest\_checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint). It is primarily intended for use by higher level checkpoint management utilities. save provides a very basic implementation of these features.

#### Args:

* **file\_prefix**: A prefix to use for the checkpoint filenames (/path/to/directory/and\_a\_prefix).

#### Returns:

The full path to the checkpoint (i.e. file\_prefix).

# tf.train.CheckpointManager

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager#top_of_page)
* [Class CheckpointManager](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager#class_checkpointmanager)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/CheckpointManager#used_in_the_tutorials)

## Class CheckpointManager

Deletes old checkpoints.

### Aliases:

* Class tf.compat.v1.train.CheckpointManager
* Class tf.compat.v2.train.CheckpointManager
* Class tf.train.CheckpointManager

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

### Used in the guide:

* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

#### Example usage:

import tensorflow as tf  
checkpoint = tf.train.Checkpoint(optimizer=optimizer, model=model)  
manager = tf.contrib.checkpoint.CheckpointManager(  
    checkpoint, directory="/tmp/model", max\_to\_keep=5)  
status = checkpoint.restore(manager.latest\_checkpoint)  
while True:  
  # train  
  manager.save()

CheckpointManager preserves its own state across instantiations (see the \_\_init\_\_documentation for details). Only one should be active in a particular directory at a time.

## \_\_init\_\_

\_\_init\_\_(  
    checkpoint,  
    directory,  
    max\_to\_keep,  
    keep\_checkpoint\_every\_n\_hours=None,  
    checkpoint\_name='ckpt'  
)

Configure a CheckpointManager for use in directory.

If a CheckpointManager was previously used in directory, its state will be restored. This includes the list of managed checkpoints and the timestamp bookkeeping necessary to supportkeep\_checkpoint\_every\_n\_hours. The behavior of the new CheckpointManager will be the same as the previous CheckpointManager, including cleaning up existing checkpoints if appropriate.

Checkpoints are only considered for deletion just after a new checkpoint has been added. At that point, max\_to\_keep checkpoints will remain in an "active set". Once a checkpoint is preserved bykeep\_checkpoint\_every\_n\_hours it will not be deleted by this CheckpointManager or any future CheckpointManager instantiated in directory (regardless of the new setting ofkeep\_checkpoint\_every\_n\_hours). The max\_to\_keep checkpoints in the active set may be deleted by this CheckpointManager or a future CheckpointManager instantiated in directory (subject to its max\_to\_keep and keep\_checkpoint\_every\_n\_hours settings).

#### Args:

* **checkpoint**: The [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint) instance to save and manage checkpoints for.
* **directory**: The path to a directory in which to write checkpoints. A special file named "checkpoint" is also written to this directory (in a human-readable text format) which contains the state of the CheckpointManager.
* **max\_to\_keep**: An integer, the number of checkpoints to keep. Unless preserved by keep\_checkpoint\_every\_n\_hours, checkpoints will be deleted from the active set, oldest first, until only max\_to\_keep checkpoints remain. If None, no checkpoints are deleted and everything stays in the active set. Note that max\_to\_keep=None will keep all checkpoint paths in memory and in the checkpoint state protocol buffer on disk.
* **keep\_checkpoint\_every\_n\_hours**: Upon removal from the active set, a checkpoint will be preserved if it has been at least keep\_checkpoint\_every\_n\_hours since the last preserved checkpoint. The default setting of None does not preserve any checkpoints in this way.
* **checkpoint\_name**: Custom name for the checkpoint file.

#### Raises:

* **ValueError**: If max\_to\_keep is not a positive integer.

## Properties

### checkpoints

A list of managed checkpoints.

Note that checkpoints saved due to keep\_checkpoint\_every\_n\_hours will not show up in this list (to avoid ever-growing filename lists).

#### Returns:

A list of filenames, sorted from oldest to newest.

### latest\_checkpoint

The prefix of the most recent checkpoint in directory.

Equivalent to tf.train.latest\_checkpoint(directory) where directory is the constructor argument to CheckpointManager.

Suitable for passing to [tf.train.Checkpoint.restore](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#restore) to resume training.

#### Returns:

The checkpoint prefix. If there are no checkpoints, returns None.

## Methods

### save

save(checkpoint\_number=None)

Creates a new checkpoint and manages it.

#### Args:

* **checkpoint\_number**: An optional integer, or an integer-dtype Variable or Tensor, used to number the checkpoint. If None (default), checkpoints are numbered using checkpoint.save\_counter. Even if checkpoint\_number is provided, save\_counter is still incremented. A user-provided checkpoint\_number is not incremented even if it is a Variable.

#### Returns:

The path to the new checkpoint. It is also recorded in the checkpoints and latest\_checkpointproperties.

# tf.train.checkpoints\_iterator

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

Continuously yield new checkpoint files as they appear.

### Aliases:

* tf.compat.v1.train.checkpoints\_iterator
* tf.compat.v2.train.checkpoints\_iterator
* tf.train.checkpoints\_iterator

tf.train.checkpoints\_iterator(  
    checkpoint\_dir,  
    min\_interval\_secs=0,  
    timeout=None,  
    timeout\_fn=None  
)

Defined in [python/training/checkpoint\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_utils.py).

The iterator only checks for new checkpoints when control flow has been reverted to it. This means it can miss checkpoints if your code takes longer to run between iterations than min\_interval\_secs or the interval at which new checkpoints are written.

The timeout argument is the maximum number of seconds to block waiting for a new checkpoint. It is used in combination with the timeout\_fn as follows:

* If the timeout expires and no timeout\_fn was specified, the iterator stops yielding.
* If a timeout\_fn was specified, that function is called and if it returns a true boolean value the iterator stops yielding.
* If the function returns a false boolean value then the iterator resumes the wait for new checkpoints. At this point the timeout logic applies again.

This behavior gives control to callers on what to do if checkpoints do not come fast enough or stop being generated. For example, if callers have a way to detect that the training has stopped and know that no new checkpoints will be generated, they can provide a timeout\_fn that returns True when the training has stopped. If they know that the training is still going on they return False instead.

#### Args:

* **checkpoint\_dir**: The directory in which checkpoints are saved.
* **min\_interval\_secs**: The minimum number of seconds between yielding checkpoints.
* **timeout**: The maximum number of seconds to wait between checkpoints. If left as None, then the process will wait indefinitely.
* **timeout\_fn**: Optional function to call after a timeout. If the function returns True, then it means that no new checkpoints will be generated and the iterator will exit. The function is called with no arguments.

#### Yields:

String paths to latest checkpoint files as they arrive.

# tf.train.ClusterDef

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef#top_of_page)
* [Class ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef#class_clusterdef)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef#properties)
  + [job](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef#job)

## Class ClusterDef

### Aliases:

* Class tf.compat.v1.train.ClusterDef
* Class tf.compat.v2.train.ClusterDef
* Class tf.train.ClusterDef

Defined in [core/protobuf/cluster.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/protobuf/cluster.proto).

## Properties

### job

repeated JobDef job

# tf.train.ClusterSpec

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#top_of_page)
* [Class ClusterSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#class_clusterspec)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#properties)

## Class ClusterSpec

Represents a cluster as a set of "tasks", organized into "jobs".

### Aliases:

* Class tf.compat.v1.train.ClusterSpec
* Class tf.compat.v2.train.ClusterSpec
* Class tf.train.ClusterSpec

Defined in [python/training/server\_lib.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/server_lib.py).

A [tf.train.ClusterSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec) represents the set of processes that participate in a distributed TensorFlow computation. Every [tf.distribute.Server](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Server) is constructed in a particular cluster.

To create a cluster with two jobs and five tasks, you specify the mapping from job names to lists of network addresses (typically hostname-port pairs).

cluster = tf.train.ClusterSpec({"worker": ["worker0.example.com:2222",  
                                           "worker1.example.com:2222",  
                                           "worker2.example.com:2222"],  
                                "ps": ["ps0.example.com:2222",  
                                       "ps1.example.com:2222"]})

Each job may also be specified as a sparse mapping from task indices to network addresses. This enables a server to be configured without needing to know the identity of (for example) all other worker tasks:

cluster = tf.train.ClusterSpec({"worker": {1: "worker1.example.com:2222"},  
                                "ps": ["ps0.example.com:2222",  
                                       "ps1.example.com:2222"]})

## \_\_init\_\_

\_\_init\_\_(cluster)

Creates a ClusterSpec.

#### Args:

* **cluster**: A dictionary mapping one or more job names to (i) a list of network addresses, or (ii) a dictionary mapping integer task indices to network addresses; or a [tf.train.ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef)protocol buffer.

#### Raises:

* **TypeError**: If cluster is not a dictionary mapping strings to lists of strings, and not a [tf.train.ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef) protobuf.

## Properties

### jobs

Returns a list of job names in this cluster.

#### Returns:

A list of strings, corresponding to the names of jobs in this cluster.

## Methods

### \_\_bool\_\_

\_\_bool\_\_()

### \_\_eq\_\_

\_\_eq\_\_(other)

### \_\_ne\_\_

\_\_ne\_\_(other)

### \_\_nonzero\_\_

\_\_nonzero\_\_()

### as\_cluster\_def

as\_cluster\_def()

Returns a [tf.train.ClusterDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterDef) protocol buffer based on this cluster.

### as\_dict

as\_dict()

Returns a dictionary from job names to their tasks.

For each job, if the task index space is dense, the corresponding value will be a list of network addresses; otherwise it will be a dictionary mapping (sparse) task indices to the corresponding addresses.

#### Returns:

A dictionary mapping job names to lists or dictionaries describing the tasks in those jobs.

### job\_tasks

job\_tasks(job\_name)

Returns a mapping from task ID to address in the given job.

NOTE: For backwards compatibility, this method returns a list. If the given job was defined with a sparse set of task indices, the length of this list may not reflect the number of tasks defined in this job. Use the [tf.train.ClusterSpec.num\_tasks](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ClusterSpec#num_tasks) method to find the number of tasks defined in a particular job.

#### Args:

* **job\_name**: The string name of a job in this cluster.

#### Returns:

A list of task addresses, where the index in the list corresponds to the task index of each task. The list may contain None if the job was defined with a sparse set of task indices.

#### Raises:

* **ValueError**: If job\_name does not name a job in this cluster.

### num\_tasks

num\_tasks(job\_name)

Returns the number of tasks defined in the given job.

#### Args:

* **job\_name**: The string name of a job in this cluster.

#### Returns:

The number of tasks defined in the given job.

#### Raises:

* **ValueError**: If job\_name does not name a job in this cluster.

### task\_address

task\_address(  
    job\_name,  
    task\_index  
)

Returns the address of the given task in the given job.

#### Args:

* **job\_name**: The string name of a job in this cluster.
* **task\_index**: A non-negative integer.

#### Returns:

The address of the given task in the given job.

#### Raises:

* **ValueError**: If job\_name does not name a job in this cluster, or no task with index task\_index is defined in that job.

### task\_indices

task\_indices(job\_name)

Returns a list of valid task indices in the given job.

#### Args:

* **job\_name**: The string name of a job in this cluster.

#### Returns:

A list of valid task indices in the given job.

#### Raises:

* **ValueError**: If job\_name does not name a job in this cluster, or no task with index task\_index is defined in that job.

# tf.train.Coordinator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator#top_of_page)
* [Class Coordinator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator#class_coordinator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Coordinator#properties)

## Class Coordinator

A coordinator for threads.

### Aliases:

* Class tf.compat.v1.train.Coordinator
* Class tf.compat.v2.train.Coordinator
* Class tf.train.Coordinator

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

This class implements a simple mechanism to coordinate the termination of a set of threads.

#### Usage:

# Create a coordinator.  
coord = Coordinator()  
# Start a number of threads, passing the coordinator to each of them.  
...start thread 1...(coord, ...)  
...start thread N...(coord, ...)  
# Wait for all the threads to terminate.  
coord.join(threads)

Any of the threads can call coord.request\_stop() to ask for all the threads to stop. To cooperate with the requests, each thread must check for coord.should\_stop() on a regular basis.coord.should\_stop() returns True as soon as coord.request\_stop() has been called.

A typical thread running with a coordinator will do something like:

while not coord.should\_stop():  
  ...do some work...

#### Exception handling:

A thread can report an exception to the coordinator as part of the request\_stop() call. The exception will be re-raised from the coord.join() call.

#### Thread code:

try:  
  while not coord.should\_stop():  
    ...do some work...  
except Exception as e:  
  coord.request\_stop(e)

#### Main code:

try:  
  ...  
  coord = Coordinator()  
  # Start a number of threads, passing the coordinator to each of them.  
  ...start thread 1...(coord, ...)  
  ...start thread N...(coord, ...)  
  # Wait for all the threads to terminate.  
  coord.join(threads)  
except Exception as e:  
  ...exception that was passed to coord.request\_stop()

To simplify the thread implementation, the Coordinator provides a context handler stop\_on\_exception() that automatically requests a stop if an exception is raised. Using the context handler the thread code above can be written as:

with coord.stop\_on\_exception():  
  while not coord.should\_stop():  
    ...do some work...

#### Grace period for stopping:

After a thread has called coord.request\_stop() the other threads have a fixed time to stop, this is called the 'stop grace period' and defaults to 2 minutes. If any of the threads is still alive after the grace period expires coord.join() raises a RuntimeError reporting the laggards.

try:  
  ...  
  coord = Coordinator()  
  # Start a number of threads, passing the coordinator to each of them.  
  ...start thread 1...(coord, ...)  
  ...start thread N...(coord, ...)  
  # Wait for all the threads to terminate, give them 10s grace period  
  coord.join(threads, stop\_grace\_period\_secs=10)  
except RuntimeError:  
  ...one of the threads took more than 10s to stop after request\_stop()  
  ...was called.  
except Exception:  
  ...exception that was passed to coord.request\_stop()

## \_\_init\_\_

\_\_init\_\_(clean\_stop\_exception\_types=None)

Create a new Coordinator.

#### Args:

* **clean\_stop\_exception\_types**: Optional tuple of Exception types that should cause a clean stop of the coordinator. If an exception of one of these types is reported to request\_stop(ex)the coordinator will behave as if request\_stop(None) was called. Defaults to(tf.errors.OutOfRangeError,) which is used by input queues to signal the end of input. When feeding training data from a Python iterator it is common to add StopIteration to this list.

## Properties

### joined

## Methods

### clear\_stop

clear\_stop()

Clears the stop flag.

After this is called, calls to should\_stop() will return False.

### join

join(  
    threads=None,  
    stop\_grace\_period\_secs=120,  
    ignore\_live\_threads=False  
)

Wait for threads to terminate.

This call blocks until a set of threads have terminated. The set of thread is the union of the threads passed in the threads argument and the list of threads that registered with the coordinator by callingCoordinator.register\_thread().

After the threads stop, if an exc\_info was passed to request\_stop, that exception is re-raised.

Grace period handling: When request\_stop() is called, threads are given 'stop\_grace\_period\_secs' seconds to terminate. If any of them is still alive after that period expires, a RuntimeError is raised. Note that if an exc\_info was passed to request\_stop() then it is raised instead of that RuntimeError.

#### Args:

* **threads**: List of threading.Threads. The started threads to join in addition to the registered threads.
* **stop\_grace\_period\_secs**: Number of seconds given to threads to stop after request\_stop()has been called.
* **ignore\_live\_threads**: If False, raises an error if any of the threads are still alive after stop\_grace\_period\_secs.

#### Raises:

* **RuntimeError**: If any thread is still alive after request\_stop() is called and the grace period expires.

### raise\_requested\_exception

raise\_requested\_exception()

If an exception has been passed to request\_stop, this raises it.

### register\_thread

register\_thread(thread)

Register a thread to join.

#### Args:

* **thread**: A Python thread to join.

### request\_stop

request\_stop(ex=None)

Request that the threads stop.

After this is called, calls to should\_stop() will return True.

**Note:** If an exception is being passed in, in must be in the context of handling the exception (i.e. **try: ... except Exception as ex: ...**) and not a newly created one.

#### Args:

* **ex**: Optional Exception, or Python exc\_info tuple as returned by sys.exc\_info(). If this is the first call to request\_stop() the corresponding exception is recorded and re-raised from join().

### should\_stop

should\_stop()

Check if stop was requested.

#### Returns:

True if a stop was requested.

### stop\_on\_exception

stop\_on\_exception(  
    \*args,  
    \*\*kwds  
)

Context manager to request stop when an Exception is raised.

Code that uses a coordinator must catch exceptions and pass them to the request\_stop() method to stop the other threads managed by the coordinator.

This context handler simplifies the exception handling. Use it as follows:

with coord.stop\_on\_exception():  
  # Any exception raised in the body of the with  
  # clause is reported to the coordinator before terminating  
  # the execution of the body.  
  ...body...

This is completely equivalent to the slightly longer code:

try:  
  ...body...  
except:  
  coord.request\_stop(sys.exc\_info())

#### Yields:

nothing.

### wait\_for\_stop

wait\_for\_stop(timeout=None)

Wait till the Coordinator is told to stop.

#### Args:

* **timeout**: Float. Sleep for up to that many seconds waiting for should\_stop() to become True.

#### Returns:

True if the Coordinator is told stop, False if the timeout expired.

# tf.train.Example

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#top_of_page)
* [Class Example](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#class_example)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#used_in_the_guide)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#properties)
  + [features](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Example#features)

## Class Example

### Aliases:

* Class tf.compat.v1.train.Example
* Class tf.compat.v2.train.Example
* Class tf.train.Example

Defined in [core/example/example.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/example.proto).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Properties

### features

Features features

# tf.train.ExponentialMovingAverage

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage#top_of_page)
* [Class ExponentialMovingAverage](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage#class_exponentialmovingaverage)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ExponentialMovingAverage#properties)

## Class ExponentialMovingAverage

Maintains moving averages of variables by employing an exponential decay.

### Aliases:

* Class tf.compat.v1.train.ExponentialMovingAverage
* Class tf.compat.v2.train.ExponentialMovingAverage
* Class tf.train.ExponentialMovingAverage

Defined in [python/training/moving\_averages.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/moving_averages.py).

When training a model, it is often beneficial to maintain moving averages of the trained parameters. Evaluations that use averaged parameters sometimes produce significantly better results than the final trained values.

The apply() method adds shadow copies of trained variables and add ops that maintain a moving average of the trained variables in their shadow copies. It is used when building the training model. The ops that maintain moving averages are typically run after each training step. The average() and average\_name() methods give access to the shadow variables and their names. They are useful when building an evaluation model, or when restoring a model from a checkpoint file. They help use the moving averages in place of the last trained values for evaluations.

The moving averages are computed using exponential decay. You specify the decay value when creating the ExponentialMovingAverage object. The shadow variables are initialized with the same initial values as the trained variables. When you run the ops to maintain the moving averages, each shadow variable is updated with the formula:

shadow\_variable -= (1 - decay) \* (shadow\_variable - variable)

This is mathematically equivalent to the classic formula below, but the use of an assign\_sub op (the "-=" in the formula) allows concurrent lockless updates to the variables:

shadow\_variable = decay \* shadow\_variable + (1 - decay) \* variable

Reasonable values for decay are close to 1.0, typically in the multiple-nines range: 0.999, 0.9999, etc.

Example usage when creating a training model:

# Create variables.  
var0 = tf.Variable(...)  
var1 = tf.Variable(...)  
# ... use the variables to build a training model...  
...  
# Create an op that applies the optimizer.  This is what we usually  
# would use as a training op.  
opt\_op = opt.minimize(my\_loss, [var0, var1])  
  
# Create an ExponentialMovingAverage object  
ema = tf.train.ExponentialMovingAverage(decay=0.9999)  
  
with tf.control\_dependencies([opt\_op]):  
    # Create the shadow variables, and add ops to maintain moving averages  
    # of var0 and var1. This also creates an op that will update the moving  
    # averages after each training step.  This is what we will use in place  
    # of the usual training op.  
    training\_op = ema.apply([var0, var1])  
  
...train the model by running training\_op...

There are two ways to use the moving averages for evaluations:

* Build a model that uses the shadow variables instead of the variables. For this, use the average() method which returns the shadow variable for a given variable.
* Build a model normally but load the checkpoint files to evaluate by using the shadow variable names. For this use the average\_name() method. See the [tf.compat.v1.train.Saver](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/Saver) for more information on restoring saved variables.

Example of restoring the shadow variable values:

# Create a Saver that loads variables from their saved shadow values.  
shadow\_var0\_name = ema.average\_name(var0)  
shadow\_var1\_name = ema.average\_name(var1)  
saver = tf.compat.v1.train.Saver({shadow\_var0\_name: var0, shadow\_var1\_name:  
var1})  
saver.restore(...checkpoint filename...)  
# var0 and var1 now hold the moving average values

## \_\_init\_\_

\_\_init\_\_(  
    decay,  
    num\_updates=None,  
    zero\_debias=False,  
    name='ExponentialMovingAverage'  
)

Creates a new ExponentialMovingAverage object.

The apply() method has to be called to create shadow variables and add ops to maintain moving averages.

The optional num\_updates parameter allows one to tweak the decay rate dynamically. It is typical to pass the count of training steps, usually kept in a variable that is incremented at each step, in which case the decay rate is lower at the start of training. This makes moving averages move faster. If passed, the actual decay rate used is:

min(decay, (1 + num\_updates) / (10 + num\_updates))

#### Args:

* **decay**: Float. The decay to use.
* **num\_updates**: Optional count of number of updates applied to variables.
* **zero\_debias**: If True, zero debias moving-averages that are initialized with tensors.
* **name**: String. Optional prefix name to use for the name of ops added in apply().

## Properties

### name

The name of this ExponentialMovingAverage object.

## Methods

### apply

apply(var\_list=None)

Maintains moving averages of variables.

var\_list must be a list of Variable or Tensor objects. This method creates shadow variables for all elements of var\_list. Shadow variables for Variable objects are initialized to the variable's initial value. They will be added to the GraphKeys.MOVING\_AVERAGE\_VARIABLES collection. For Tensor objects, the shadow variables are initialized to 0 and zero debiased (see docstring in assign\_moving\_average for more details).

shadow variables are created with trainable=False and added to the GraphKeys.ALL\_VARIABLEScollection. They will be returned by calls to tf.compat.v1.global\_variables().

Returns an op that updates all shadow variables from the current value of their associated variables.

Note that apply() can be called multiple times. When eager execution is enabled each call to apply will update the variables once, so this needs to be called in a loop.

#### Args:

* **var\_list**: A list of Variable or Tensor objects. The variables and Tensors must be of types bfloat16, float16, float32, or float64.

#### Returns:

An Operation that updates the moving averages.

#### Raises:

* **TypeError**: If the arguments are not an allowed type.

### average

average(var)

Returns the Variable holding the average of var.

#### Args:

* **var**: A Variable object.

#### Returns:

A Variable object or None if the moving average of var is not maintained.

### average\_name

average\_name(var)

Returns the name of the Variable holding the average for var.

The typical scenario for ExponentialMovingAverage is to compute moving averages of variables during training, and restore the variables from the computed moving averages during evaluations.

To restore variables, you have to know the name of the shadow variables. That name and the original variable can then be passed to a Saver() object to restore the variable from the moving average value with: saver = tf.compat.v1.train.Saver({ema.average\_name(var): var})

average\_name() can be called whether or not apply() has been called.

#### Args:

* **var**: A Variable object.

#### Returns:

A string: The name of the variable that will be used or was used by the ExponentialMovingAverage class to hold the moving average of var.

### variables\_to\_restore

variables\_to\_restore(moving\_avg\_variables=None)

Returns a map of names to Variables to restore.

If a variable has a moving average, use the moving average variable name as the restore name; otherwise, use the variable name.

For example,

  variables\_to\_restore = ema.variables\_to\_restore()  
  saver = tf.compat.v1.train.Saver(variables\_to\_restore)

Below is an example of such mapping:

  conv/batchnorm/gamma/ExponentialMovingAverage: conv/batchnorm/gamma,  
  conv\_4/conv2d\_params/ExponentialMovingAverage: conv\_4/conv2d\_params,  
  global\_step: global\_step

#### Args:

* **moving\_avg\_variables**: a list of variables that require to use of the moving average variable name to be restored. If None, it will default to variables.moving\_average\_variables() + variables.trainable\_variables()

#### Returns:

A map from restore\_names to variables. The restore\_name is either the original or the moving average version of the variable name, depending on whether the variable name is in the moving\_avg\_variables.

# tf.train.Feature

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature#top_of_page)
* [Class Feature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature#class_feature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Feature#properties)

## Class Feature

### Aliases:

* Class tf.compat.v1.train.Feature
* Class tf.compat.v2.train.Feature
* Class tf.train.Feature

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Properties

### bytes\_list

BytesList bytes\_list

### float\_list

FloatList float\_list

### int64\_list

Int64List int64\_list

# tf.train.FeatureList

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList#top_of_page)
* [Class FeatureList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList#class_featurelist)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList#properties)
  + [feature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureList#feature)

## Class FeatureList

### Aliases:

* Class tf.compat.v1.train.FeatureList
* Class tf.compat.v2.train.FeatureList
* Class tf.train.FeatureList

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

## Properties

### feature

repeated Feature feature

# tf.train.FeatureLists

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#top_of_page)
* [Class FeatureLists](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#class_featurelists)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#aliases)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#child_classes)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#properties)
  + [feature\_list](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists#feature_list)

## Class FeatureLists

### Aliases:

* Class tf.compat.v1.train.FeatureLists
* Class tf.compat.v2.train.FeatureLists
* Class tf.train.FeatureLists

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

## Child Classes

[class FeatureListEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry)

## Properties

### feature\_list

repeated FeatureListEntry feature\_list

# tf.train.FeatureLists.FeatureListEntry

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#top_of_page)
* [Class FeatureListEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#class_featurelistentry)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#properties)
  + [key](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#key)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FeatureLists/FeatureListEntry#value)

## Class FeatureListEntry

### Aliases:

* Class tf.compat.v1.train.FeatureLists.FeatureListEntry
* Class tf.compat.v2.train.FeatureLists.FeatureListEntry
* Class tf.train.FeatureLists.FeatureListEntry

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

## Properties

### key

string key

### value

FeatureList value

# tf.train.Features

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#top_of_page)
* [Class Features](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#class_features)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#used_in_the_tutorials)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#child_classes)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#properties)
  + [feature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features#feature)

## Class Features

### Aliases:

* Class tf.compat.v1.train.Features
* Class tf.compat.v2.train.Features
* Class tf.train.Features

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Child Classes

[class FeatureEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry)

## Properties

### feature

repeated FeatureEntry feature

# tf.train.Features.FeatureEntry

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#top_of_page)
* [Class FeatureEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#class_featureentry)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#properties)
  + [key](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#key)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Features/FeatureEntry#value)

## Class FeatureEntry

### Aliases:

* Class tf.compat.v1.train.Features.FeatureEntry
* Class tf.compat.v2.train.Features.FeatureEntry
* Class tf.train.Features.FeatureEntry

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

## Properties

### key

string key

### value

Feature value

# tf.train.FloatList

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#top_of_page)
* [Class FloatList](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#class_floatlist)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#properties)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/FloatList#value)

## Class FloatList

### Aliases:

* Class tf.compat.v1.train.FloatList
* Class tf.compat.v2.train.FloatList
* Class tf.train.FloatList

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Properties

### value

repeated float value

# tf.train.get\_checkpoint\_state

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

Returns CheckpointState proto from the "checkpoint" file.

### Aliases:

* tf.compat.v1.train.get\_checkpoint\_state
* tf.compat.v2.train.get\_checkpoint\_state
* tf.train.get\_checkpoint\_state

tf.train.get\_checkpoint\_state(  
    checkpoint\_dir,  
    latest\_filename=None  
)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

If the "checkpoint" file contains a valid CheckpointState proto, returns it.

#### Args:

* **checkpoint\_dir**: The directory of checkpoints.
* **latest\_filename**: Optional name of the checkpoint file. Default to 'checkpoint'.

#### Returns:

A CheckpointState if the state was available, None otherwise.

#### Raises:

* **ValueError**: if the checkpoint read doesn't have model\_checkpoint\_path set.

# tf.train.Int64List

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#top_of_page)
* [Class Int64List](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#class_int64list)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#properties)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Int64List#value)

## Class Int64List

### Aliases:

* Class tf.compat.v1.train.Int64List
* Class tf.compat.v2.train.Int64List
* Class tf.train.Int64List

Defined in [core/example/feature.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/feature.proto).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

## Properties

### value

repeated int64 value

# tf.train.JobDef

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#top_of_page)
* [Class JobDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#class_jobdef)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#aliases)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#child_classes)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#properties)
  + [name](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#name)
  + [tasks](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef#tasks)

## Class JobDef

### Aliases:

* Class tf.compat.v1.train.JobDef
* Class tf.compat.v2.train.JobDef
* Class tf.train.JobDef

Defined in [core/protobuf/cluster.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/protobuf/cluster.proto).

## Child Classes

[class TasksEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry)

## Properties

### name

string name

### tasks

repeated TasksEntry tasks

# tf.train.JobDef.TasksEntry

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#top_of_page)
* [Class TasksEntry](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#class_tasksentry)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#properties)
  + [key](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#key)
  + [value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/JobDef/TasksEntry#value)

## Class TasksEntry

### Aliases:

* Class tf.compat.v1.train.JobDef.TasksEntry
* Class tf.compat.v2.train.JobDef.TasksEntry
* Class tf.train.JobDef.TasksEntry

Defined in [core/protobuf/cluster.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/protobuf/cluster.proto).

## Properties

### key

int32 key

### value

string value

# tf.train.latest\_checkpoint

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/latest_checkpoint#used_in_the_tutorials)

Finds the filename of latest saved checkpoint file.

### Aliases:

* tf.compat.v1.train.latest\_checkpoint
* tf.compat.v2.train.latest\_checkpoint
* tf.train.latest\_checkpoint

tf.train.latest\_checkpoint(  
    checkpoint\_dir,  
    latest\_filename=None  
)

Defined in [python/training/checkpoint\_management.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_management.py).

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

### Used in the tutorials:

* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)
* [Distributed training with Keras](https://www.tensorflow.org/beta/tutorials/distribute/keras)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Save and restore models](https://www.tensorflow.org/beta/tutorials/keras/save_and_restore_models)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

#### Args:

* **checkpoint\_dir**: Directory where the variables were saved.
* **latest\_filename**: Optional name for the protocol buffer file that contains the list of most recent checkpoint filenames. See the corresponding argument to Saver.save().

#### Returns:

The full path to the latest checkpoint or None if no checkpoint was found.

# tf.train.list\_variables

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/list_variables#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/list_variables#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/list_variables#used_in_the_guide)

Returns list of all variables in the checkpoint.

### Aliases:

* tf.compat.v1.train.list\_variables
* tf.compat.v2.train.list\_variables
* tf.train.list\_variables

tf.train.list\_variables(ckpt\_dir\_or\_file)

Defined in [python/training/checkpoint\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_utils.py).

### Used in the guide:

* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)

#### Args:

* **ckpt\_dir\_or\_file**: Directory with checkpoints file or path to checkpoint.

#### Returns:

List of tuples (name, shape).

# tf.train.load\_checkpoint

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

Returns CheckpointReader for checkpoint found in ckpt\_dir\_or\_file.

### Aliases:

* tf.compat.v1.train.load\_checkpoint
* tf.compat.v2.train.load\_checkpoint
* tf.train.load\_checkpoint

tf.train.load\_checkpoint(ckpt\_dir\_or\_file)

Defined in [python/training/checkpoint\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_utils.py).

If ckpt\_dir\_or\_file resolves to a directory with multiple checkpoints, reader for the latest checkpoint is returned.

#### Args:

* **ckpt\_dir\_or\_file**: Directory with checkpoints file or path to checkpoint file.

#### Returns:

CheckpointReader object.

#### Raises:

* **ValueError**: If ckpt\_dir\_or\_file resolves to a directory with no checkpoints.

# tf.train.load\_variable

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

Returns the tensor value of the given variable in the checkpoint.

### Aliases:

* tf.compat.v1.train.load\_variable
* tf.compat.v2.train.load\_variable
* tf.train.load\_variable

tf.train.load\_variable(  
    ckpt\_dir\_or\_file,  
    name  
)

Defined in [python/training/checkpoint\_utils.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/checkpoint_utils.py).

#### Args:

* **ckpt\_dir\_or\_file**: Directory with checkpoints file or path to checkpoint.
* **name**: Name of the variable to return.

#### Returns:

A numpy ndarray with a copy of the value of this variable.

# tf.train.SequenceExample

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#top_of_page)
* [Class SequenceExample](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#class_sequenceexample)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#properties)
  + [context](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#context)
  + [feature\_lists](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/SequenceExample#feature_lists)

## Class SequenceExample

### Aliases:

* Class tf.compat.v1.train.SequenceExample
* Class tf.compat.v2.train.SequenceExample
* Class tf.train.SequenceExample

Defined in [core/example/example.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/example/example.proto).

## Properties

### context

Features context

### feature\_lists

FeatureLists feature\_lists

# tf.train.ServerDef

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef#top_of_page)
* [Class ServerDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef#class_serverdef)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef#properties)
  + [cluster](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/ServerDef#cluster)

## Class ServerDef

### Aliases:

* Class tf.compat.v1.train.ServerDef
* Class tf.compat.v2.train.ServerDef
* Class tf.train.ServerDef

Defined in [core/protobuf/tensorflow\_server.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/protobuf/tensorflow_server.proto).

## Properties

### cluster

ClusterDef cluster

### default\_session\_config

ConfigProto default\_session\_config

### job\_name

string job\_name

### protocol

string protocol

### task\_index

int32 task\_index

Module: tf.train.experimental

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

Public API for tf.train.experimental namespace.

Classes

[class DynamicLossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale): Loss scale that dynamically adjusts itself.

[class FixedLossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/FixedLossScale): Loss scale with a fixed value.

[class LossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/LossScale): Loss scale base class.

[class PythonState](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState): A mixin for putting Python state in an object-based checkpoint.

# tf.train.experimental.DynamicLossScale

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale#top_of_page)
* [Class DynamicLossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale#class_dynamiclossscale)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/DynamicLossScale#properties)

## Class DynamicLossScale

Loss scale that dynamically adjusts itself.

Inherits From: [LossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/LossScale)

### Aliases:

* Class tf.compat.v1.train.experimental.DynamicLossScale
* Class tf.compat.v2.train.experimental.DynamicLossScale
* Class tf.train.experimental.DynamicLossScale

Defined in [python/training/experimental/loss\_scale.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/experimental/loss_scale.py).

Dynamic loss scaling works by adjusting the loss scale as training progresses. The goal is to keep the loss scale as high as possible without overflowing the gradients. As long as the gradients do not overflow, raising the loss scale never hurts.

The algorithm starts by setting the loss scale to an initial value. Every N steps that the gradients are finite, the loss scale is increased by some factor. However, if a NaN or Inf gradient is found, the gradients for that step are not applied, and the loss scale is decreased by the factor. This process tends to keep the loss scale as high as possible without gradients overflowing.

## \_\_init\_\_

\_\_init\_\_(  
    initial\_loss\_scale=(2 \*\* 15),  
    increment\_period=2000,  
    multiplier=2.0  
)

Creates the dynamic loss scale.

#### Args:

* **initial\_loss\_scale**: A Python float. The loss scale to use at the beginning. It's better to start this at a very high number, because a loss scale that is too high gets lowered far more quickly than a loss scale that is to low gets raised. The default is 2 \*\* 15, which is approximately half the maximum float16 value.
* **increment\_period**: Increases loss scale every increment\_period consecutive steps that finite gradients are encountered. If a nonfinite gradient is encountered, the count is reset back to zero.
* **multiplier**: The multiplier to use when increasing or decreasing the loss scale.

## Properties

### increment\_period

### initial\_loss\_scale

### multiplier

## Methods

### \_\_call\_\_

\_\_call\_\_()

### from\_config

from\_config(  
    cls,  
    config  
)

Creates the LossScale from its config.

### get\_config

get\_config()

### update

update(grads)

Updates loss scale based on if gradients are finite in current step.

# tf.train.experimental.FixedLossScale

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

## Class FixedLossScale

Loss scale with a fixed value.

Inherits From: [LossScale](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/LossScale)

### Aliases:

* Class tf.compat.v1.train.experimental.FixedLossScale
* Class tf.compat.v2.train.experimental.FixedLossScale
* Class tf.train.experimental.FixedLossScale

Defined in [python/training/experimental/loss\_scale.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/experimental/loss_scale.py).

The loss scale is not updated for the lifetime of instances of this class. A given instance of this class always returns the same number when called.

## \_\_init\_\_

\_\_init\_\_(loss\_scale\_value)

Creates the fixed loss scale.

#### Args:

* **loss\_scale\_value**: A Python float. Its ideal value varies depending on models to run. Choosing a too small loss\_scale might affect model quality; a too big loss\_scale might cause inf or nan. There is no single right loss\_scale to apply. There is no harm choosing a relatively big number as long as no nan or inf is encountered in training.

#### Raises:

* **ValueError**: If loss\_scale is less than 1.

## Methods

### \_\_call\_\_

\_\_call\_\_()

### from\_config

from\_config(  
    cls,  
    config  
)

Creates the LossScale from its config.

### get\_config

get\_config()

### update

update(grads)

# tf.train.experimental.LossScale

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

## Class LossScale

Loss scale base class.

### Aliases:

* Class tf.compat.v1.train.experimental.LossScale
* Class tf.compat.v2.train.experimental.LossScale
* Class tf.train.experimental.LossScale

Defined in [python/training/experimental/loss\_scale.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/experimental/loss_scale.py).

Loss scaling is a process that multiplies the loss by a multiplier called the loss scale, and divides each gradient by the same multiplier. The pseudocode for this process is:

loss = ...  
loss \*= loss\_scale  
grads = gradients(loss, vars)  
grads /= loss\_scale

Mathematically, loss scaling has no effect, but can help avoid numerical underflow in intermediate gradients when float16 tensors are used for mixed precision training. By multiplying the loss, each intermediate gradient will have the same multiplier applied.

Instances of this class represent a loss scale. Calling instances of this class returns the loss scale as a scalar float32 tensor, while method update() updates the loss scale depending on the values of the gradients. Optimizers use instances of this class to scale loss and gradients.

## \_\_init\_\_

\_\_init\_\_()

Initializes the loss scale class.

## Methods

### \_\_call\_\_

\_\_call\_\_()

Returns the current loss scale as a scalar float32 tensor.

### from\_config

@classmethod  
from\_config(  
    cls,  
    config  
)

Creates the LossScale from its config.

### get\_config

get\_config()

Returns the config of this loss scale.

### update

update(grads)

Updates the value of the loss scale.

The loss scale will be potentially updated, based on the value of grads. The tensor returned by calling this class is only updated when this function is evaluated.

In eager mode, this directly updates the loss scale, so that calling \_\_call\_\_ will return the newly updated loss scale. In graph mode, this returns an op that, when evaluated, updates the loss scale.

This function also returns a should\_apply\_gradients bool. If False, gradients should not be applied to the variables that step, as nonfinite gradients were found, and the loss scale has been be updated to reduce the chance of finding nonfinite gradients in the next step. Some loss scale classes will always return True, as they cannot adjust themselves in response to nonfinite gradients.

When a DistributionStrategy is used, this function may only be called in a cross-replica context.

#### Args:

* **grads**: A list of unscaled gradients, each which is the gradient of the loss with respect to a weight. The gradients should have already been divided by the loss scale being before passed to this function. 'None' gradients are accepted, and are ignored.

#### Returns:

* **update\_op**: In eager mode, None. In graph mode, an op to update the loss scale.
* **should\_apply\_gradients**: Either a bool or a scalar boolean tensor. If False, the caller should skip applying grads to the variables this step.

# tf.train.experimental.PythonState

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#top_of_page)
* [Class PythonState](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#class_pythonstate)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#methods)
  + [deserialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#deserialize)
  + [serialize](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/experimental/PythonState#serialize)

## Class PythonState

A mixin for putting Python state in an object-based checkpoint.

### Aliases:

* Class tf.compat.v1.train.experimental.PythonState
* Class tf.compat.v2.train.experimental.PythonState
* Class tf.train.experimental.PythonState

Defined in [python/training/tracking/python\_state.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/training/tracking/python_state.py).

This is an abstract class which allows extensions to TensorFlow's object-based checkpointing (see [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint)). For example a wrapper for NumPy arrays:

import io  
import numpy  
  
class NumpyWrapper(tf.train.experimental.PythonState):  
  
  def \_\_init\_\_(self, array):  
    self.array = array  
  
  def serialize(self):  
    string\_file = io.BytesIO()  
    try:  
      numpy.save(string\_file, self.array, allow\_pickle=False)  
      serialized = string\_file.getvalue()  
    finally:  
      string\_file.close()  
    return serialized  
  
  def deserialize(self, string\_value):  
    string\_file = io.BytesIO(string\_value)  
    try:  
      self.array = numpy.load(string\_file, allow\_pickle=False)  
    finally:  
      string\_file.close()

Instances of NumpyWrapper are checkpointable objects, and will be saved and restored from checkpoints along with TensorFlow state like variables.

root = tf.train.Checkpoint(numpy=NumpyWrapper(numpy.array([1.])))  
save\_path = root.save(prefix)  
root.numpy.array \*= 2.  
assert [2.] == root.numpy.array  
root.restore(save\_path)  
assert [1.] == root.numpy.array

## Methods

### deserialize

deserialize(string\_value)

Callback to deserialize the object.

### serialize

serialize()

Callback to serialize the object. Returns a string.

# Module: tf.compat.v1.feature\_column

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

Public API for tf.feature\_column namespace.

## Functions

[bucketized\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/bucketized_column): Represents discretized dense input.

[categorical\_column\_with\_hash\_bucket(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_hash_bucket): Represents sparse feature where ids are set by hashing.

[categorical\_column\_with\_identity(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_identity): A CategoricalColumn that returns identity values.

[categorical\_column\_with\_vocabulary\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/feature_column/categorical_column_with_vocabulary_file): A CategoricalColumn with a vocabulary file.

[categorical\_column\_with\_vocabulary\_list(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_vocabulary_list): A CategoricalColumn with in-memory vocabulary.

[crossed\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/crossed_column): Returns a column for performing crosses of categorical features.

[embedding\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/embedding_column): DenseColumn that converts from sparse, categorical input.

[indicator\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/indicator_column): Represents multi-hot representation of given categorical column.

[input\_layer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/feature_column/input_layer): Returns a dense Tensor as input layer based on given feature\_columns.

[linear\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/feature_column/linear_model): Returns a linear prediction Tensor based on given feature\_columns.

[make\_parse\_example\_spec(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/feature_column/make_parse_example_spec): Creates parsing spec dictionary from input feature\_columns.

[numeric\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/numeric_column): Represents real valued or numerical features.

[sequence\_categorical\_column\_with\_hash\_bucket(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/sequence_categorical_column_with_hash_bucket): A sequence of categorical terms where ids are set by hashing.

[sequence\_categorical\_column\_with\_identity(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/sequence_categorical_column_with_identity): Returns a feature column that represents sequences of integers.

[sequence\_categorical\_column\_with\_vocabulary\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/sequence_categorical_column_with_vocabulary_file): A sequence of categorical terms where ids use a vocabulary file.

[sequence\_categorical\_column\_with\_vocabulary\_list(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/sequence_categorical_column_with_vocabulary_list): A sequence of categorical terms where ids use an in-memory list.

[sequence\_numeric\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/sequence_numeric_column): Returns a feature column that represents sequences of numeric data.

[shared\_embedding\_columns(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/feature_column/shared_embedding_columns): List of dense columns that convert from sparse, categorical input.

[weighted\_categorical\_column(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/weighted_categorical_column): Applies weight values to a CategoricalColumn.

# tf.compat.v1.feature\_column.input\_layer

Returns a dense Tensor as input layer based on given feature\_columns.

tf.compat.v1.feature\_column.input\_layer(

    features,

    feature\_columns,

    weight\_collections=None,

    trainable=True,

    cols\_to\_vars=None,

    cols\_to\_output\_tensors=None

)

Defined in [python/feature\_column/feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column.py).

Generally a single example in training data is described with FeatureColumns. At the first layer of the model, this column oriented data should be converted to a single Tensor.

#### Example:

price = numeric\_column('price')

keywords\_embedded = embedding\_column(

    categorical\_column\_with\_hash\_bucket("keywords", 10K), dimensions=16)

columns = [price, keywords\_embedded, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

for units in [128, 64, 32]:

  dense\_tensor = tf.compat.v1.layers.dense(dense\_tensor, units, tf.nn.relu)

prediction = tf.compat.v1.layers.dense(dense\_tensor, 1)

#### Args:

* features: A mapping from key to tensors. \_FeatureColumns look up via these keys. For example numeric\_column('price') will look at 'price' key in this dict. Values can be a SparseTensor or a Tensor depends on corresponding \_FeatureColumn.
* feature\_columns: An iterable containing the FeatureColumns to use as inputs to your model. All items should be instances of classes derived from \_DenseColumn such as numeric\_column, embedding\_column, bucketized\_column, indicator\_column. If you have categorical features, you can wrap them with an embedding\_column or indicator\_column.
* weight\_collections: A list of collection names to which the Variable will be added. Note that variables will also be added to collections tf.GraphKeys.GLOBAL\_VARIABLES and ops.GraphKeys.MODEL\_VARIABLES.
* trainable: If True also add the variable to the graph collectionGraphKeys.TRAINABLE\_VARIABLES (see [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable)).
* cols\_to\_vars: If not None, must be a dictionary that will be filled with a mapping from \_FeatureColumn to list of Variables. For example, after the call, we might have cols\_to\_vars = {\_EmbeddingColumn( categorical\_column=\_HashedCategoricalColumn( key='sparse\_feature', hash\_bucket\_size=5, dtype=tf.string), dimension=10): [<tf.Variable 'some\_variable:0' shape=(5, 10), <tf.Variable 'some\_variable:1' shape=(5, 10)]} If a column creates no variables, its value will be an empty list.
* cols\_to\_output\_tensors: If not None, must be a dictionary that will be filled with a mapping from '\_FeatureColumn' to the associated output Tensors.

#### Returns:

A Tensor which represents input layer of a model. Its shape is (batch\_size, first\_layer\_dimension) and its dtype is float32. first\_layer\_dimension is determined based on given feature\_columns.

#### Raises:

* ValueError: if an item in feature\_columns is not a \_DenseColumn.

# tf.compat.v1.feature\_column.linear\_model

Returns a linear prediction Tensor based on given feature\_columns.

tf.compat.v1.feature\_column.linear\_model(

    features,

    feature\_columns,

    units=1,

    sparse\_combiner='sum',

    weight\_collections=None,

    trainable=True,

    cols\_to\_vars=None

)

Defined in [python/feature\_column/feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column.py).

This function generates a weighted sum based on output dimension units. Weighted sum refers to logits in classification problems. It refers to the prediction itself for linear regression problems.

Note on supported columns: linear\_model treats categorical columns as indicator\_columns. To be specific, assume the input as SparseTensor looks like:

  shape = [2, 2]

  {

      [0, 0]: "a"

      [1, 0]: "b"

      [1, 1]: "c"

  }

linear\_model assigns weights for the presence of "a", "b", "c' implicitly, just like indicator\_column, while input\_layer explicitly requires wrapping each of categorical columns with an embedding\_column or an indicator\_column.

#### Example of usage:

price = numeric\_column('price')

price\_buckets = bucketized\_column(price, boundaries=[0., 10., 100., 1000.])

keywords = categorical\_column\_with\_hash\_bucket("keywords", 10K)

keywords\_price = crossed\_column('keywords', price\_buckets, ...)

columns = [price\_buckets, keywords, keywords\_price ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

prediction = linear\_model(features, columns)

The sparse\_combiner argument works as follows For example, for two features represented as the categorical columns:

  # Feature 1

  shape = [2, 2]

  {

      [0, 0]: "a"

      [0, 1]: "b"

      [1, 0]: "c"

  }

  # Feature 2

  shape = [2, 3]

  {

      [0, 0]: "d"

      [1, 0]: "e"

      [1, 1]: "f"

      [1, 2]: "f"

  }

with sparse\_combiner as "mean", the linear model outputs consequently are:

  y\_0 = 1.0 / 2.0 \* ( w\_a + w\_b ) + w\_d + b

  y\_1 = w\_c + 1.0 / 3.0 \* ( w\_e + 2.0 \* w\_f ) + b

where y\_i is the output, b is the bias, and w\_x is the weight assigned to the presence of x in the input features.

#### Args:

* features: A mapping from key to tensors. \_FeatureColumns look up via these keys. For example numeric\_column('price') will look at 'price' key in this dict. Values are Tensor or SparseTensor depending on corresponding \_FeatureColumn.
* feature\_columns: An iterable containing the FeatureColumns to use as inputs to your model. All items should be instances of classes derived from \_FeatureColumns.
* units: An integer, dimensionality of the output space. Default value is 1.
* sparse\_combiner: A string specifying how to reduce if a categorical column is multivalent. Except numeric\_column, almost all columns passed to linear\_model are considered as categorical columns. It combines each categorical column independently. Currently "mean", "sqrtn" and "sum" are supported, with "sum" the default for linear model. "sqrtn" often achieves good accuracy, in particular with bag-of-words columns.
  + "sum": do not normalize features in the column
  + "mean": do l1 normalization on features in the column
  + "sqrtn": do l2 normalization on features in the column
* weight\_collections: A list of collection names to which the Variable will be added. Note that, variables will also be added to collections tf.GraphKeys.GLOBAL\_VARIABLES and ops.GraphKeys.MODEL\_VARIABLES.
* trainable: If True also add the variable to the graph collectionGraphKeys.TRAINABLE\_VARIABLES (see [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable)).
* cols\_to\_vars: If not None, must be a dictionary that will be filled with a mapping from \_FeatureColumn to associated list of Variables. For example, after the call, we might have cols\_to\_vars = { \_NumericColumn( key='numeric\_feature1', shape=(1,): [], 'bias': [], \_NumericColumn( key='numeric\_feature2', shape=(2,)): []} If a column creates no variables, its value will be an empty list. Note that cols\_to\_vars will also contain a string key 'bias' that maps to a list of Variables.

#### Returns:

A Tensor which represents predictions/logits of a linear model. Its shape is (batch\_size, units) and its dtype is float32.

#### Raises:

* ValueError: if an item in feature\_columns is neither a \_DenseColumn nor \_CategoricalColumn.

# tf.compat.v1.feature\_column.shared\_embedding\_columns

List of dense columns that convert from sparse, categorical input.

tf.compat.v1.feature\_column.shared\_embedding\_columns(

    categorical\_columns,

    dimension,

    combiner='mean',

    initializer=None,

    shared\_embedding\_collection\_name=None,

    ckpt\_to\_load\_from=None,

    tensor\_name\_in\_ckpt=None,

    max\_norm=None,

    trainable=True

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

This is similar to embedding\_column, except that it produces a list of embedding columns that share the same embedding weights.

Use this when your inputs are sparse and of the same type (e.g. watched and impression video IDs that share the same vocabulary), and you want to convert them to a dense representation (e.g., to feed to a DNN).

Inputs must be a list of categorical columns created by any of the categorical\_column\_\* function. They must all be of the same type and have the same arguments except key. E.g. they can be categorical\_column\_with\_vocabulary\_file with the same vocabulary\_file. Some or all columns could also be weighted\_categorical\_column.

Here is an example embedding of two features for a DNNClassifier model:

watched\_video\_id = categorical\_column\_with\_vocabulary\_file(

    'watched\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

impression\_video\_id = categorical\_column\_with\_vocabulary\_file(

    'impression\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

columns = shared\_embedding\_columns(

    [watched\_video\_id, impression\_video\_id], dimension=10)

estimator = tf.estimator.DNNClassifier(feature\_columns=columns, ...)

label\_column = ...

def input\_fn():

  features = tf.io.parse\_example(

      ..., features=make\_parse\_example\_spec(columns + [label\_column]))

  labels = features.pop(label\_column.name)

  return features, labels

estimator.train(input\_fn=input\_fn, steps=100)

Here is an example using shared\_embedding\_columns with model\_fn:

def model\_fn(features, ...):

  watched\_video\_id = categorical\_column\_with\_vocabulary\_file(

      'watched\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

  impression\_video\_id = categorical\_column\_with\_vocabulary\_file(

      'impression\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

  columns = shared\_embedding\_columns(

      [watched\_video\_id, impression\_video\_id], dimension=10)

  dense\_tensor = input\_layer(features, columns)

  # Form DNN layers, calculate loss, and return EstimatorSpec.

  ...

#### Args:

* categorical\_columns: List of categorical columns created by acategorical\_column\_with\_\* function. These columns produce the sparse IDs that are inputs to the embedding lookup. All columns must be of the same type and have the same arguments except key. E.g. they can be categorical\_column\_with\_vocabulary\_file with the same vocabulary\_file. Some or all columns could also be weighted\_categorical\_column.
* dimension: An integer specifying dimension of the embedding, must be > 0.
* combiner: A string specifying how to reduce if there are multiple entries in a single row. Currently 'mean', 'sqrtn' and 'sum' are supported, with 'mean' the default. 'sqrtn' often achieves good accuracy, in particular with bag-of-words columns. Each of this can be thought as example level normalizations on the column. For more information, see tf.embedding\_lookup\_sparse.
* initializer: A variable initializer function to be used in embedding variable initialization. If not specified, defaults to [tf.compat.v1.truncated\_normal\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/truncated_normal_initializer) with mean 0.0 and standard deviation 1/sqrt(dimension).
* shared\_embedding\_collection\_name: Optional name of the collection where shared embedding weights are added. If not given, a reasonable name will be chosen based on the names of categorical\_columns. This is also used in variable\_scope when creating shared embedding weights.
* ckpt\_to\_load\_from: String representing checkpoint name/pattern from which to restore column weights. Required if tensor\_name\_in\_ckpt is not None.
* tensor\_name\_in\_ckpt: Name of the Tensor in ckpt\_to\_load\_from from which to restore the column weights. Required if ckpt\_to\_load\_from is not None.
* max\_norm: If not None, each embedding is clipped if its l2-norm is larger than this value, before combining.
* trainable: Whether or not the embedding is trainable. Default is True.

#### Returns:

A list of dense columns that converts from sparse input. The order of results follows the ordering of categorical\_columns.

#### Raises:

* ValueError: if dimension not > 0.
* ValueError: if any of the given categorical\_columns is of different type or has different arguments than the others.
* ValueError: if exactly one of ckpt\_to\_load\_from and tensor\_name\_in\_ckpt is specified.
* ValueError: if initializer is specified and is not callable.
* RuntimeError: if eager execution is enabled.

# tf.feature\_column.bucketized\_column

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/bucketized_column#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/bucketized_column#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/bucketized_column#used_in_the_tutorials)

Represents discretized dense input.

### Aliases:

* tf.compat.v1.feature\_column.bucketized\_column
* tf.compat.v2.feature\_column.bucketized\_column
* tf.feature\_column.bucketized\_column

tf.feature\_column.bucketized\_column(

    source\_column,

    boundaries

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Buckets include the left boundary, and exclude the right boundary. Namely, boundaries=[0., 1., 2.] generates buckets (-inf, 0.), [0., 1.), [1., 2.), and [2., +inf).

For example, if the inputs are

boundaries = [0, 10, 100]

input tensor = [[-5, 10000]

                [150,   10]

                [5,    100]]

then the output will be

output = [[0, 3]

          [3, 2]

          [1, 3]]

#### Example:

price = numeric\_column('price')

bucketized\_price = bucketized\_column(price, boundaries=[...])

columns = [bucketized\_price, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

# or

columns = [bucketized\_price, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

bucketized\_column can also be crossed with another categorical column using crossed\_column:

price = numeric\_column('price')

# bucketized\_column converts numerical feature to a categorical one.

bucketized\_price = bucketized\_column(price, boundaries=[...])

# 'keywords' is a string feature.

price\_x\_keywords = crossed\_column([bucketized\_price, 'keywords'], 50K)

columns = [price\_x\_keywords, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

#### Args:

* source\_column: A one-dimensional dense column which is generated with numeric\_column.
* boundaries: A sorted list or tuple of floats specifying the boundaries.

#### Returns:

A BucketizedColumn.

#### Raises:

* ValueError: If source\_column is not a numeric column, or if it is not one-dimensional.
* ValueError: If boundaries is not a sorted list or tuple.

# tf.feature\_column.categorical\_column\_with\_hash\_bucket

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_hash_bucket#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_hash_bucket#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_hash_bucket#used_in_the_tutorials)

Represents sparse feature where ids are set by hashing.

### Aliases:

* tf.compat.v1.feature\_column.categorical\_column\_with\_hash\_bucket
* tf.compat.v2.feature\_column.categorical\_column\_with\_hash\_bucket
* tf.feature\_column.categorical\_column\_with\_hash\_bucket

tf.feature\_column.categorical\_column\_with\_hash\_bucket(

    key,

    hash\_bucket\_size,

    dtype=tf.dtypes.string

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Use this when your sparse features are in string or integer format, and you want to distribute your inputs into a finite number of buckets by hashing. output\_id = Hash(input\_feature\_string) % bucket\_size for string type input. For int type input, the value is converted to its string representation first and then hashed by the same formula.

For input dictionary features, features[key] is either Tensor or SparseTensor. If Tensor, missing values can be represented by -1 for int and '' for string, which will be dropped by this feature column.

#### Example:

keywords = categorical\_column\_with\_hash\_bucket("keywords", 10K)

columns = [keywords, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

# or

keywords\_embedded = embedding\_column(keywords, 16)

columns = [keywords\_embedded, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

#### Args:

* key: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature Tensor objects, and feature columns.
* hash\_bucket\_size: An int > 1. The number of buckets.
* dtype: The type of features. Only string and integer types are supported.

#### Returns:

A HashedCategoricalColumn.

#### Raises:

* ValueError: hash\_bucket\_size is not greater than 1.
* ValueError: dtype is neither string nor integer.

# tf.feature\_column.categorical\_column\_with\_identity

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

A CategoricalColumn that returns identity values.

### Aliases:

* tf.compat.v1.feature\_column.categorical\_column\_with\_identity
* tf.compat.v2.feature\_column.categorical\_column\_with\_identity
* tf.feature\_column.categorical\_column\_with\_identity

tf.feature\_column.categorical\_column\_with\_identity(

    key,

    num\_buckets,

    default\_value=None

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

Use this when your inputs are integers in the range [0, num\_buckets), and you want to use the input value itself as the categorical ID. Values outside this range will result in default\_value if specified, otherwise it will fail.

Typically, this is used for contiguous ranges of integer indexes, but it doesn't have to be. This might be inefficient, however, if many of IDs are unused. Consider categorical\_column\_with\_hash\_bucket in that case.

For input dictionary features, features[key] is either Tensor or SparseTensor. If Tensor, missing values can be represented by -1 for int and '' for string, which will be dropped by this feature column.

In the following examples, each input in the range [0, 1000000) is assigned the same value. All other inputs are assigned default\_value 0. Note that a literal 0 in inputs will result in the same default ID.

#### Linear model:

video\_id = categorical\_column\_with\_identity(

    key='video\_id', num\_buckets=1000000, default\_value=0)

columns = [video\_id, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction, \_, \_ = linear\_model(features, columns)

Embedding for a DNN model:

columns = [embedding\_column(video\_id, 9),...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

#### Args:

* key: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature Tensor objects, and feature columns.
* num\_buckets: Range of inputs and outputs is [0, num\_buckets).
* default\_value: If None, this column's graph operations will fail for out-of-range inputs. Otherwise, this value must be in the range [0, num\_buckets), and will replace inputs in that range.

#### Returns:

A CategoricalColumn that returns identity values.

#### Raises:

* ValueError: if num\_buckets is less than one.
* ValueError: if default\_value is not in range [0, num\_buckets).

# tf.feature\_column.categorical\_column\_with\_vocabulary\_file

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

A CategoricalColumn with a vocabulary file.

### Aliases:

* tf.compat.v2.feature\_column.categorical\_column\_with\_vocabulary\_file
* tf.feature\_column.categorical\_column\_with\_vocabulary\_file

tf.feature\_column.categorical\_column\_with\_vocabulary\_file(

    key,

    vocabulary\_file,

    vocabulary\_size=None,

    dtype=tf.dtypes.string,

    default\_value=None,

    num\_oov\_buckets=0

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

Use this when your inputs are in string or integer format, and you have a vocabulary file that maps each value to an integer ID. By default, out-of-vocabulary values are ignored. Use either (but not both) ofnum\_oov\_buckets and default\_value to specify how to include out-of-vocabulary values.

For input dictionary features, features[key] is either Tensor or SparseTensor. If Tensor, missing values can be represented by -1 for int and '' for string, which will be dropped by this feature column.

Example with num\_oov\_buckets: File '/us/states.txt' contains 50 lines, each with a 2-character U.S. state abbreviation. All inputs with values in that file are assigned an ID 0-49, corresponding to its line number. All other values are hashed and assigned an ID 50-54.

states = categorical\_column\_with\_vocabulary\_file(

    key='states', vocabulary\_file='/us/states.txt', vocabulary\_size=50,

    num\_oov\_buckets=5)

columns = [states, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

Example with default\_value: File '/us/states.txt' contains 51 lines - the first line is 'XX', and the other 50 each have a 2-character U.S. state abbreviation. Both a literal 'XX' in input, and other values missing from the file, will be assigned ID 0. All others are assigned the corresponding line number 1-50.

states = categorical\_column\_with\_vocabulary\_file(

    key='states', vocabulary\_file='/us/states.txt', vocabulary\_size=51,

    default\_value=0)

columns = [states, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction, \_, \_ = linear\_model(features, columns)

And to make an embedding with either:

columns = [embedding\_column(states, 3),...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

#### Args:

* key: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature Tensor objects, and feature columns.
* vocabulary\_file: The vocabulary file name.
* vocabulary\_size: Number of the elements in the vocabulary. This must be no greater than length of vocabulary\_file, if less than length, later values are ignored. If None, it is set to the length of vocabulary\_file.
* dtype: The type of features. Only string and integer types are supported.
* default\_value: The integer ID value to return for out-of-vocabulary feature values, defaults to -1. This can not be specified with a positive num\_oov\_buckets.
* num\_oov\_buckets: Non-negative integer, the number of out-of-vocabulary buckets. All out-of-vocabulary inputs will be assigned IDs in the range [vocabulary\_size, vocabulary\_size+num\_oov\_buckets) based on a hash of the input value. A positive num\_oov\_buckets can not be specified with default\_value.

#### Returns:

A CategoricalColumn with a vocabulary file.

#### Raises:

* ValueError: vocabulary\_file is missing or cannot be opened.
* ValueError: vocabulary\_size is missing or < 1.
* ValueError: num\_oov\_buckets is a negative integer.
* ValueError: num\_oov\_buckets and default\_value are both specified.
* ValueError: dtype is neither string nor integer.

# tf.feature\_column.categorical\_column\_with\_vocabulary\_list

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_vocabulary_list#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_vocabulary_list#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/categorical_column_with_vocabulary_list#used_in_the_tutorials)

A CategoricalColumn with in-memory vocabulary.

### Aliases:

* tf.compat.v1.feature\_column.categorical\_column\_with\_vocabulary\_list
* tf.compat.v2.feature\_column.categorical\_column\_with\_vocabulary\_list
* tf.feature\_column.categorical\_column\_with\_vocabulary\_list

tf.feature\_column.categorical\_column\_with\_vocabulary\_list(

    key,

    vocabulary\_list,

    dtype=None,

    default\_value=-1,

    num\_oov\_buckets=0

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Build a linear model with Estimators](https://www.tensorflow.org/beta/tutorials/estimators/linear)
* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Use this when your inputs are in string or integer format, and you have an in-memory vocabulary mapping each value to an integer ID. By default, out-of-vocabulary values are ignored. Use either (but not both) of num\_oov\_buckets and default\_value to specify how to include out-of-vocabulary values.

For input dictionary features, features[key] is either Tensor or SparseTensor. If Tensor, missing values can be represented by -1 for int and '' for string, which will be dropped by this feature column.

Example with num\_oov\_buckets: In the following example, each input in vocabulary\_list is assigned an ID 0-3 corresponding to its index (e.g., input 'B' produces output 2). All other inputs are hashed and assigned an ID 4-5.

colors = categorical\_column\_with\_vocabulary\_list(

    key='colors', vocabulary\_list=('R', 'G', 'B', 'Y'),

    num\_oov\_buckets=2)

columns = [colors, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction, \_, \_ = linear\_model(features, columns)

Example with default\_value: In the following example, each input in vocabulary\_list is assigned an ID 0-4 corresponding to its index (e.g., input 'B' produces output 3). All other inputs are assigned default\_value 0.

colors = categorical\_column\_with\_vocabulary\_list(

    key='colors', vocabulary\_list=('X', 'R', 'G', 'B', 'Y'), default\_value=0)

columns = [colors, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction, \_, \_ = linear\_model(features, columns)

And to make an embedding with either:

columns = [embedding\_column(colors, 3),...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

#### Args:

* key: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature Tensor objects, and feature columns.
* vocabulary\_list: An ordered iterable defining the vocabulary. Each feature is mapped to the index of its value (if present) in vocabulary\_list. Must be castable to dtype.
* dtype: The type of features. Only string and integer types are supported. If None, it will be inferred from vocabulary\_list.
* default\_value: The integer ID value to return for out-of-vocabulary feature values, defaults to -1. This can not be specified with a positive num\_oov\_buckets.
* num\_oov\_buckets: Non-negative integer, the number of out-of-vocabulary buckets. All out-of-vocabulary inputs will be assigned IDs in the range [len(vocabulary\_list), len(vocabulary\_list)+num\_oov\_buckets) based on a hash of the input value. A positive num\_oov\_buckets can not be specified with default\_value.

#### Returns:

A CategoricalColumn with in-memory vocabulary.

#### Raises:

* ValueError: if vocabulary\_list is empty, or contains duplicate keys.
* ValueError: num\_oov\_buckets is a negative integer.
* ValueError: num\_oov\_buckets and default\_value are both specified.
* ValueError: if dtype is not integer or string.

# tf.feature\_column.crossed\_column

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/crossed_column#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/crossed_column#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/crossed_column#used_in_the_tutorials)

Returns a column for performing crosses of categorical features.

### Aliases:

* tf.compat.v1.feature\_column.crossed\_column
* tf.compat.v2.feature\_column.crossed\_column
* tf.feature\_column.crossed\_column

tf.feature\_column.crossed\_column(

    keys,

    hash\_bucket\_size,

    hash\_key=None

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Build a linear model with Estimators](https://www.tensorflow.org/beta/tutorials/estimators/linear)
* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Crossed features will be hashed according to hash\_bucket\_size. Conceptually, the transformation can be thought of as: Hash(cartesian product of features) % hash\_bucket\_size

For example, if the input features are:

* SparseTensor referred by first key:

shape = [2, 2]

{

    [0, 0]: "a"

    [1, 0]: "b"

    [1, 1]: "c"

}

* SparseTensor referred by second key:

shape = [2, 1]

{

    [0, 0]: "d"

    [1, 0]: "e"

}

then crossed feature will look like:

 shape = [2, 2]

{

    [0, 0]: Hash64("d", Hash64("a")) % hash\_bucket\_size

    [1, 0]: Hash64("e", Hash64("b")) % hash\_bucket\_size

    [1, 1]: Hash64("e", Hash64("c")) % hash\_bucket\_size

}

Here is an example to create a linear model with crosses of string features:

keywords\_x\_doc\_terms = crossed\_column(['keywords', 'doc\_terms'], 50K)

columns = [keywords\_x\_doc\_terms, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

You could also use vocabulary lookup before crossing:

keywords = categorical\_column\_with\_vocabulary\_file(

    'keywords', '/path/to/vocabulary/file', vocabulary\_size=1K)

keywords\_x\_doc\_terms = crossed\_column([keywords, 'doc\_terms'], 50K)

columns = [keywords\_x\_doc\_terms, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

If an input feature is of numeric type, you can use categorical\_column\_with\_identity, or bucketized\_column, as in the example:

# vertical\_id is an integer categorical feature.

vertical\_id = categorical\_column\_with\_identity('vertical\_id', 10K)

price = numeric\_column('price')

# bucketized\_column converts numerical feature to a categorical one.

bucketized\_price = bucketized\_column(price, boundaries=[...])

vertical\_id\_x\_price = crossed\_column([vertical\_id, bucketized\_price], 50K)

columns = [vertical\_id\_x\_price, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

To use crossed column in DNN model, you need to add it in an embedding column as in this example:

vertical\_id\_x\_price = crossed\_column([vertical\_id, bucketized\_price], 50K)

vertical\_id\_x\_price\_embedded = embedding\_column(vertical\_id\_x\_price, 10)

dense\_tensor = input\_layer(features, [vertical\_id\_x\_price\_embedded, ...])

#### Args:

* keys: An iterable identifying the features to be crossed. Each element can be either:
  + string: Will use the corresponding feature which must be of string type.
  + CategoricalColumn: Will use the transformed tensor produced by this column. Does not support hashed categorical column.
* hash\_bucket\_size: An int > 1. The number of buckets.
* hash\_key: Specify the hash\_key that will be used by the FingerprintCat64 function to combine the crosses fingerprints on SparseCrossOp (optional).

#### Returns:

A CrossedColumn.

#### Raises:

* ValueError: If len(keys) < 2.
* ValueError: If any of the keys is neither a string nor CategoricalColumn.
* ValueError: If any of the keys is HashedCategoricalColumn.
* ValueError: If hash\_bucket\_size < 1.

# tf.feature\_column.embedding\_column

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/embedding_column#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/embedding_column#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/embedding_column#used_in_the_tutorials)

DenseColumn that converts from sparse, categorical input.

### Aliases:

* tf.compat.v1.feature\_column.embedding\_column
* tf.compat.v2.feature\_column.embedding\_column
* tf.feature\_column.embedding\_column

tf.feature\_column.embedding\_column(

    categorical\_column,

    dimension,

    combiner='mean',

    initializer=None,

    ckpt\_to\_load\_from=None,

    tensor\_name\_in\_ckpt=None,

    max\_norm=None,

    trainable=True

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)

Use this when your inputs are sparse, but you want to convert them to a dense representation (e.g., to feed to a DNN).

Inputs must be a CategoricalColumn created by any of the categorical\_column\_\* function. Here is an example of using embedding\_column with DNNClassifier:

video\_id = categorical\_column\_with\_identity(

    key='video\_id', num\_buckets=1000000, default\_value=0)

columns = [embedding\_column(video\_id, 9),...]

estimator = tf.estimator.DNNClassifier(feature\_columns=columns, ...)

label\_column = ...

def input\_fn():

  features = tf.io.parse\_example(

      ..., features=make\_parse\_example\_spec(columns + [label\_column]))

  labels = features.pop(label\_column.name)

  return features, labels

estimator.train(input\_fn=input\_fn, steps=100)

Here is an example using embedding\_column with model\_fn:

def model\_fn(features, ...):

  video\_id = categorical\_column\_with\_identity(

      key='video\_id', num\_buckets=1000000, default\_value=0)

  columns = [embedding\_column(video\_id, 9),...]

  dense\_tensor = input\_layer(features, columns)

  # Form DNN layers, calculate loss, and return EstimatorSpec.

  ...

#### Args:

* categorical\_column: A CategoricalColumn created by a categorical\_column\_with\_\*function. This column produces the sparse IDs that are inputs to the embedding lookup.
* dimension: An integer specifying dimension of the embedding, must be > 0.
* combiner: A string specifying how to reduce if there are multiple entries in a single row. Currently 'mean', 'sqrtn' and 'sum' are supported, with 'mean' the default. 'sqrtn' often achieves good accuracy, in particular with bag-of-words columns. Each of this can be thought as example level normalizations on the column. For more information, see tf.embedding\_lookup\_sparse.
* initializer: A variable initializer function to be used in embedding variable initialization. If not specified, defaults to [tf.compat.v1.truncated\_normal\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/truncated_normal_initializer) with mean 0.0 and standard deviation 1/sqrt(dimension).
* ckpt\_to\_load\_from: String representing checkpoint name/pattern from which to restore column weights. Required if tensor\_name\_in\_ckpt is not None.
* tensor\_name\_in\_ckpt: Name of the Tensor in ckpt\_to\_load\_from from which to restore the column weights. Required if ckpt\_to\_load\_from is not None.
* max\_norm: If not None, embedding values are l2-normalized to this value.
* trainable: Whether or not the embedding is trainable. Default is True.

#### Returns:

DenseColumn that converts from sparse input.

#### Raises:

* ValueError: if dimension not > 0.
* ValueError: if exactly one of ckpt\_to\_load\_from and tensor\_name\_in\_ckpt is specified.
* ValueError: if initializer is specified and is not callable.
* RuntimeError: If eager execution is enabled.

# tf.feature\_column.indicator\_column

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/indicator_column#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/indicator_column#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/indicator_column#used_in_the_tutorials)

Represents multi-hot representation of given categorical column.

### Aliases:

* tf.compat.v1.feature\_column.indicator\_column
* tf.compat.v2.feature\_column.indicator\_column
* tf.feature\_column.indicator\_column

tf.feature\_column.indicator\_column(categorical\_column)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the tutorials:

* [Build a linear model with Estimators](https://www.tensorflow.org/beta/tutorials/estimators/linear)
* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)
* For DNN model, indicator\_column can be used to wrap any categorical\_column\_\* (e.g., to feed to DNN). Consider to Use embedding\_column if the number of buckets/unique(values) are large.
* For Wide (aka linear) model, indicator\_column is the internal representation for categorical column when passing categorical column directly (as any element in feature\_columns) to linear\_model. See linear\_model for details.

name = indicator\_column(categorical\_column\_with\_vocabulary\_list(

    'name', ['bob', 'george', 'wanda'])

columns = [name, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

dense\_tensor == [[1, 0, 0]]  # If "name" bytes\_list is ["bob"]

dense\_tensor == [[1, 0, 1]]  # If "name" bytes\_list is ["bob", "wanda"]

dense\_tensor == [[2, 0, 0]]  # If "name" bytes\_list is ["bob", "bob"]

#### Args:

* categorical\_column: A CategoricalColumn which is created bycategorical\_column\_with\_\* or crossed\_column functions.

#### Returns:

An IndicatorColumn.

# tf.feature\_column.make\_parse\_example\_spec

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/make_parse_example_spec#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/make_parse_example_spec#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/make_parse_example_spec#used_in_the_guide)

Creates parsing spec dictionary from input feature\_columns.

### Aliases:

* tf.compat.v2.feature\_column.make\_parse\_example\_spec
* tf.feature\_column.make\_parse\_example\_spec

tf.feature\_column.make\_parse\_example\_spec(feature\_columns)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

The returned dictionary can be used as arg 'features' in [tf.io.parse\_example](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_example).

#### Typical usage example:

# Define features and transformations

feature\_a = categorical\_column\_with\_vocabulary\_file(...)

feature\_b = numeric\_column(...)

feature\_c\_bucketized = bucketized\_column(numeric\_column("feature\_c"), ...)

feature\_a\_x\_feature\_c = crossed\_column(

    columns=["feature\_a", feature\_c\_bucketized], ...)

feature\_columns = set(

    [feature\_b, feature\_c\_bucketized, feature\_a\_x\_feature\_c])

features = tf.io.parse\_example(

    serialized=serialized\_examples,

    features=make\_parse\_example\_spec(feature\_columns))

For the above example, make\_parse\_example\_spec would return the dict:

{

    "feature\_a": parsing\_ops.VarLenFeature(tf.string),

    "feature\_b": parsing\_ops.FixedLenFeature([1], dtype=tf.float32),

    "feature\_c": parsing\_ops.FixedLenFeature([1], dtype=tf.float32)

}

#### Args:

* feature\_columns: An iterable containing all feature columns. All items should be instances of classes derived from FeatureColumn.

#### Returns:

A dict mapping each feature key to a FixedLenFeature or VarLenFeature value.

#### Raises:

* ValueError: If any of the given feature\_columns is not a FeatureColumn instance.

# tf.feature\_column.numeric\_column

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/numeric_column#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/numeric_column#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/numeric_column#used_in_the_guide)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/feature_column/numeric_column#used_in_the_tutorials)

Represents real valued or numerical features.

### Aliases:

* tf.compat.v1.feature\_column.numeric\_column
* tf.compat.v2.feature\_column.numeric\_column
* tf.feature\_column.numeric\_column

tf.feature\_column.numeric\_column(

    key,

    shape=(1,),

    default\_value=None,

    dtype=tf.dtypes.float32,

    normalizer\_fn=None

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

### Used in the guide:

* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

### Used in the tutorials:

* [Build a linear model with Estimators](https://www.tensorflow.org/beta/tutorials/estimators/linear)
* [Classify structured data](https://www.tensorflow.org/beta/tutorials/keras/feature_columns)
* [Premade Estimators](https://www.tensorflow.org/beta/tutorials/estimators/premade_estimators)

#### Example:

price = numeric\_column('price')

columns = [price, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

dense\_tensor = input\_layer(features, columns)

# or

bucketized\_price = bucketized\_column(price, boundaries=[...])

columns = [bucketized\_price, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction = linear\_model(features, columns)

#### Args:

* key: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature Tensor objects, and feature columns.
* shape: An iterable of integers specifies the shape of the Tensor. An integer can be given which means a single dimension Tensor with given width. The Tensor representing the column will have the shape of [batch\_size] + shape.
* default\_value: A single value compatible with dtype or an iterable of values compatible with dtype which the column takes on during tf.Example parsing if data is missing. A default value of None will cause [tf.io.parse\_example](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_example) to fail if an example does not contain this column. If a single value is provided, the same value will be applied as the default value for every item. If an iterable of values is provided, the shape of the default\_value should be equal to the given shape.
* dtype: defines the type of values. Default value is [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32). Must be a non-quantized, real integer or floating point type.
* normalizer\_fn: If not None, a function that can be used to normalize the value of the tensor after default\_value is applied for parsing. Normalizer function takes the input Tensor as its argument, and returns the output Tensor. (e.g. lambda x: (x - 3.0) / 4.2). Please note that even though the most common use case of this function is normalization, it can be used for any kind of Tensorflow transformations.

#### Returns:

A NumericColumn.

#### Raises:

* TypeError: if any dimension in shape is not an int
* ValueError: if any dimension in shape is not a positive integer
* TypeError: if default\_value is an iterable but not compatible with shape
* TypeError: if default\_value is not compatible with dtype.
* ValueError: if dtype is not convertible to [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32).

# tf.feature\_column.sequence\_categorical\_column\_with\_hash\_bucket

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

A sequence of categorical terms where ids are set by hashing.

### Aliases:

* tf.compat.v1.feature\_column.sequence\_categorical\_column\_with\_hash\_bucket
* tf.compat.v2.feature\_column.sequence\_categorical\_column\_with\_hash\_bucket
* tf.feature\_column.sequence\_categorical\_column\_with\_hash\_bucket

tf.feature\_column.sequence\_categorical\_column\_with\_hash\_bucket(

    key,

    hash\_bucket\_size,

    dtype=tf.dtypes.string

)

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

Pass this to embedding\_column or indicator\_column to convert sequence categorical data into dense representation for input to sequence NN, such as RNN.

#### Example:

tokens = sequence\_categorical\_column\_with\_hash\_bucket(

    'tokens', hash\_bucket\_size=1000)

tokens\_embedding = embedding\_column(tokens, dimension=10)

columns = [tokens\_embedding]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

sequence\_feature\_layer = SequenceFeatures(columns)

sequence\_input, sequence\_length = sequence\_feature\_layer(features)

sequence\_length\_mask = tf.sequence\_mask(sequence\_length)

rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)

rnn\_layer = tf.keras.layers.RNN(rnn\_cell)

outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

#### Args:

* key: A unique string identifying the input feature.
* hash\_bucket\_size: An int > 1. The number of buckets.
* dtype: The type of features. Only string and integer types are supported.

#### Returns:

A SequenceCategoricalColumn.

#### Raises:

* ValueError: hash\_bucket\_size is not greater than 1.
* ValueError: dtype is neither string nor integer.

# tf.feature\_column.sequence\_categorical\_column\_with\_identity

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

Returns a feature column that represents sequences of integers.

### Aliases:

* tf.compat.v1.feature\_column.sequence\_categorical\_column\_with\_identity
* tf.compat.v2.feature\_column.sequence\_categorical\_column\_with\_identity
* tf.feature\_column.sequence\_categorical\_column\_with\_identity

tf.feature\_column.sequence\_categorical\_column\_with\_identity(

    key,

    num\_buckets,

    default\_value=None

)

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

Pass this to embedding\_column or indicator\_column to convert sequence categorical data into dense representation for input to sequence NN, such as RNN.

#### Example:

watches = sequence\_categorical\_column\_with\_identity(

    'watches', num\_buckets=1000)

watches\_embedding = embedding\_column(watches, dimension=10)

columns = [watches\_embedding]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

sequence\_feature\_layer = SequenceFeatures(columns)

sequence\_input, sequence\_length = sequence\_feature\_layer(features)

sequence\_length\_mask = tf.sequence\_mask(sequence\_length)

rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)

rnn\_layer = tf.keras.layers.RNN(rnn\_cell)

outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

#### Args:

* key: A unique string identifying the input feature.
* num\_buckets: Range of inputs. Namely, inputs are expected to be in the range [0, num\_buckets).
* default\_value: If None, this column's graph operations will fail for out-of-range inputs. Otherwise, this value must be in the range [0, num\_buckets), and will replace out-of-range inputs.

#### Returns:

A SequenceCategoricalColumn.

#### Raises:

* ValueError: if num\_buckets is less than one.
* ValueError: if default\_value is not in range [0, num\_buckets).

# tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_file

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

A sequence of categorical terms where ids use a vocabulary file.

### Aliases:

* tf.compat.v1.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_file
* tf.compat.v2.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_file
* tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_file

tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_file(

    key,

    vocabulary\_file,

    vocabulary\_size=None,

    num\_oov\_buckets=0,

    default\_value=None,

    dtype=tf.dtypes.string

)

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

Pass this to embedding\_column or indicator\_column to convert sequence categorical data into dense representation for input to sequence NN, such as RNN.

#### Example:

states = sequence\_categorical\_column\_with\_vocabulary\_file(

    key='states', vocabulary\_file='/us/states.txt', vocabulary\_size=50,

    num\_oov\_buckets=5)

states\_embedding = embedding\_column(states, dimension=10)

columns = [states\_embedding]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

sequence\_feature\_layer = SequenceFeatures(columns)

sequence\_input, sequence\_length = sequence\_feature\_layer(features)

sequence\_length\_mask = tf.sequence\_mask(sequence\_length)

rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)

rnn\_layer = tf.keras.layers.RNN(rnn\_cell)

outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

#### Args:

* key: A unique string identifying the input feature.
* vocabulary\_file: The vocabulary file name.
* vocabulary\_size: Number of the elements in the vocabulary. This must be no greater than length of vocabulary\_file, if less than length, later values are ignored. If None, it is set to the length of vocabulary\_file.
* num\_oov\_buckets: Non-negative integer, the number of out-of-vocabulary buckets. All out-of-vocabulary inputs will be assigned IDs in the range [vocabulary\_size, vocabulary\_size+num\_oov\_buckets) based on a hash of the input value. A positive num\_oov\_buckets can not be specified with default\_value.
* default\_value: The integer ID value to return for out-of-vocabulary feature values, defaults to -1. This can not be specified with a positive num\_oov\_buckets.
* dtype: The type of features. Only string and integer types are supported.

#### Returns:

A SequenceCategoricalColumn.

#### Raises:

* ValueError: vocabulary\_file is missing or cannot be opened.
* ValueError: vocabulary\_size is missing or < 1.
* ValueError: num\_oov\_buckets is a negative integer.
* ValueError: num\_oov\_buckets and default\_value are both specified.
* ValueError: dtype is neither string nor integer.

# tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_list

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

A sequence of categorical terms where ids use an in-memory list.

### Aliases:

* tf.compat.v1.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_list
* tf.compat.v2.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_list
* tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_list

tf.feature\_column.sequence\_categorical\_column\_with\_vocabulary\_list(

    key,

    vocabulary\_list,

    dtype=None,

    default\_value=-1,

    num\_oov\_buckets=0

)

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

Pass this to embedding\_column or indicator\_column to convert sequence categorical data into dense representation for input to sequence NN, such as RNN.

#### Example:

colors = sequence\_categorical\_column\_with\_vocabulary\_list(

    key='colors', vocabulary\_list=('R', 'G', 'B', 'Y'),

    num\_oov\_buckets=2)

colors\_embedding = embedding\_column(colors, dimension=3)

columns = [colors\_embedding]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

sequence\_feature\_layer = SequenceFeatures(columns)

sequence\_input, sequence\_length = sequence\_feature\_layer(features)

sequence\_length\_mask = tf.sequence\_mask(sequence\_length)

rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)

rnn\_layer = tf.keras.layers.RNN(rnn\_cell)

outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

#### Args:

* key: A unique string identifying the input feature.
* vocabulary\_list: An ordered iterable defining the vocabulary. Each feature is mapped to the index of its value (if present) in vocabulary\_list. Must be castable to dtype.
* dtype: The type of features. Only string and integer types are supported. If None, it will be inferred from vocabulary\_list.
* default\_value: The integer ID value to return for out-of-vocabulary feature values, defaults to -1. This can not be specified with a positive num\_oov\_buckets.
* num\_oov\_buckets: Non-negative integer, the number of out-of-vocabulary buckets. All out-of-vocabulary inputs will be assigned IDs in the range [len(vocabulary\_list), len(vocabulary\_list)+num\_oov\_buckets) based on a hash of the input value. A positive num\_oov\_buckets can not be specified with default\_value.

#### Returns:

A SequenceCategoricalColumn.

#### Raises:

* ValueError: if vocabulary\_list is empty, or contains duplicate keys.
* ValueError: num\_oov\_buckets is a negative integer.
* ValueError: num\_oov\_buckets and default\_value are both specified.
* ValueError: if dtype is not integer or string.

# tf.feature\_column.sequence\_numeric\_column

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

Returns a feature column that represents sequences of numeric data.

### Aliases:

* tf.compat.v1.feature\_column.sequence\_numeric\_column
* tf.compat.v2.feature\_column.sequence\_numeric\_column
* tf.feature\_column.sequence\_numeric\_column

tf.feature\_column.sequence\_numeric\_column(

    key,

    shape=(1,),

    default\_value=0.0,

    dtype=tf.dtypes.float32,

    normalizer\_fn=None

)

Defined in [python/feature\_column/sequence\_feature\_column.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/sequence_feature_column.py).

#### Example:

temperature = sequence\_numeric\_column('temperature')

columns = [temperature]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

sequence\_feature\_layer = SequenceFeatures(columns)

sequence\_input, sequence\_length = sequence\_feature\_layer(features)

sequence\_length\_mask = tf.sequence\_mask(sequence\_length)

rnn\_cell = tf.keras.layers.SimpleRNNCell(hidden\_size)

rnn\_layer = tf.keras.layers.RNN(rnn\_cell)

outputs, state = rnn\_layer(sequence\_input, mask=sequence\_length\_mask)

#### Args:

* key: A unique string identifying the input features.
* shape: The shape of the input data per sequence id. E.g. if shape=(2,), each example must contain 2 \* sequence\_length values.
* default\_value: A single value compatible with dtype that is used for padding the sparse data into a dense Tensor.
* dtype: The type of values.
* normalizer\_fn: If not None, a function that can be used to normalize the value of the tensor after default\_value is applied for parsing. Normalizer function takes the input Tensor as its argument, and returns the output Tensor. (e.g. lambda x: (x - 3.0) / 4.2). Please note that even though the most common use case of this function is normalization, it can be used for any kind of Tensorflow transformations.

#### Returns:

A SequenceNumericColumn.

#### Raises:

* TypeError: if any dimension in shape is not an int.
* ValueError: if any dimension in shape is not a positive integer.
* ValueError: if dtype is not convertible to [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32).

# tf.feature\_column.shared\_embeddings

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

List of dense columns that convert from sparse, categorical input.

### Aliases:

* tf.compat.v2.feature\_column.shared\_embeddings
* tf.feature\_column.shared\_embeddings

tf.feature\_column.shared\_embeddings(

    categorical\_columns,

    dimension,

    combiner='mean',

    initializer=None,

    shared\_embedding\_collection\_name=None,

    ckpt\_to\_load\_from=None,

    tensor\_name\_in\_ckpt=None,

    max\_norm=None,

    trainable=True

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

This is similar to embedding\_column, except that it produces a list of embedding columns that share the same embedding weights.

Use this when your inputs are sparse and of the same type (e.g. watched and impression video IDs that share the same vocabulary), and you want to convert them to a dense representation (e.g., to feed to a DNN).

Inputs must be a list of categorical columns created by any of the categorical\_column\_\* function. They must all be of the same type and have the same arguments except key. E.g. they can be categorical\_column\_with\_vocabulary\_file with the same vocabulary\_file. Some or all columns could also be weighted\_categorical\_column.

Here is an example embedding of two features for a DNNClassifier model:

watched\_video\_id = categorical\_column\_with\_vocabulary\_file(

    'watched\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

impression\_video\_id = categorical\_column\_with\_vocabulary\_file(

    'impression\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

columns = shared\_embedding\_columns(

    [watched\_video\_id, impression\_video\_id], dimension=10)

estimator = tf.estimator.DNNClassifier(feature\_columns=columns, ...)

label\_column = ...

def input\_fn():

  features = tf.io.parse\_example(

      ..., features=make\_parse\_example\_spec(columns + [label\_column]))

  labels = features.pop(label\_column.name)

  return features, labels

estimator.train(input\_fn=input\_fn, steps=100)

Here is an example using shared\_embedding\_columns with model\_fn:

def model\_fn(features, ...):

  watched\_video\_id = categorical\_column\_with\_vocabulary\_file(

      'watched\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

  impression\_video\_id = categorical\_column\_with\_vocabulary\_file(

      'impression\_video\_id', video\_vocabulary\_file, video\_vocabulary\_size)

  columns = shared\_embedding\_columns(

      [watched\_video\_id, impression\_video\_id], dimension=10)

  dense\_tensor = input\_layer(features, columns)

  # Form DNN layers, calculate loss, and return EstimatorSpec.

  ...

#### Args:

* categorical\_columns: List of categorical columns created by acategorical\_column\_with\_\* function. These columns produce the sparse IDs that are inputs to the embedding lookup. All columns must be of the same type and have the same arguments except key. E.g. they can be categorical\_column\_with\_vocabulary\_file with the same vocabulary\_file. Some or all columns could also be weighted\_categorical\_column.
* dimension: An integer specifying dimension of the embedding, must be > 0.
* combiner: A string specifying how to reduce if there are multiple entries in a single row. Currently 'mean', 'sqrtn' and 'sum' are supported, with 'mean' the default. 'sqrtn' often achieves good accuracy, in particular with bag-of-words columns. Each of this can be thought as example level normalizations on the column. For more information, see tf.embedding\_lookup\_sparse.
* initializer: A variable initializer function to be used in embedding variable initialization. If not specified, defaults to [tf.compat.v1.truncated\_normal\_initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/truncated_normal_initializer) with mean 0.0 and standard deviation 1/sqrt(dimension).
* shared\_embedding\_collection\_name: Optional collective name of these columns. If not given, a reasonable name will be chosen based on the names of categorical\_columns.
* ckpt\_to\_load\_from: String representing checkpoint name/pattern from which to restore column weights. Required if tensor\_name\_in\_ckpt is not None.
* tensor\_name\_in\_ckpt: Name of the Tensor in ckpt\_to\_load\_from from which to restore the column weights. Required if ckpt\_to\_load\_from is not None.
* max\_norm: If not None, each embedding is clipped if its l2-norm is larger than this value, before combining.
* trainable: Whether or not the embedding is trainable. Default is True.

#### Returns:

A list of dense columns that converts from sparse input. The order of results follows the ordering of categorical\_columns.

#### Raises:

* ValueError: if dimension not > 0.
* ValueError: if any of the given categorical\_columns is of different type or has different arguments than the others.
* ValueError: if exactly one of ckpt\_to\_load\_from and tensor\_name\_in\_ckpt is specified.
* ValueError: if initializer is specified and is not callable.
* RuntimeError: if eager execution is enabled.

# tf.feature\_column.weighted\_categorical\_column

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

Applies weight values to a CategoricalColumn.

### Aliases:

* tf.compat.v1.feature\_column.weighted\_categorical\_column
* tf.compat.v2.feature\_column.weighted\_categorical\_column
* tf.feature\_column.weighted\_categorical\_column

tf.feature\_column.weighted\_categorical\_column(

    categorical\_column,

    weight\_feature\_key,

    dtype=tf.dtypes.float32

)

Defined in [python/feature\_column/feature\_column\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/feature_column/feature_column_v2.py).

Use this when each of your sparse inputs has both an ID and a value. For example, if you're representing text documents as a collection of word frequencies, you can provide 2 parallel sparse input features ('terms' and 'frequencies' below).

#### Example:

Input tf.Example objects:

[

  features {

    feature {

      key: "terms"

      value {bytes\_list {value: "very" value: "model"}}

    }

    feature {

      key: "frequencies"

      value {float\_list {value: 0.3 value: 0.1}}

    }

  },

  features {

    feature {

      key: "terms"

      value {bytes\_list {value: "when" value: "course" value: "human"}}

    }

    feature {

      key: "frequencies"

      value {float\_list {value: 0.4 value: 0.1 value: 0.2}}

    }

  }

]

categorical\_column = categorical\_column\_with\_hash\_bucket(

    column\_name='terms', hash\_bucket\_size=1000)

weighted\_column = weighted\_categorical\_column(

    categorical\_column=categorical\_column, weight\_feature\_key='frequencies')

columns = [weighted\_column, ...]

features = tf.io.parse\_example(..., features=make\_parse\_example\_spec(columns))

linear\_prediction, \_, \_ = linear\_model(features, columns)

This assumes the input dictionary contains a SparseTensor for key 'terms', and a SparseTensor for key 'frequencies'. These 2 tensors must have the same indices and dense shape.

#### Args:

* categorical\_column: A CategoricalColumn created by categorical\_column\_with\_\*functions.
* weight\_feature\_key: String key for weight values.
* dtype: Type of weights, such as [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32). Only float and integer weights are supported.

#### Returns:

A CategoricalColumn composed of two sparse features: one represents id, the other represents weight (value) of the id feature in that example.

#### Raises:

* ValueError: if dtype is not convertible to float.

Module: tf.image

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

Image processing and decoding ops.

See the [Images](https://tensorflow.org/api_guides/python/image) guide.

Classes

[class ResizeMethod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod)

Functions

[adjust\_brightness(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_brightness): Adjust the brightness of RGB or Grayscale images.

[adjust\_contrast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_contrast): Adjust contrast of RGB or grayscale images.

[adjust\_gamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_gamma): Performs Gamma Correction on the input image.

[adjust\_hue(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_hue): Adjust hue of RGB images.

[adjust\_jpeg\_quality(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_jpeg_quality): Adjust jpeg encoding quality of an RGB image.

[adjust\_saturation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_saturation): Adjust saturation of RGB images.

[central\_crop(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/central_crop): Crop the central region of the image(s).

[combined\_non\_max\_suppression(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/combined_non_max_suppression): Greedily selects a subset of bounding boxes in descending order of score.

[convert\_image\_dtype(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/convert_image_dtype): Convert image to dtype, scaling its values if needed.

[crop\_and\_resize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/crop_and_resize): Extracts crops from the input image tensor and resizes them.

[crop\_to\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/crop_to_bounding_box): Crops an image to a specified bounding box.

[decode\_and\_crop\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_and_crop_jpeg): Decode and Crop a JPEG-encoded image to a uint8 tensor.

[decode\_bmp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_bmp): Decode the first frame of a BMP-encoded image to a uint8 tensor.

[decode\_gif(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_gif): Decode the frame(s) of a GIF-encoded image to a uint8 tensor.

[decode\_image(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image): Function for decode\_bmp, decode\_gif, decode\_jpeg, and decode\_png.

[decode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg): Decode a JPEG-encoded image to a uint8 tensor.

[decode\_png(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_png): Decode a PNG-encoded image to a uint8 or uint16 tensor.

[draw\_bounding\_boxes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes): Draw bounding boxes on a batch of images.

[encode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_jpeg): JPEG-encode an image.

[encode\_png(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/encode_png): PNG-encode an image.

[extract\_glimpse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/extract_glimpse): Extracts a glimpse from the input tensor.

[extract\_jpeg\_shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/extract_jpeg_shape): Extract the shape information of a JPEG-encoded image.

[extract\_patches(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/extract_patches): Extract patches from images and put them in the "depth" output dimension.

[flip\_left\_right(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_left_right): Flip an image horizontally (left to right).

[flip\_up\_down(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_up_down): Flip an image vertically (upside down).

[grayscale\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/grayscale_to_rgb): Converts one or more images from Grayscale to RGB.

[hsv\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/hsv_to_rgb): Convert one or more images from HSV to RGB.

[image\_gradients(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/image_gradients): Returns image gradients (dy, dx) for each color channel.

[is\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/is_jpeg): Convenience function to check if the 'contents' encodes a JPEG image.

[non\_max\_suppression(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_overlaps(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_overlaps): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_padded(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_padded): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_with\_scores(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_with_scores): Greedily selects a subset of bounding boxes in descending order of score.

[pad\_to\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/pad_to_bounding_box): Pad image with zeros to the specified height and width.

[per\_image\_standardization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/per_image_standardization): Linearly scales each image in image to have mean 0 and variance 1.

[psnr(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/psnr): Returns the Peak Signal-to-Noise Ratio between a and b.

[random\_brightness(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_brightness): Adjust the brightness of images by a random factor.

[random\_contrast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_contrast): Adjust the contrast of an image or images by a random factor.

[random\_crop(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_crop): Randomly crops a tensor to a given size.

[random\_flip\_left\_right(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_flip_left_right): Randomly flip an image horizontally (left to right).

[random\_flip\_up\_down(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_flip_up_down): Randomly flips an image vertically (upside down).

[random\_hue(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_hue): Adjust the hue of RGB images by a random factor.

[random\_jpeg\_quality(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_jpeg_quality): Randomly changes jpeg encoding quality for inducing jpeg noise.

[random\_saturation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_saturation): Adjust the saturation of RGB images by a random factor.

[resize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize): Resize images to size using the specified method.

[resize\_with\_crop\_or\_pad(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_crop_or_pad): Crops and/or pads an image to a target width and height.

[resize\_with\_pad(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_pad): Resizes and pads an image to a target width and height.

[rgb\_to\_grayscale(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_grayscale): Converts one or more images from RGB to Grayscale.

[rgb\_to\_hsv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_hsv): Converts one or more images from RGB to HSV.

[rgb\_to\_yiq(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_yiq): Converts one or more images from RGB to YIQ.

[rgb\_to\_yuv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_yuv): Converts one or more images from RGB to YUV.

[rot90(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90): Rotate image(s) counter-clockwise by 90 degrees.

[sample\_distorted\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sample_distorted_bounding_box): Generate a single randomly distorted bounding box for an image.

[sobel\_edges(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sobel_edges): Returns a tensor holding Sobel edge maps.

[ssim(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ssim): Computes SSIM index between img1 and img2.

[ssim\_multiscale(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ssim_multiscale): Computes the MS-SSIM between img1 and img2.

[total\_variation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/total_variation): Calculate and return the total variation for one or more images.

[transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/transpose): Transpose image(s) by swapping the height and width dimension.

[yiq\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/yiq_to_rgb): Converts one or more images from YIQ to RGB.

[yuv\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/yuv_to_rgb): Converts one or more images from YUV to RGB.

# tf.compat.v1.image.resize\_area

Resize images to size using area interpolation.

tf.compat.v1.image.resize\_area(

    images,

    size,

    align\_corners=False,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

Input images can be of different types but output images are always float.

The range of pixel values for the output image might be slightly different from the range for the input image because of limited numerical precision. To guarantee an output range, for example [0.0, 1.0], apply [tf.clip\_by\_value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/clip_by_value) to the output.

Each output pixel is computed by first transforming the pixel's footprint into the input tensor and then averaging the pixels that intersect the footprint. An input pixel's contribution to the average is weighted by the fraction of its area that intersects the footprint. This is the same as OpenCV's INTER\_AREA.

#### Args:

* images: A Tensor. Must be one of the following types: int8, uint8, int16, uint16, int32, int64, half, float32, float64. 4-D with shape [batch, height, width, channels].
* size: A 1-D int32 Tensor of 2 elements: new\_height, new\_width. The new size for the images.
* align\_corners: An optional bool. Defaults to False. If true, the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. Defaults to false.
* name: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.compat.v1.image.resize\_bicubic

tf.compat.v1.image.resize\_bicubic(

    images,

    size,

    align\_corners=False,

    name=None,

    half\_pixel\_centers=False

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

# tf.compat.v1.image.resize\_bilinear

tf.compat.v1.image.resize\_bilinear(

    images,

    size,

    align\_corners=False,

    name=None,

    half\_pixel\_centers=False

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

# tf.compat.v1.image.resize\_image\_with\_pad

Resizes and pads an image to a target width and height.

tf.compat.v1.image.resize\_image\_with\_pad(

    image,

    target\_height,

    target\_width,

    method=ResizeMethodV1.BILINEAR,

    align\_corners=False

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Resizes an image to a target width and height by keeping the aspect ratio the same without distortion. If the target dimensions don't match the image dimensions, the image is resized and then padded with zeroes to match requested dimensions.

#### Args:

* image: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* target\_height: Target height.
* target\_width: Target width.
* method: Method to use for resizing image. See resize\_images()
* align\_corners: bool. If True, the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. Defaults to False.

#### Raises:

* ValueError: if target\_height or target\_width are zero or negative.

#### Returns:

Resized and padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.compat.v1.image.resize\_nearest\_neighbor

tf.compat.v1.image.resize\_nearest\_neighbor(

    images,

    size,

    align\_corners=False,

    name=None,

    half\_pixel\_centers=False

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

# tf.image.adjust\_brightness

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

Adjust the brightness of RGB or Grayscale images.

### Aliases:

* tf.compat.v1.image.adjust\_brightness
* tf.compat.v2.image.adjust\_brightness
* tf.image.adjust\_brightness

tf.image.adjust\_brightness(

    image,

    delta

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is a convenience method that converts RGB images to float representation, adjusts their brightness, and then converts them back to the original data type. If several adjustments are chained, it is advisable to minimize the number of redundant conversions.

The value delta is added to all components of the tensor image. image is converted to float and scaled appropriately if it is in fixed-point representation, and delta is converted to the same data type. For regular images, delta should be in the range [0,1), as it is added to the image in floating point representation, where pixel values are in the [0,1) range.

#### Args:

* **image**: RGB image or images to adjust.
* **delta**: A scalar. Amount to add to the pixel values.

#### Returns:

A brightness-adjusted tensor of the same shape and type as image.

# tf.image.adjust\_contrast

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

Adjust contrast of RGB or grayscale images.

### Aliases:

* tf.compat.v1.image.adjust\_contrast
* tf.compat.v2.image.adjust\_contrast
* tf.image.adjust\_contrast

tf.image.adjust\_contrast(

    images,

    contrast\_factor

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is a convenience method that converts RGB images to float representation, adjusts their contrast, and then converts them back to the original data type. If several adjustments are chained, it is advisable to minimize the number of redundant conversions.

images is a tensor of at least 3 dimensions. The last 3 dimensions are interpreted as [height, width, channels]. The other dimensions only represent a collection of images, such as [batch, height, width, channels].

Contrast is adjusted independently for each channel of each image.

For each channel, this Op computes the mean of the image pixels in the channel and then adjusts each component x of each pixel to (x - mean) \* contrast\_factor + mean.

#### Args:

* **images**: Images to adjust. At least 3-D.
* **contrast\_factor**: A float multiplier for adjusting contrast.

#### Returns:

The contrast-adjusted image or images.

# tf.image.adjust\_gamma

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

Performs Gamma Correction on the input image.

### Aliases:

* tf.compat.v1.image.adjust\_gamma
* tf.compat.v2.image.adjust\_gamma
* tf.image.adjust\_gamma

tf.image.adjust\_gamma(

    image,

    gamma=1,

    gain=1

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Also known as Power Law Transform. This function converts the input images at first to float representation, then transforms them pixelwise according to the equation Out = gain \* In\*\*gamma, and then converts the back to the original data type.

#### Args:

* **image**: RGB image or images to adjust.
* **gamma**: A scalar or tensor. Non negative real number.
* **gain**: A scalar or tensor. The constant multiplier.

#### Returns:

A Tensor. A Gamma-adjusted tensor of the same shape and type as image.

#### Raises:

* **ValueError**: If gamma is negative.

#### Notes:

For gamma greater than 1, the histogram will shift towards left and the output image will be darker than the input image. For gamma less than 1, the histogram will shift towards right and the output image will be brighter than the input image.

#### References:

[1] http://en.wikipedia.org/wiki/Gamma\_correction

# tf.image.adjust\_hue

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

Adjust hue of RGB images.

### Aliases:

* tf.compat.v1.image.adjust\_hue
* tf.compat.v2.image.adjust\_hue
* tf.image.adjust\_hue

tf.image.adjust\_hue(

    image,

    delta,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is a convenience method that converts an RGB image to float representation, converts it to HSV, add an offset to the hue channel, converts back to RGB and then back to the original data type. If several adjustments are chained it is advisable to minimize the number of redundant conversions.

image is an RGB image. The image hue is adjusted by converting the image(s) to HSV and rotating the hue channel (H) by delta. The image is then converted back to RGB.

delta must be in the interval [-1, 1].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **delta**: float. How much to add to the hue channel.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.adjust\_jpeg\_quality

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

Adjust jpeg encoding quality of an RGB image.

### Aliases:

* tf.compat.v1.image.adjust\_jpeg\_quality
* tf.compat.v2.image.adjust\_jpeg\_quality
* tf.image.adjust\_jpeg\_quality

tf.image.adjust\_jpeg\_quality(

    image,

    jpeg\_quality,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is a convenience method that adjusts jpeg encoding quality of an RGB image.

image is an RGB image. The image's encoding quality is adjusted to jpeg\_quality. jpeg\_qualitymust be in the interval [0, 100].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **jpeg\_quality**: Python int or Tensor of type int32. jpeg encoding quality.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.adjust\_saturation

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

Adjust saturation of RGB images.

### Aliases:

* tf.compat.v1.image.adjust\_saturation
* tf.compat.v2.image.adjust\_saturation
* tf.image.adjust\_saturation

tf.image.adjust\_saturation(

    image,

    saturation\_factor,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is a convenience method that converts RGB images to float representation, converts them to HSV, add an offset to the saturation channel, converts back to RGB and then back to the original data type. If several adjustments are chained it is advisable to minimize the number of redundant conversions.

image is an RGB image or images. The image saturation is adjusted by converting the images to HSV and multiplying the saturation (S) channel by saturation\_factor and clipping. The images are then converted back to RGB.

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **saturation\_factor**: float. Factor to multiply the saturation by.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.central\_crop

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

Crop the central region of the image(s).

### Aliases:

* tf.compat.v1.image.central\_crop
* tf.compat.v2.image.central\_crop
* tf.image.central\_crop

tf.image.central\_crop(

    image,

    central\_fraction

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Remove the outer parts of an image but retain the central region of the image along each dimension. If we specify central\_fraction = 0.5, this function returns the region marked with "X" in the below diagram.

 --------

|        |

|  XXXX  |

|  XXXX  |

|        |   where "X" is the central 50% of the image.

 --------

This function works on either a single image (image is a 3-D Tensor), or a batch of images (image is a 4-D Tensor).

#### Args:

* **image**: Either a 3-D float Tensor of shape [height, width, depth], or a 4-D Tensor of shape [batch\_size, height, width, depth].
* **central\_fraction**: float (0, 1], fraction of size to crop

#### Raises:

* **ValueError**: if central\_crop\_fraction is not within (0, 1].

#### Returns:

3-D / 4-D float Tensor, as per the input.

# tf.image.combined\_non\_max\_suppression

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.combined\_non\_max\_suppression
* tf.compat.v2.image.combined\_non\_max\_suppression
* tf.image.combined\_non\_max\_suppression

tf.image.combined\_non\_max\_suppression(

    boxes,

    scores,

    max\_output\_size\_per\_class,

    max\_total\_size,

    iou\_threshold=0.5,

    score\_threshold=float('-inf'),

    pad\_per\_class=False,

    clip\_boxes=True,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This operation performs non\_max\_suppression on the inputs per batch, across all classes. Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Also note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is the final boxes, scores and classes tensor returned after performing non\_max\_suppression.

#### Args:

* **boxes**: A 4-D float Tensor of shape [batch\_size, num\_boxes, q, 4]. If q is 1 then same boxes are used for all classes otherwise, if q is equal to number of classes, class-specific boxes are used.
* **scores**: A 3-D float Tensor of shape [batch\_size, num\_boxes, num\_classes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size\_per\_class**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression per class
* **max\_total\_size**: A scalar representing maximum number of boxes retained over all classes.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **pad\_per\_class**: If false, the output nmsed boxes, scores and classes are padded/clipped to max\_total\_size. If true, the output nmsed boxes, scores and classes are padded to be of length max\_size\_per\_class\*num\_classes, unless it exceeds max\_total\_size in which case it is clipped to max\_total\_size. Defaults to false.
* **clip\_boxes**: If true, the coordinates of output nmsed boxes will be clipped to [0, 1]. If false, output the box coordinates as it is. Defaults to true.
* **name**: A name for the operation (optional).

#### Returns:

'nmsed\_boxes': A [batch\_size, max\_detections, 4] float32 tensor containing the non-max suppressed boxes. 'nmsed\_scores': A [batch\_size, max\_detections] float32 tensor containing the scores for the boxes. 'nmsed\_classes': A [batch\_size, max\_detections] float32 tensor containing the class for boxes. 'valid\_detections': A [batch\_size] int32 tensor indicating the number of valid detections per batch item. Only the top valid\_detections[i] entries in nms\_boxes[i], nms\_scores[i] and nms\_class[i] are valid. The rest of the entries are zero paddings.

# tf.image.convert\_image\_dtype

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/convert_image_dtype#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/convert_image_dtype#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/convert_image_dtype#used_in_the_tutorials)

Convert image to dtype, scaling its values if needed.

### Aliases:

* tf.compat.v1.image.convert\_image\_dtype
* tf.compat.v2.image.convert\_image\_dtype
* tf.image.convert\_image\_dtype

tf.image.convert\_image\_dtype(

    image,

    dtype,

    saturate=False,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Images that are represented using floating point values are expected to have values in the range [0,1). Image data stored in integer data types are expected to have values in the range [0,MAX], where MAXis the largest positive representable number for the data type.

This op converts between data types, scaling the values appropriately before casting.

Note that converting from floating point inputs to integer types may lead to over/underflow problems. Set saturate to True to avoid such problem in problematic conversions. If enabled, saturation will clip the output into the allowed range before performing a potentially dangerous cast (and only before performing such a cast, i.e., when casting from a floating point to an integer type, and when casting from a signed to an unsigned type; saturate has no effect on casts between floats, or on casts that increase the type's range).

#### Args:

* **image**: An image.
* **dtype**: A DType to convert image to.
* **saturate**: If True, clip the input before casting (if necessary).
* **name**: A name for this operation (optional).

#### Returns:

image, converted to dtype.

# tf.image.crop\_and\_resize

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

Extracts crops from the input image tensor and resizes them.

### Aliases:

* tf.compat.v2.image.crop\_and\_resize
* tf.image.crop\_and\_resize

tf.image.crop\_and\_resize(

    image,

    boxes,

    box\_indices,

    crop\_size,

    method='bilinear',

    extrapolation\_value=0,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Extracts crops from the input image tensor and resizes them using bilinear sampling or nearest neighbor sampling (possibly with aspect ratio change) to a common output size specified by crop\_size. This is more general than the crop\_to\_bounding\_box op which extracts a fixed size slice from the input image and does not allow resizing or aspect ratio change.

Returns a tensor with crops from the input image at positions defined at the bounding box locations in boxes. The cropped boxes are all resized (with bilinear or nearest neighbor interpolation) to a fixedsize = [crop\_height, crop\_width]. The result is a 4-D tensor [num\_boxes, crop\_height, crop\_width, depth]. The resizing is corner aligned. In particular, if boxes = [[0, 0, 1, 1]], the method will give identical results to using tf.compat.v1.image.resize\_bilinear() ortf.compat.v1.image.resize\_nearest\_neighbor()(depends on the method argument) withalign\_corners=True.

#### Args:

* **image**: A 4-D tensor of shape [batch, image\_height, image\_width, depth]. Both image\_height and image\_width need to be positive.
* **boxes**: A 2-D tensor of shape [num\_boxes, 4]. The i-th row of the tensor specifies the coordinates of a box in the box\_ind[i] image and is specified in normalized coordinates [y1, x1, y2, x2]. A normalized coordinate value of y is mapped to the image coordinate at y \* (image\_height - 1), so as the [0, 1] interval of normalized image height is mapped to [0, image\_height - 1] in image height coordinates. We do allow y1 > y2, in which case the sampled crop is an up-down flipped version of the original image. The width dimension is treated similarly. Normalized coordinates outside the [0, 1] range are allowed, in which case we use extrapolation\_value to extrapolate the input image values.
* **box\_indices**: A 1-D tensor of shape [num\_boxes] with int32 values in [0, batch). The value of box\_ind[i] specifies the image that the i-th box refers to.
* **crop\_size**: A 1-D tensor of 2 elements, size = [crop\_height, crop\_width]. All cropped image patches are resized to this size. The aspect ratio of the image content is not preserved. Both crop\_height and crop\_width need to be positive.
* **method**: An optional string specifying the sampling method for resizing. It can be either "bilinear" or "nearest" and default to "bilinear". Currently two sampling methods are supported: Bilinear and Nearest Neighbor.
* **extrapolation\_value**: An optional float. Defaults to 0. Value used for extrapolation, when applicable.
* **name**: A name for the operation (optional).

#### Returns:

A 4-D tensor of shape [num\_boxes, crop\_height, crop\_width, depth].

# tf.image.crop\_to\_bounding\_box

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

Crops an image to a specified bounding box.

### Aliases:

* tf.compat.v1.image.crop\_to\_bounding\_box
* tf.compat.v2.image.crop\_to\_bounding\_box
* tf.image.crop\_to\_bounding\_box

tf.image.crop\_to\_bounding\_box(

    image,

    offset\_height,

    offset\_width,

    target\_height,

    target\_width

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This op cuts a rectangular part out of image. The top-left corner of the returned image is at offset\_height, offset\_width in image, and its lower-right corner is at offset\_height + target\_height, offset\_width + target\_width.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **offset\_height**: Vertical coordinate of the top-left corner of the result in the input.
* **offset\_width**: Horizontal coordinate of the top-left corner of the result in the input.
* **target\_height**: Height of the result.
* **target\_width**: Width of the result.

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, target\_height, target\_width, channels] If image was 3-D, a 3-D float Tensor of shape [target\_height, target\_width, channels]

#### Raises:

* **ValueError**: If the shape of image is incompatible with the offset\_\* or target\_\*arguments, or either offset\_height or offset\_width is negative, or either target\_heightor target\_width is not positive.

# tf.image.draw\_bounding\_boxes

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

Draw bounding boxes on a batch of images.

### Aliases:

* tf.compat.v2.image.draw\_bounding\_boxes
* tf.image.draw\_bounding\_boxes

tf.image.draw\_bounding\_boxes(

    images,

    boxes,

    colors,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a copy of images but draws on top of the pixels zero or more bounding boxes specified by the locations in boxes. The coordinates of the each bounding box in boxes are encoded as [y\_min, x\_min, y\_max, x\_max]. The bounding box coordinates are floats in [0.0, 1.0] relative to the width and height of the underlying image.

For example, if an image is 100 x 200 pixels (height x width) and the bounding box is [0.1, 0.2, 0.5, 0.9], the upper-left and bottom-right coordinates of the bounding box will be (40, 10) to (180, 50) (in (x,y) coordinates).

Parts of the bounding box may fall outside the image.

#### Args:

* **images**: A Tensor. Must be one of the following types: float32, half. 4-D with shape [batch, height, width, depth]. A batch of images.
* **boxes**: A Tensor of type float32. 3-D with shape [batch, num\_bounding\_boxes, 4]containing bounding boxes.
* **colors**: A Tensor of type float32. 2-D. A list of RGBA colors to cycle through for the boxes.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.encode\_png

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

PNG-encode an image.

### Aliases:

* tf.compat.v1.image.encode\_png
* tf.compat.v2.image.encode\_png
* tf.image.encode\_png

tf.image.encode\_png(

    image,

    compression=-1,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

image is a 3-D uint8 or uint16 Tensor of shape [height, width, channels] where channels is:

* 1: for grayscale.
* 2: for grayscale + alpha.
* 3: for RGB.
* 4: for RGBA.

The ZLIB compression level, compression, can be -1 for the PNG-encoder default or a value from 0 to 9. 9 is the highest compression level, generating the smallest output, but is slower.

#### Args:

* **image**: A Tensor. Must be one of the following types: uint8, uint16. 3-D with shape [height, width, channels].
* **compression**: An optional int. Defaults to -1. Compression level.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.image.extract\_glimpse

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

Extracts a glimpse from the input tensor.

### Aliases:

* tf.compat.v2.image.extract\_glimpse
* tf.image.extract\_glimpse

tf.image.extract\_glimpse(

    input,

    size,

    offsets,

    centered=True,

    normalized=True,

    noise='uniform',

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Returns a set of windows called glimpses extracted at location offsets from the input tensor. If the windows only partially overlaps the inputs, the non overlapping areas will be filled with random noise.

The result is a 4-D tensor of shape [batch\_size, glimpse\_height, glimpse\_width, channels]. The channels and batch dimensions are the same as that of the input tensor. The height and width of the output windows are specified in the size parameter.

The argument normalized and centered controls how the windows are built:

* If the coordinates are normalized but not centered, 0.0 and 1.0 correspond to the minimum and maximum of each height and width dimension.
* If the coordinates are both normalized and centered, they range from -1.0 to 1.0. The coordinates (-1.0, -1.0) correspond to the upper left corner, the lower right corner is located at (1.0, 1.0) and the center is at (0, 0).
* If the coordinates are not normalized they are interpreted as numbers of pixels.

#### Args:

* **input**: A Tensor of type float32. A 4-D float tensor of shape [batch\_size, height, width, channels].
* **size**: A Tensor of type int32. A 1-D tensor of 2 elements containing the size of the glimpses to extract. The glimpse height must be specified first, following by the glimpse width.
* **offsets**: A Tensor of type float32. A 2-D integer tensor of shape [batch\_size, 2]containing the y, x locations of the center of each window.
* **centered**: An optional bool. Defaults to True. indicates if the offset coordinates are centered relative to the image, in which case the (0, 0) offset is relative to the center of the input images. If false, the (0,0) offset corresponds to the upper left corner of the input images.
* **normalized**: An optional bool. Defaults to True. indicates if the offset coordinates are normalized.
* **noise**: An optional string. Defaults to uniform. indicates if the noise should be uniform(uniform distribution), gaussian (gaussian distribution), or zero (zero padding).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.image.extract\_patches

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

Extract patches from images and put them in the "depth" output dimension.

### Aliases:

* tf.compat.v1.image.extract\_patches
* tf.compat.v2.image.extract\_patches
* tf.image.extract\_patches

tf.image.extract\_patches(

    images,

    sizes,

    strides,

    rates,

    padding,

    name=None

)

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

#### Args:

* **images**: A 4-D Tensor with shape `[batch, in\_rows, in\_cols, depth]
* **sizes**: The size of the sliding window for each dimension of images.
* **strides**: A 1-D Tensor of length 4. How far the centers of two consecutive patches are in the images. Must be: [1, stride\_rows, stride\_cols, 1].
* **rates**: A 1-D Tensor of length 4. Must be: [1, rate\_rows, rate\_cols, 1]. This is the input stride, specifying how far two consecutive patch samples are in the input. Equivalent to extracting patches with patch\_sizes\_eff = patch\_sizes + (patch\_sizes - 1) \* (rates - 1), followed by subsampling them spatially by a factor of rates. This is equivalent to ratein dilated (a.k.a. Atrous) convolutions.
* **padding**: The type of padding algorithm to use. We specify the size-related attributes as: ```python ksizes = [1, ksize\_rows, ksize\_cols, 1] strides = [1, strides\_rows, strides\_cols, 1] rates = [1, rates\_rows, rates\_cols, 1]
* **name**: A name for the operation (optional).

#### Returns:

A 4-D Tensor. Has the same type as images, and with shape [batch, out\_rows, out\_cols, ksize\_rows \* ksize\_cols \* depth] containing image patches with size ksize\_rows x ksize\_cols x depth vectorized in the "depth" dimension. Note out\_rows and out\_cols are the dimensions of the output patches.

# tf.image.flip\_left\_right

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_left_right#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_left_right#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_left_right#used_in_the_tutorials)

Flip an image horizontally (left to right).

### Aliases:

* tf.compat.v1.image.flip\_left\_right
* tf.compat.v2.image.flip\_left\_right
* tf.image.flip\_left\_right

tf.image.flip\_left\_right(image)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Outputs the contents of image flipped along the width dimension.

See also reverse().

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.flip\_up\_down

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

Flip an image vertically (upside down).

### Aliases:

* tf.compat.v1.image.flip\_up\_down
* tf.compat.v2.image.flip\_up\_down
* tf.image.flip\_up\_down

tf.image.flip\_up\_down(image)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs the contents of image flipped along the height dimension.

See also reverse().

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].

#### Returns:

A Tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.grayscale\_to\_rgb

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

Converts one or more images from Grayscale to RGB.

### Aliases:

* tf.compat.v1.image.grayscale\_to\_rgb
* tf.compat.v2.image.grayscale\_to\_rgb
* tf.image.grayscale\_to\_rgb

tf.image.grayscale\_to\_rgb(

    images,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same DType and rank as images. The size of the last dimension of the output is 3, containing the RGB value of the pixels. The input images' last dimension must be size 1.

#### Args:

* **images**: The Grayscale tensor to convert. Last dimension must be size 1.
* **name**: A name for the operation (optional).

#### Returns:

The converted grayscale image(s).

# tf.image.hsv\_to\_rgb

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

Convert one or more images from HSV to RGB.

### Aliases:

* tf.compat.v1.image.hsv\_to\_rgb
* tf.compat.v2.image.hsv\_to\_rgb
* tf.image.hsv\_to\_rgb

tf.image.hsv\_to\_rgb(

    images,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the value in images are in [0,1].

See rgb\_to\_hsv for a description of the HSV encoding.

#### Args:

* **images**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 1-D or higher rank. HSV data to convert. Last dimension must be size 3.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.image\_gradients

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

Returns image gradients (dy, dx) for each color channel.

### Aliases:

* tf.compat.v1.image.image\_gradients
* tf.compat.v2.image.image\_gradients
* tf.image.image\_gradients

tf.image.image\_gradients(image)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Both output tensors have the same shape as the input: [batch\_size, h, w, d]. The gradient values are organized so that [I(x+1, y) - I(x, y)] is in location (x, y). That means that dy will always have zeros in the last row, and dx will always have zeros in the last column.

#### Arguments:

* **image**: Tensor with shape [batch\_size, h, w, d].

#### Returns:

Pair of tensors (dy, dx) holding the vertical and horizontal image gradients (1-step finite difference).

#### Raises:

* **ValueError**: If image is not a 4D tensor.

# tf.image.non\_max\_suppression

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression
* tf.compat.v2.image.non\_max\_suppression
* tf.image.non\_max\_suppression

tf.image.non\_max\_suppression(

    boxes,

    scores,

    max\_output\_size,

    iou\_threshold=0.5,

    score\_threshold=float('-inf'),

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather)operation. For example:

selected\_indices = tf.image.non\_max\_suppression(

    boxes, scores, max\_output\_size, iou\_threshold)

selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.

# tf.image.non\_max\_suppression\_overlaps

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_overlaps
* tf.compat.v2.image.non\_max\_suppression\_overlaps
* tf.image.non\_max\_suppression\_overlaps

tf.image.non\_max\_suppression\_overlaps(

    overlaps,

    scores,

    max\_output\_size,

    overlap\_threshold=0.5,

    score\_threshold=float('-inf'),

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Prunes away boxes that have high overlap with previously selected boxes. N-by-n overlap values are supplied as square matrix. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) operation. For example:

selected\_indices = tf.image.non\_max\_suppression\_overlaps(

    overlaps, scores, max\_output\_size, iou\_threshold)

selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **overlaps**: A 2-D float Tensor of shape [num\_boxes, num\_boxes].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **overlap\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to the provided overlap values.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the overlaps tensor, where M <= max\_output\_size.

# tf.image.non\_max\_suppression\_padded

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_padded
* tf.compat.v2.image.non\_max\_suppression\_padded
* tf.image.non\_max\_suppression\_padded

tf.image.non\_max\_suppression\_padded(

    boxes,

    scores,

    max\_output\_size,

    iou\_threshold=0.5,

    score\_threshold=float('-inf'),

    pad\_to\_max\_output\_size=False,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Performs algorithmically equivalent operation to tf.image.non\_max\_suppression, with the addition of an optional parameter which zero-pads the output to be of size max\_output\_size. The output of this operation is a tuple containing the set of integers indexing into the input collection of bounding boxes representing the selected boxes and the number of valid indices in the index set. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice) and [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) operations. For example:

selected\_indices\_padded, num\_valid = tf.image.non\_max\_suppression\_padded(

    boxes, scores, max\_output\_size, iou\_threshold,

    score\_threshold, pad\_to\_max\_output\_size=True)

selected\_indices = tf.slice(

    selected\_indices\_padded, tf.constant([0]), num\_valid)

selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **pad\_to\_max\_output\_size**: bool. If True, size of selected\_indices output is padded to max\_output\_size.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.
* **valid\_outputs**: A scalar integer Tensor denoting how many elements in selected\_indicesare valid. Valid elements occur first, then padding.

# tf.image.non\_max\_suppression\_with\_scores

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_with\_scores
* tf.compat.v2.image.non\_max\_suppression\_with\_scores
* tf.image.non\_max\_suppression\_with\_scores

tf.image.non\_max\_suppression\_with\_scores(

    boxes,

    scores,

    max\_output\_size,

    iou\_threshold=0.5,

    score\_threshold=float('-inf'),

    soft\_nms\_sigma=0.0,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather)operation. For example:

selected\_indices, selected\_scores = tf.image.non\_max\_suppression\_v2(

    boxes, scores, max\_output\_size, iou\_threshold=1.0, score\_threshold=0.1,

    soft\_nms\_sigma=0.5)

selected\_boxes = tf.gather(boxes, selected\_indices)

This function generalizes the [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression) op by also supporting a Soft-NMS (with Gaussian weighting) mode (c.f. Bodla et al, https://arxiv.org/abs/1704.04503) where boxes reduce the score of other overlapping boxes instead of directly causing them to be pruned. Consequently, in contrast to [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression),tf.image.non\_max\_suppression\_v2 returns the new scores of each input box in the second output, selected\_scores.

To enable this Soft-NMS mode, set the soft\_nms\_sigma parameter to be larger than 0. When soft\_nms\_sigma equals 0, the behavior of tf.image.non\_max\_suppression\_v2 is identical to that of [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression) (except for the extra output) both in function and in running time.

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **soft\_nms\_sigma**: A scalar float representing the Soft NMS sigma parameter; See Bodla et al, https://arxiv.org/abs/1704.04503). When soft\_nms\_sigma=0.0 (which is default), we fall back to standard (hard) NMS.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.
* **selected\_scores**: A 1-D float tensor of shape [M] representing the corresponding scores for each selected box, where M <= max\_output\_size. Scores only differ from corresponding input scores when using Soft NMS (i.e. when soft\_nms\_sigma>0)

# tf.image.pad\_to\_bounding\_box

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

Pad image with zeros to the specified height and width.

### Aliases:

* tf.compat.v1.image.pad\_to\_bounding\_box
* tf.compat.v2.image.pad\_to\_bounding\_box
* tf.image.pad\_to\_bounding\_box

tf.image.pad\_to\_bounding\_box(

    image,

    offset\_height,

    offset\_width,

    target\_height,

    target\_width

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Adds offset\_height rows of zeros on top, offset\_width columns of zeros on the left, and then pads the image on the bottom and right with zeros until it has dimensions target\_height, target\_width.

This op does nothing if offset\_\* is zero and the image already has size target\_height by target\_width.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **offset\_height**: Number of rows of zeros to add on top.
* **offset\_width**: Number of columns of zeros to add on the left.
* **target\_height**: Height of output image.
* **target\_width**: Width of output image.

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, target\_height, target\_width, channels] If image was 3-D, a 3-D float Tensor of shape [target\_height, target\_width, channels]

#### Raises:

* **ValueError**: If the shape of image is incompatible with the offset\_\* or target\_\*arguments, or either offset\_height or offset\_width is negative.

# tf.image.per\_image\_standardization

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

Linearly scales each image in image to have mean 0 and variance 1.

### Aliases:

* tf.compat.v1.image.per\_image\_standardization
* tf.compat.v2.image.per\_image\_standardization
* tf.image.per\_image\_standardization

tf.image.per\_image\_standardization(image)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

For each 3-D image x in image, computes (x - mean) / adjusted\_stddev, where

* mean is the average of all values in x
* adjusted\_stddev = max(stddev, 1.0/sqrt(N)) is capped away from 0 to protect against division by 0 when handling uniform images
  + N is the number of elements in x
  + stddev is the standard deviation of all values in x

#### Args:

* **image**: An n-D Tensor with at least 3 dimensions, the last 3 of which are the dimensions of each image.

#### Returns:

A Tensor with same shape and dtype as image.

#### Raises:

* **ValueError**: if the shape of 'image' is incompatible with this function.

# tf.image.psnr

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

Returns the Peak Signal-to-Noise Ratio between a and b.

### Aliases:

* tf.compat.v1.image.psnr
* tf.compat.v2.image.psnr
* tf.image.psnr

tf.image.psnr(

    a,

    b,

    max\_val,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This is intended to be used on signals (or images). Produces a PSNR value for each image in batch.

The last three dimensions of input are expected to be [height, width, depth].

#### Example:

    # Read images from file.

    im1 = tf.decode\_png('path/to/im1.png')

    im2 = tf.decode\_png('path/to/im2.png')

    # Compute PSNR over tf.uint8 Tensors.

    psnr1 = tf.image.psnr(im1, im2, max\_val=255)

    # Compute PSNR over tf.float32 Tensors.

    im1 = tf.image.convert\_image\_dtype(im1, tf.float32)

    im2 = tf.image.convert\_image\_dtype(im2, tf.float32)

    psnr2 = tf.image.psnr(im1, im2, max\_val=1.0)

    # psnr1 and psnr2 both have type tf.float32 and are almost equal.

#### Arguments:

* **a**: First set of images.
* **b**: Second set of images.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **name**: Namespace to embed the computation in.

#### Returns:

The scalar PSNR between a and b. The returned tensor has type [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) and shape [batch\_size, 1].

# tf.image.random\_brightness

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

Adjust the brightness of images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_brightness
* tf.compat.v2.image.random\_brightness
* tf.image.random\_brightness

tf.image.random\_brightness(

    image,

    max\_delta,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Equivalent to adjust\_brightness() using a delta randomly picked in the interval [-max\_delta, max\_delta).

#### Args:

* **image**: An image or images to adjust.
* **max\_delta**: float, must be non-negative.
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

The brightness-adjusted image(s).

#### Raises:

* **ValueError**: if max\_delta is negative.

# tf.image.random\_contrast

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

Adjust the contrast of an image or images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_contrast
* tf.compat.v2.image.random\_contrast
* tf.image.random\_contrast

tf.image.random\_contrast(

    image,

    lower,

    upper,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Equivalent to adjust\_contrast() but uses a contrast\_factor randomly picked in the interval [lower, upper].

#### Args:

* **image**: An image tensor with 3 or more dimensions.
* **lower**: float. Lower bound for the random contrast factor.
* **upper**: float. Upper bound for the random contrast factor.
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

The contrast-adjusted image(s).

#### Raises:

* **ValueError**: if upper <= lower or if lower < 0.

# tf.image.random\_crop

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_crop#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_crop#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_crop#used_in_the_tutorials)

Randomly crops a tensor to a given size.

### Aliases:

* tf.compat.v1.image.random\_crop
* tf.compat.v1.random\_crop
* tf.compat.v2.image.random\_crop
* tf.image.random\_crop

tf.image.random\_crop(

    value,

    size,

    seed=None,

    name=None

)

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

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Slices a shape size portion out of value at a uniformly chosen offset. Requires value.shape >= size.

If a dimension should not be cropped, pass the full size of that dimension. For example, RGB images can be cropped with size = [crop\_height, crop\_width, 3].

#### Args:

* **value**: Input tensor to crop.
* **size**: 1-D tensor with size the rank of value.
* **seed**: Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: A name for this operation (optional).

#### Returns:

A cropped tensor of the same rank as value and shape size.

# tf.image.random\_flip\_left\_right

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

Randomly flip an image horizontally (left to right).

### Aliases:

* tf.compat.v1.image.random\_flip\_left\_right
* tf.compat.v2.image.random\_flip\_left\_right
* tf.image.random\_flip\_left\_right

tf.image.random\_flip\_left\_right(

    image,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

With a 1 in 2 chance, outputs the contents of image flipped along the second dimension, which is width. Otherwise output the image as-is.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.random\_flip\_up\_down

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

Randomly flips an image vertically (upside down).

### Aliases:

* tf.compat.v1.image.random\_flip\_up\_down
* tf.compat.v2.image.random\_flip\_up\_down
* tf.image.random\_flip\_up\_down

tf.image.random\_flip\_up\_down(

    image,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

With a 1 in 2 chance, outputs the contents of image flipped along the first dimension, which is height. Otherwise output the image as-is.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.random\_hue

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

Adjust the hue of RGB images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_hue
* tf.compat.v2.image.random\_hue
* tf.image.random\_hue

tf.image.random\_hue(

    image,

    max\_delta,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Equivalent to adjust\_hue() but uses a delta randomly picked in the interval [-max\_delta, max\_delta].

max\_delta must be in the interval [0, 0.5].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **max\_delta**: float. Maximum value for the random delta.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if max\_delta is invalid.

# tf.image.random\_jpeg\_quality

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

Randomly changes jpeg encoding quality for inducing jpeg noise.

### Aliases:

* tf.compat.v1.image.random\_jpeg\_quality
* tf.compat.v2.image.random\_jpeg\_quality
* tf.image.random\_jpeg\_quality

tf.image.random\_jpeg\_quality(

    image,

    min\_jpeg\_quality,

    max\_jpeg\_quality,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

min\_jpeg\_quality must be in the interval [0, 100] and less than max\_jpeg\_quality.max\_jpeg\_quality must be in the interval [0, 100].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **min\_jpeg\_quality**: Minimum jpeg encoding quality to use.
* **max\_jpeg\_quality**: Maximum jpeg encoding quality to use.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if min\_jpeg\_quality or max\_jpeg\_quality is invalid.

# tf.image.random\_saturation

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

Adjust the saturation of RGB images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_saturation
* tf.compat.v2.image.random\_saturation
* tf.image.random\_saturation

tf.image.random\_saturation(

    image,

    lower,

    upper,

    seed=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Equivalent to adjust\_saturation() but uses a saturation\_factor randomly picked in the interval [lower, upper].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **lower**: float. Lower bound for the random saturation factor.
* **upper**: float. Upper bound for the random saturation factor.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if upper <= lower or if lower < 0.

# tf.image.resize

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize#used_in_the_tutorials)

Resize images to size using the specified method.

### Aliases:

* tf.compat.v2.image.resize
* tf.image.resize

tf.image.resize(

    images,

    size,

    method=ResizeMethod.BILINEAR,

    preserve\_aspect\_ratio=False,

    antialias=False,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Transfer Learning Using Pretrained ConvNets](https://www.tensorflow.org/beta/tutorials/images/transfer_learning)

Resized images will be distorted if their original aspect ratio is not the same as size. To avoid distortions see [tf.image.resize\_with\_pad](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_pad).

When 'antialias' is true, the sampling filter will anti-alias the input image as well as interpolate. When downsampling an image with [anti-aliasing](https://en.wikipedia.org/wiki/Spatial_anti-aliasing) the sampling filter kernel is scaled in order to properly anti-alias the input image signal. 'antialias' has no effect when upsampling an image.

* **bilinear**: [Bilinear interpolation.](https://en.wikipedia.org/wiki/Bilinear_interpolation) If 'antialias' is true, becomes a hat/tent filter function with radius 1 when downsampling.
* **lanczos3**: [Lanczos kernel](https://en.wikipedia.org/wiki/Lanczos_resampling) with radius 3. High-quality practical filter but may have some ringing especially on synthetic images.
* **lanczos5**: [Lanczos kernel](https://en.wikipedia.org/wiki/Lanczos_resampling) with radius 5. Very-high-quality filter but may have stronger ringing.
* **bicubic**: [Cubic interpolant](https://en.wikipedia.org/wiki/Bicubic_interpolation) of Keys. Equivalent to Catmull-Rom kernel. Reasonably good quality and faster than Lanczos3Kernel, particularly when upsampling.
* **gaussian**: [Gaussian kernel](https://en.wikipedia.org/wiki/Gaussian_filter) with radius 3, sigma = 1.5 / 3.0.
* **nearest**: [Nearest neighbor interpolation.](https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation) 'antialias' has no effect when used with nearest neighbor interpolation.
* **area**: Anti-aliased resampling with area interpolation. 'antialias' has no effect when used with area interpolation; it always anti-aliases.
* **mitchellcubic**: Mitchell-Netravali Cubic non-interpolating filter. For synthetic images (especially those lacking proper prefiltering), less ringing than Keys cubic kernel but less sharp.

Note that near image edges the filtering kernel may be partially outside the image boundaries. For these pixels, only input pixels inside the image will be included in the filter sum, and the output value will be appropriately normalized.

The return value has the same type as images if method is ResizeMethod.NEAREST\_NEIGHBOR. Otherwise, the return value has type float32.

#### Args:

* **images**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **size**: A 1-D int32 Tensor of 2 elements: new\_height, new\_width. The new size for the images.
* **method**: ResizeMethod. Defaults to bilinear.
* **preserve\_aspect\_ratio**: Whether to preserve the aspect ratio. If this is set, then images will be resized to a size that fits in size while preserving the aspect ratio of the original image. Scales up the image if size is bigger than the current size of the image. Defaults to False.
* **antialias**: Whether to use an anti-aliasing filter when downsampling an image.
* **name**: A name for this operation (optional).

#### Raises:

* **ValueError**: if the shape of images is incompatible with the shape arguments to this function
* **ValueError**: if size has invalid shape or type.
* **ValueError**: if an unsupported resize method is specified.

#### Returns:

If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.image.ResizeMethod

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

## Class ResizeMethod

### Aliases:

* Class tf.compat.v2.image.ResizeMethod
* Class tf.image.ResizeMethod

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

## Class Members

* AREA = 'area'
* BICUBIC = 'bicubic'
* BILINEAR = 'bilinear'
* GAUSSIAN = 'gaussian'
* LANCZOS3 = 'lanczos3'
* LANCZOS5 = 'lanczos5'
* MITCHELLCUBIC = 'mitchellcubic'
* NEAREST\_NEIGHBOR = 'nearest'

# tf.image.resize\_with\_crop\_or\_pad

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

Crops and/or pads an image to a target width and height.

### Aliases:

* tf.compat.v1.image.resize\_image\_with\_crop\_or\_pad
* tf.compat.v1.image.resize\_with\_crop\_or\_pad
* tf.compat.v2.image.resize\_with\_crop\_or\_pad
* tf.image.resize\_with\_crop\_or\_pad

tf.image.resize\_with\_crop\_or\_pad(

    image,

    target\_height,

    target\_width

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Resizes an image to a target width and height by either centrally cropping the image or padding it evenly with zeros.

If width or height is greater than the specified target\_width or target\_height respectively, this op centrally crops along that dimension. If width or height is smaller than the specified target\_width or target\_height respectively, this op centrally pads with 0 along that dimension.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **target\_height**: Target height.
* **target\_width**: Target width.

#### Raises:

* **ValueError**: if target\_height or target\_width are zero or negative.

#### Returns:

Cropped and/or padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape[new\_height, new\_width, channels].

# tf.image.resize\_with\_pad

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

Resizes and pads an image to a target width and height.

### Aliases:

* tf.compat.v2.image.resize\_with\_pad
* tf.image.resize\_with\_pad

tf.image.resize\_with\_pad(

    image,

    target\_height,

    target\_width,

    method=ResizeMethod.BILINEAR,

    antialias=False

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Resizes an image to a target width and height by keeping the aspect ratio the same without distortion. If the target dimensions don't match the image dimensions, the image is resized and then padded with zeroes to match requested dimensions.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **target\_height**: Target height.
* **target\_width**: Target width.
* **method**: Method to use for resizing image. See image.resize()
* **antialias**: Whether to use anti-aliasing when resizing. See 'image.resize()'.

#### Raises:

* **ValueError**: if target\_height or target\_width are zero or negative.

#### Returns:

Resized and padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.image.rgb\_to\_grayscale

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

Converts one or more images from RGB to Grayscale.

### Aliases:

* tf.compat.v1.image.rgb\_to\_grayscale
* tf.compat.v2.image.rgb\_to\_grayscale
* tf.image.rgb\_to\_grayscale

tf.image.rgb\_to\_grayscale(

    images,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same DType and rank as images. The size of the last dimension of the output is 1, containing the Grayscale value of the pixels.

#### Args:

* **images**: The RGB tensor to convert. Last dimension must have size 3 and should contain RGB values.
* **name**: A name for the operation (optional).

#### Returns:

The converted grayscale image(s).

# tf.image.rgb\_to\_hsv

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

Converts one or more images from RGB to HSV.

### Aliases:

* tf.compat.v1.image.rgb\_to\_hsv
* tf.compat.v2.image.rgb\_to\_hsv
* tf.image.rgb\_to\_hsv

tf.image.rgb\_to\_hsv(

    images,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

Outputs a tensor of the same shape as the images tensor, containing the HSV value of the pixels. The output is only well defined if the value in images are in [0,1].

output[..., 0] contains hue, output[..., 1] contains saturation, and output[..., 2]contains value. All HSV values are in [0,1]. A hue of 0 corresponds to pure red, hue 1/3 is pure green, and 2/3 is pure blue.

#### Args:

* **images**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 1-D or higher rank. RGB data to convert. Last dimension must be size 3.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.rgb\_to\_yiq

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

Converts one or more images from RGB to YIQ.

### Aliases:

* tf.compat.v1.image.rgb\_to\_yiq
* tf.compat.v2.image.rgb\_to\_yiq
* tf.image.rgb\_to\_yiq

tf.image.rgb\_to\_yiq(images)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same shape as the images tensor, containing the YIQ value of the pixels. The output is only well defined if the value in images are in [0,1].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.rgb\_to\_yuv

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

Converts one or more images from RGB to YUV.

### Aliases:

* tf.compat.v1.image.rgb\_to\_yuv
* tf.compat.v2.image.rgb\_to\_yuv
* tf.image.rgb\_to\_yuv

tf.image.rgb\_to\_yuv(images)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same shape as the images tensor, containing the YUV value of the pixels. The output is only well defined if the value in images are in [0,1].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.rot90

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90#aliases)

Rotate image(s) counter-clockwise by 90 degrees.

### Aliases:

* tf.compat.v1.image.rot90
* tf.compat.v2.image.rot90
* tf.image.rot90

tf.image.rot90(

    image,

    k=1,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **k**: A scalar integer. The number of times the image is rotated by 90 degrees.
* **name**: A name for this operation (optional).

#### Returns:

A rotated tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.sample\_distorted\_bounding\_box

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

Generate a single randomly distorted bounding box for an image.

### Aliases:

* tf.compat.v2.image.sample\_distorted\_bounding\_box
* tf.image.sample\_distorted\_bounding\_box

tf.image.sample\_distorted\_bounding\_box(

    image\_size,

    bounding\_boxes,

    seed=0,

    min\_object\_covered=0.1,

    aspect\_ratio\_range=None,

    area\_range=None,

    max\_attempts=None,

    use\_image\_if\_no\_bounding\_boxes=None,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Bounding box annotations are often supplied in addition to ground-truth labels in image recognition or object localization tasks. A common technique for training such a system is to randomly distort an image while preserving its content, i.e. data augmentation. This Op outputs a randomly distorted localization of an object, i.e. bounding box, given an image\_size, bounding\_boxes and a series of constraints.

The output of this Op is a single bounding box that may be used to crop the original image. The output is returned as 3 tensors: begin, size and bboxes. The first 2 tensors can be fed directly into [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice) to crop the image. The latter may be supplied to [tf.image.draw\_bounding\_boxes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes) to visualize what the bounding box looks like.

Bounding boxes are supplied and returned as [y\_min, x\_min, y\_max, x\_max]. The bounding box coordinates are floats in [0.0, 1.0] relative to the width and height of the underlying image.

For example,

    # Generate a single distorted bounding box.

    begin, size, bbox\_for\_draw = tf.image.sample\_distorted\_bounding\_box(

        tf.shape(image),

        bounding\_boxes=bounding\_boxes,

        min\_object\_covered=0.1)

    # Draw the bounding box in an image summary.

    image\_with\_box = tf.image.draw\_bounding\_boxes(tf.expand\_dims(image, 0),

                                                  bbox\_for\_draw)

    tf.compat.v1.summary.image('images\_with\_box', image\_with\_box)

    # Employ the bounding box to distort the image.

    distorted\_image = tf.slice(image, begin, size)

Note that if no bounding box information is available, setting use\_image\_if\_no\_bounding\_boxes = true will assume there is a single implicit bounding box covering the whole image. If use\_image\_if\_no\_bounding\_boxes is false and no bounding boxes are supplied, an error is raised.

#### Args:

* **image\_size**: A Tensor. Must be one of the following types: uint8, int8, int16, int32, int64. 1-D, containing [height, width, channels].
* **bounding\_boxes**: A Tensor of type float32. 3-D with shape [batch, N, 4] describing the N bounding boxes associated with the image.
* **seed**: An optional int. Defaults to 0. If seed is set to non-zero, the random number generator is seeded by the given seed. Otherwise, it is seeded by a random seed.
* **min\_object\_covered**: A Tensor of type float32. Defaults to 0.1. The cropped area of the image must contain at least this fraction of any bounding box supplied. The value of this parameter should be non-negative. In the case of 0, the cropped area does not need to overlap any of the bounding boxes supplied.
* **aspect\_ratio\_range**: An optional list of floats. Defaults to [0.75, 1.33]. The cropped area of the image must have an aspect ratio = width / height within this range.
* **area\_range**: An optional list of floats. Defaults to [0.05, 1]. The cropped area of the image must contain a fraction of the supplied image within this range.
* **max\_attempts**: An optional int. Defaults to 100. Number of attempts at generating a cropped region of the image of the specified constraints. After max\_attempts failures, return the entire image.
* **use\_image\_if\_no\_bounding\_boxes**: An optional bool. Defaults to False. Controls behavior if no bounding boxes supplied. If true, assume an implicit bounding box covering the whole input. If false, raise an error.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (begin, size, bboxes).

* **begin**: A Tensor. Has the same type as image\_size. 1-D, containing [offset\_height, offset\_width, 0]. Provide as input to [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice).
* **size**: A Tensor. Has the same type as image\_size. 1-D, containing [target\_height, target\_width, -1]. Provide as input to [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice).
* **bboxes**: A Tensor of type float32. 3-D with shape [1, 1, 4] containing the distorted bounding box. Provide as input to [tf.image.draw\_bounding\_boxes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes).

# tf.image.sobel\_edges

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sobel_edges#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sobel_edges#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sobel_edges#used_in_the_tutorials)

Returns a tensor holding Sobel edge maps.

### Aliases:

* tf.compat.v1.image.sobel\_edges
* tf.compat.v2.image.sobel\_edges
* tf.image.sobel\_edges

tf.image.sobel\_edges(image)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

#### Arguments:

* **image**: Image tensor with shape [batch\_size, h, w, d] and type float32 or float64. The image(s) must be 2x2 or larger.

#### Returns:

Tensor holding edge maps for each channel. Returns a tensor with shape [batch\_size, h, w, d, 2] where the last two dimensions hold [[dy[0], dx[0]], [dy[1], dx[1]], ..., [dy[d-1], dx[d-1]]] calculated using the Sobel filter.

# tf.image.ssim

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

Computes SSIM index between img1 and img2.

### Aliases:

* tf.compat.v1.image.ssim
* tf.compat.v2.image.ssim
* tf.image.ssim

tf.image.ssim(

    img1,

    img2,

    max\_val,

    filter\_size=11,

    filter\_sigma=1.5,

    k1=0.01,

    k2=0.03

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This function is based on the standard SSIM implementation from: Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing.

**Note:** The true SSIM is only defined on grayscale. This function does not perform any colorspace transform. (If input is already YUV, then it will compute YUV SSIM average.)

#### Details:

* 11x11 Gaussian filter of width 1.5 is used.
* k1 = 0.01, k2 = 0.03 as in the original paper.

The image sizes must be at least 11x11 because of the filter size.

#### Example:

    # Read images from file.

    im1 = tf.decode\_png('path/to/im1.png')

    im2 = tf.decode\_png('path/to/im2.png')

    # Compute SSIM over tf.uint8 Tensors.

    ssim1 = tf.image.ssim(im1, im2, max\_val=255, filter\_size=11,

                          filter\_sigma=1.5, k1=0.01, k2=0.03)

    # Compute SSIM over tf.float32 Tensors.

    im1 = tf.image.convert\_image\_dtype(im1, tf.float32)

    im2 = tf.image.convert\_image\_dtype(im2, tf.float32)

    ssim2 = tf.image.ssim(im1, im2, max\_val=1.0, filter\_size=11,

                          filter\_sigma=1.5, k1=0.01, k2=0.03)

    # ssim1 and ssim2 both have type tf.float32 and are almost equal.

#### Args:

* **img1**: First image batch.
* **img2**: Second image batch.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **filter\_size**: Default value 11 (size of gaussian filter).
* **filter\_sigma**: Default value 1.5 (width of gaussian filter).
* **k1**: Default value 0.01
* **k2**: Default value 0.03 (SSIM is less sensitivity to K2 for lower values, so it would be better if we taken the values in range of 0< K2 <0.4).

#### Returns:

A tensor containing an SSIM value for each image in batch. Returned SSIM values are in range (-1, 1], when pixel values are non-negative. Returns a tensor with shape: broadcast(img1.shape[:-3], img2.shape[:-3]).

# tf.image.ssim\_multiscale

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

Computes the MS-SSIM between img1 and img2.

### Aliases:

* tf.compat.v1.image.ssim\_multiscale
* tf.compat.v2.image.ssim\_multiscale
* tf.image.ssim\_multiscale

tf.image.ssim\_multiscale(

    img1,

    img2,

    max\_val,

    power\_factors=\_MSSSIM\_WEIGHTS,

    filter\_size=11,

    filter\_sigma=1.5,

    k1=0.01,

    k2=0.03

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

This function assumes that img1 and img2 are image batches, i.e. the last three dimensions are [height, width, channels].

**Note:** The true SSIM is only defined on grayscale. This function does not perform any colorspace transform. (If input is already YUV, then it will compute YUV SSIM average.)

Original paper: Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik. "Multiscale structural similarity for image quality assessment." Signals, Systems and Computers, 2004.

#### Arguments:

* **img1**: First image batch.
* **img2**: Second image batch. Must have the same rank as img1.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **power\_factors**: Iterable of weights for each of the scales. The number of scales used is the length of the list. Index 0 is the unscaled resolution's weight and each increasing scale corresponds to the image being downsampled by 2. Defaults to (0.0448, 0.2856, 0.3001, 0.2363, 0.1333), which are the values obtained in the original paper.
* **filter\_size**: Default value 11 (size of gaussian filter).
* **filter\_sigma**: Default value 1.5 (width of gaussian filter).
* **k1**: Default value 0.01
* **k2**: Default value 0.03 (SSIM is less sensitivity to K2 for lower values, so it would be better if we taken the values in range of 0< K2 <0.4).

#### Returns:

A tensor containing an MS-SSIM value for each image in batch. The values are in range [0, 1]. Returns a tensor with shape: broadcast(img1.shape[:-3], img2.shape[:-3]).

# tf.image.total\_variation

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

Calculate and return the total variation for one or more images.

### Aliases:

* tf.compat.v1.image.total\_variation
* tf.compat.v2.image.total\_variation
* tf.image.total\_variation

tf.image.total\_variation(

    images,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

The total variation is the sum of the absolute differences for neighboring pixel-values in the input images. This measures how much noise is in the images.

This can be used as a loss-function during optimization so as to suppress noise in images. If you have a batch of images, then you should calculate the scalar loss-value as the sum: loss = tf.reduce\_sum(tf.image.total\_variation(images))

This implements the anisotropic 2-D version of the formula described here:

https://en.wikipedia.org/wiki/Total\_variation\_denoising

#### Args:

* **images**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: if images.shape is not a 3-D or 4-D vector.

#### Returns:

The total variation of images.

If images was 4-D, return a 1-D float Tensor of shape [batch] with the total variation for each image in the batch. If images was 3-D, return a scalar float with the total variation for that image.

# tf.image.transpose

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

Transpose image(s) by swapping the height and width dimension.

### Aliases:

* tf.compat.v1.image.transpose
* tf.compat.v1.image.transpose\_image
* tf.compat.v2.image.transpose
* tf.image.transpose

tf.image.transpose(

    image,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **name**: A name for this operation (optional).

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, width, height, channels] If image was 3-D, a 3-D float Tensor of shape [width, height, channels]

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.yiq\_to\_rgb

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

Converts one or more images from YIQ to RGB.

### Aliases:

* tf.compat.v1.image.yiq\_to\_rgb
* tf.compat.v2.image.yiq\_to\_rgb
* tf.image.yiq\_to\_rgb

tf.image.yiq\_to\_rgb(images)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the Y value in images are in [0,1], I value are in [-0.5957,0.5957] and Q value are in [-0.5226,0.5226].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.yuv\_to\_rgb

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

Converts one or more images from YUV to RGB.

### Aliases:

* tf.compat.v1.image.yuv\_to\_rgb
* tf.compat.v2.image.yuv\_to\_rgb
* tf.image.yuv\_to\_rgb

tf.image.yuv\_to\_rgb(images)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the Y value in images are in [0,1], U and V value are in [-0.5,0.5].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

Module: tf.io

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

Public API for tf.io namespace.

Modules

[gfile](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile) module: Public API for tf.io.gfile namespace.

Classes

[class FixedLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature): Configuration for parsing a fixed-length input feature.

[class FixedLenSequenceFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature): Configuration for parsing a variable-length input feature into a Tensor.

[class SparseFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature): Configuration for parsing a sparse input feature from an Example.

[class TFRecordOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions): Options used for manipulating TFRecord files.

[class TFRecordWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter): A class to write records to a TFRecords file.

[class VarLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature): Configuration for parsing a variable-length input feature.

Functions

[decode\_and\_crop\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_and_crop_jpeg): Decode and Crop a JPEG-encoded image to a uint8 tensor.

[decode\_base64(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64): Decode web-safe base64-encoded strings.

[decode\_bmp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_bmp): Decode the first frame of a BMP-encoded image to a uint8 tensor.

[decode\_compressed(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_compressed): Decompress strings.

[decode\_csv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_csv): Convert CSV records to tensors. Each column maps to one tensor.

[decode\_gif(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_gif): Decode the frame(s) of a GIF-encoded image to a uint8 tensor.

[decode\_image(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image): Function for decode\_bmp, decode\_gif, decode\_jpeg, and decode\_png.

[decode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg): Decode a JPEG-encoded image to a uint8 tensor.

[decode\_json\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_json_example): Convert JSON-encoded Example records to binary protocol buffer strings.

[decode\_png(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_png): Decode a PNG-encoded image to a uint8 or uint16 tensor.

[decode\_proto(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_proto): The op extracts fields from a serialized protocol buffers message into tensors.

[decode\_raw(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_raw): Convert raw byte strings into tensors.

[deserialize\_many\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/deserialize_many_sparse): Deserialize and concatenate SparseTensors from a serialized minibatch.

[encode\_base64(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64): Encode strings into web-safe base64 format.

[encode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_jpeg): JPEG-encode an image.

[encode\_proto(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_proto): The op serializes protobuf messages provided in the input tensors.

[extract\_jpeg\_shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/extract_jpeg_shape): Extract the shape information of a JPEG-encoded image.

[is\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/is_jpeg): Convenience function to check if the 'contents' encodes a JPEG image.

[match\_filenames\_once(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/match_filenames_once): Save the list of files matching pattern, so it is only computed once.

[matching\_files(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/matching_files): Returns the set of files matching one or more glob patterns.

[parse\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_example): Parses Example protos into a dict of tensors.

[parse\_sequence\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_sequence_example): Parses a batch of SequenceExample protos.

[parse\_single\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example): Parses a single Example proto.

[parse\_single\_sequence\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_sequence_example): Parses a single SequenceExample proto.

[parse\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor): Transforms a serialized tensorflow.TensorProto proto into a Tensor.

[read\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file): Reads and outputs the entire contents of the input filename.

[serialize\_many\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_many_sparse): Serialize N-minibatch SparseTensor into an [N, 3] Tensor.

[serialize\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_sparse): Serialize a SparseTensor into a 3-vector (1-D Tensor) object.

[serialize\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_tensor): Transforms a Tensor into a serialized TensorProto proto.

[write\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/write_file): Writes contents to the file at input filename. Creates file and recursively

[write\_graph(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/write_graph): Writes a graph proto to a file.

# tf.compat.v1.io.TFRecordCompressionType

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#top_of_page)
* [Class TFRecordCompressionType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#class_tfrecordcompressiontype)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#class_members)

## Class TFRecordCompressionType

The type of compression for the record.

### Aliases:

* Class tf.compat.v1.io.TFRecordCompressionType
* Class tf.compat.v1.python\_io.TFRecordCompressionType

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

## Class Members

* GZIP = 2
* NONE = 0
* ZLIB = 1

# tf.compat.v1.io.tf\_record\_iterator

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/tf_record_iterator#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/tf_record_iterator#aliases)

An iteraror that read the records from a TFRecords file. (deprecated)

**Aliases:**

tf.compat.v1.io.tf\_record\_iterator

tf.compat.v1.python\_io.tf\_record\_iterator

tf.compat.v1.io.tf\_record\_iterator(

    path,

    options=None

)

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use eager execution and: **tf.data.TFRecordDataset(path)**

**Args:**

**path**: The path to the TFRecords file.

**options**: (optional) A TFRecordOptions object.

**Yields:** Strings.

**Raises: IOError**: If path cannot be opened for reading.

reset\_states(states=None)

# tf.io.decode\_and\_crop\_jpeg

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

Decode and Crop a JPEG-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_and\_crop\_jpeg
* tf.compat.v1.io.decode\_and\_crop\_jpeg
* tf.compat.v2.image.decode\_and\_crop\_jpeg
* tf.compat.v2.io.decode\_and\_crop\_jpeg
* tf.image.decode\_and\_crop\_jpeg
* tf.io.decode\_and\_crop\_jpeg

tf.io.decode\_and\_crop\_jpeg(

    contents,

    crop\_window,

    channels=0,

    ratio=1,

    fancy\_upscaling=True,

    try\_recover\_truncated=False,

    acceptable\_fraction=1,

    dct\_method='',

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the JPEG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.

If needed, the JPEG-encoded image is transformed to match the requested number of color channels.

The attr ratio allows downscaling the image by an integer factor during decoding. Allowed values are: 1, 2, 4, and 8. This is much faster than downscaling the image later.

It is equivalent to a combination of decode and crop, but much faster by only decoding partial jpeg image.

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **crop\_window**: A Tensor of type int32. 1-D. The crop window: [crop\_y, crop\_x, crop\_height, crop\_width].
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **ratio**: An optional int. Defaults to 1. Downscaling ratio.
* **fancy\_upscaling**: An optional bool. Defaults to True. If true use a slower but nicer upscaling of the chroma planes (yuv420/422 only).
* **try\_recover\_truncated**: An optional bool. Defaults to False. If true try to recover an image from truncated input.
* **acceptable\_fraction**: An optional float. Defaults to 1. The minimum required fraction of lines before a truncated input is accepted.
* **dct\_method**: An optional string. Defaults to "". string specifying a hint about the algorithm used for decompression. Defaults to "" which maps to a system-specific default. Currently valid values are ["INTEGER\_FAST", "INTEGER\_ACCURATE"]. The hint may be ignored (e.g., the internal jpeg library changes to a version that does not have that specific option.)
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_base64

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64#aliases)

Decode web-safe base64-encoded strings.

### Aliases:

* tf.compat.v1.decode\_base64
* tf.compat.v1.io.decode\_base64
* tf.compat.v2.io.decode\_base64
* tf.io.decode\_base64

tf.io.decode\_base64(

    input,

    name=None

)

Defined in generated file: python/ops/gen\_string\_ops.py.

Input may or may not have padding at the end. See EncodeBase64 for padding. Web-safe means that input must use - and \_ instead of + and /.

#### Args:

* **input**: A Tensor of type string. Base64 strings to decode.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_bmp

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

Decode the first frame of a BMP-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_bmp
* tf.compat.v1.io.decode\_bmp
* tf.compat.v2.image.decode\_bmp
* tf.compat.v2.io.decode\_bmp
* tf.image.decode\_bmp
* tf.io.decode\_bmp

tf.io.decode\_bmp(

    contents,

    channels=0,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the BMP-encoded image.
* 3: output an RGB image.
* 4: output an RGBA image.

#### Args:

* **contents**: A Tensor of type string. 0-D. The BMP-encoded image.
* **channels**: An optional int. Defaults to 0.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_compressed

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

Decompress strings.

### Aliases:

* tf.compat.v1.decode\_compressed
* tf.compat.v1.io.decode\_compressed
* tf.compat.v2.io.decode\_compressed
* tf.io.decode\_compressed

tf.io.decode\_compressed(

    bytes,

    compression\_type='',

    name=None

)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

This op decompresses each element of the bytes input Tensor, which is assumed to be compressed using the given compression\_type.

The output is a string Tensor of the same shape as bytes, each element containing the decompressed data from the corresponding element in bytes.

#### Args:

* **bytes**: A Tensor of type string. A Tensor of string which is compressed.
* **compression\_type**: An optional string. Defaults to "". A scalar containing either (i) the empty string (no compression), (ii) "ZLIB", or (iii) "GZIP".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_csv

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

Convert CSV records to tensors. Each column maps to one tensor.

### Aliases:

* tf.compat.v2.io.decode\_csv
* tf.io.decode\_csv

tf.io.decode\_csv(

    records,

    record\_defaults,

    field\_delim=',',

    use\_quote\_delim=True,

    na\_value='',

    select\_cols=None,

    name=None

)

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

RFC 4180 format is expected for the CSV records. (https://tools.ietf.org/html/rfc4180) Note that we allow leading and trailing spaces with int or float field.

#### Args:

* **records**: A Tensor of type string. Each string is a record/row in the csv and all records should have the same format.
* **record\_defaults**: A list of Tensor objects with specific types. Acceptable types are float32, float64, int32, int64, string. One tensor per column of the input record, with either a scalar default value for that column or an empty vector if the column is required.
* **field\_delim**: An optional string. Defaults to ",". char delimiter to separate fields in a record.
* **use\_quote\_delim**: An optional bool. Defaults to True. If false, treats double quotation marks as regular characters inside of the string fields (ignoring RFC 4180, Section 2, Bullet 5).
* **na\_value**: Additional string to recognize as NA/NaN.
* **select\_cols**: Optional sorted list of column indices to select. If specified, only this subset of columns will be parsed and returned.
* **name**: A name for the operation (optional).

#### Returns:

A list of Tensor objects. Has the same type as record\_defaults. Each tensor will have the same shape as records.

#### Raises:

* **ValueError**: If any of the arguments is malformed.

# tf.io.decode\_gif

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

Decode the frame(s) of a GIF-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_gif
* tf.compat.v1.io.decode\_gif
* tf.compat.v2.image.decode\_gif
* tf.compat.v2.io.decode\_gif
* tf.image.decode\_gif
* tf.io.decode\_gif

tf.io.decode\_gif(

    contents,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

GIF images with frame or transparency compression are not supported. On Linux and MacOS systems, convert animated GIFs from compressed to uncompressed by running:

convert $src.gif -coalesce $dst.gif

This op also supports decoding JPEGs and PNGs, though it is cleaner to use[tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The GIF-encoded image.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_image

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#used_in_the_tutorials)

Function for decode\_bmp, decode\_gif, decode\_jpeg, and decode\_png.

### Aliases:

* tf.compat.v1.image.decode\_image
* tf.compat.v1.io.decode\_image
* tf.compat.v2.image.decode\_image
* tf.compat.v2.io.decode\_image
* tf.image.decode\_image
* tf.io.decode\_image

tf.io.decode\_image(

    contents,

    channels=None,

    dtype=tf.dtypes.uint8,

    name=None,

    expand\_animations=True

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Detects whether an image is a BMP, GIF, JPEG, or PNG, and performs the appropriate operation to convert the input bytes string into a Tensor of type dtype.

**Note:** **decode\_gif** returns a 4-D array **[num\_frames, height, width, 3]**, as opposed to **decode\_bmp**, **decode\_jpeg** and **decode\_png**, which return 3-D arrays **[height, width, num\_channels]**. Make sure to take this into account when constructing your graph if you are intermixing GIF files with BMP, JPEG, and/or PNG files. Alternately, set the **expand\_animations** argument of this function to **False**, in which case the op will return 3-dimensional tensors and will truncate animated GIF files to the first frame.

#### Args:

* **contents**: 0-D string. The encoded image bytes.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **dtype**: The desired DType of the returned Tensor.
* **name**: A name for the operation (optional)
* **expand\_animations**: Controls the shape of the returned op's output. If True, the returned op will produce a 3-D tensor for PNG, JPEG, and BMP files; and a 4-D tensor for all GIFs, whether animated or not. If, False, the returned op will produce a 3-D tensor for all file types and will truncate animated GIFs to the first frame.

#### Returns:

Tensor with type dtype and a 3- or 4-dimensional shape, depending on the file type and the value of the expand\_animations parameter.

#### Raises:

* **ValueError**: On incorrect number of channels.

# tf.io.decode\_jpeg

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#used_in_the_tutorials)

Decode a JPEG-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_jpeg
* tf.compat.v1.io.decode\_jpeg
* tf.compat.v2.image.decode\_jpeg
* tf.compat.v2.io.decode\_jpeg
* tf.image.decode\_jpeg
* tf.io.decode\_jpeg

tf.io.decode\_jpeg(

    contents,

    channels=0,

    ratio=1,

    fancy\_upscaling=True,

    try\_recover\_truncated=False,

    acceptable\_fraction=1,

    dct\_method='',

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the JPEG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.

If needed, the JPEG-encoded image is transformed to match the requested number of color channels.

The attr ratio allows downscaling the image by an integer factor during decoding. Allowed values are: 1, 2, 4, and 8. This is much faster than downscaling the image later.

This op also supports decoding PNGs and non-animated GIFs since the interface is the same, though it is cleaner to use [tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **ratio**: An optional int. Defaults to 1. Downscaling ratio.
* **fancy\_upscaling**: An optional bool. Defaults to True. If true use a slower but nicer upscaling of the chroma planes (yuv420/422 only).
* **try\_recover\_truncated**: An optional bool. Defaults to False. If true try to recover an image from truncated input.
* **acceptable\_fraction**: An optional float. Defaults to 1. The minimum required fraction of lines before a truncated input is accepted.
* **dct\_method**: An optional string. Defaults to "". string specifying a hint about the algorithm used for decompression. Defaults to "" which maps to a system-specific default. Currently valid values are ["INTEGER\_FAST", "INTEGER\_ACCURATE"]. The hint may be ignored (e.g., the internal jpeg library changes to a version that does not have that specific option.)
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_json\_example

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

Convert JSON-encoded Example records to binary protocol buffer strings.

### Aliases:

* tf.compat.v1.decode\_json\_example
* tf.compat.v1.io.decode\_json\_example
* tf.compat.v2.io.decode\_json\_example
* tf.io.decode\_json\_example

tf.io.decode\_json\_example(

    json\_examples,

    name=None

)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

This op translates a tensor containing Example records, encoded using the [standard JSON mapping](https://developers.google.com/protocol-buffers/docs/proto3#json), into a tensor containing the same records encoded as binary protocol buffers. The resulting tensor can then be fed to any of the other Example-parsing ops.

#### Args:

* **json\_examples**: A Tensor of type string. Each string is a JSON object serialized according to the JSON mapping of the Example proto.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_png

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

Decode a PNG-encoded image to a uint8 or uint16 tensor.

### Aliases:

* tf.compat.v1.image.decode\_png
* tf.compat.v1.io.decode\_png
* tf.compat.v2.image.decode\_png
* tf.compat.v2.io.decode\_png
* tf.image.decode\_png
* tf.io.decode\_png

tf.io.decode\_png(

    contents,

    channels=0,

    dtype=tf.dtypes.uint8,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the PNG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.
* 4: output an RGBA image.

If needed, the PNG-encoded image is transformed to match the requested number of color channels.

This op also supports decoding JPEGs and non-animated GIFs since the interface is the same, though it is cleaner to use [tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The PNG-encoded image.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **dtype**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.uint8, tf.uint16. Defaults to [tf.uint8](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#uint8).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type dtype.

# tf.io.decode\_proto

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

The op extracts fields from a serialized protocol buffers message into tensors.

### Aliases:

* tf.compat.v1.io.decode\_proto
* tf.compat.v2.io.decode\_proto
* tf.io.decode\_proto

tf.io.decode\_proto(

    bytes,

    message\_type,

    field\_names,

    output\_types,

    descriptor\_source='local://',

    message\_format='binary',

    sanitize=False,

    name=None

)

Defined in generated file: python/ops/gen\_decode\_proto\_ops.py.

The decode\_proto op extracts fields from a serialized protocol buffers message into tensors. The fields in field\_names are decoded and converted to the corresponding output\_types if possible.

A message\_type name must be provided to give context for the field names. The actual message descriptor can be looked up either in the linked-in descriptor pool or a filename provided by the caller using the descriptor\_source attribute.

Each output tensor is a dense tensor. This means that it is padded to hold the largest number of repeated elements seen in the input minibatch. (The shape is also padded by one to prevent zero-sized dimensions). The actual repeat counts for each example in the minibatch can be found in the sizesoutput. In many cases the output of decode\_proto is fed immediately into tf.squeeze if missing values are not a concern. When using tf.squeeze, always pass the squeeze dimension explicitly to avoid surprises.

For the most part, the mapping between Proto field types and TensorFlow dtypes is straightforward. However, there are a few special cases:

* A proto field that contains a submessage or group can only be converted to DT\_STRING (the serialized submessage). This is to reduce the complexity of the API. The resulting string can be used as input to another instance of the decode\_proto op.
* TensorFlow lacks support for unsigned integers. The ops represent uint64 types as a DT\_INT64with the same twos-complement bit pattern (the obvious way). Unsigned int32 values can be represented exactly by specifying type DT\_INT64, or using twos-complement if the caller specifies DT\_INT32 in the output\_types attribute.

The descriptor\_source attribute selects a source of protocol descriptors to consult when looking up message\_type. This may be a filename containing a serialized FileDescriptorSet message, or the special value local://, in which case only descriptors linked into the code will be searched; the filename can be on any filesystem accessible to TensorFlow.

You can build a descriptor\_source file using the --descriptor\_set\_out and --include\_imports options to the protocol compiler protoc.

The local:// database only covers descriptors linked into the code via C++ libraries, not Python imports. You can link in a proto descriptor by creating a cc\_library target with alwayslink=1.

Both binary and text proto serializations are supported, and can be chosen using the format attribute.

#### Args:

* **bytes**: A Tensor of type string. Tensor of serialized protos with shape batch\_shape.
* **message\_type**: A string. Name of the proto message type to decode.
* **field\_names**: A list of strings. List of strings containing proto field names. An extension field can be decoded by using its full name, e.g. EXT\_PACKAGE.EXT\_FIELD\_NAME.
* **output\_types**: A list of tf.DTypes. List of TF types to use for the respective field in field\_names.
* **descriptor\_source**: An optional string. Defaults to "local://". Either the special value local:// or a path to a file containing a serialized FileDescriptorSet.
* **message\_format**: An optional string. Defaults to "binary". Either binary or text.
* **sanitize**: An optional bool. Defaults to False. Whether to sanitize the result or not.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (sizes, values).

* **sizes**: A Tensor of type int32.
* **values**: A list of Tensor objects of type output\_types.

# tf.io.decode\_raw

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

Convert raw byte strings into tensors.

### Aliases:

* tf.compat.v2.io.decode\_raw
* tf.io.decode\_raw

tf.io.decode\_raw(

    input\_bytes,

    out\_type,

    little\_endian=True,

    fixed\_length=None,

    name=None

)

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

#### Args:

* **input\_bytes**: Each element of the input Tensor is converted to an array of bytes.
* **out\_type**: DType of the output. Acceptable types are half, float, double, int32, uint16, uint8, int16, int8, int64.
* **little\_endian**: Whether the input\_bytes data is in little-endian format. Data will be converted into host byte order if necessary.
* **fixed\_length**: If set, the first fixed\_length bytes of each element will be converted. Data will be zero-padded or truncated to the specified length.

fixed\_length must be a multiple of the size of out\_type. fixed\_length must be specified if the elements of input\_bytes are of variable length.

* **name**: A name for the operation (optional).

#### Returns:

A Tensor object storing the decoded bytes.

# tf.io.deserialize\_many\_sparse

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

Deserialize and concatenate SparseTensors from a serialized minibatch.

### Aliases:

* tf.compat.v1.deserialize\_many\_sparse
* tf.compat.v1.io.deserialize\_many\_sparse
* tf.compat.v2.io.deserialize\_many\_sparse
* tf.io.deserialize\_many\_sparse

tf.io.deserialize\_many\_sparse(

    serialized\_sparse,

    dtype,

    rank=None,

    name=None

)

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

The input serialized\_sparse must be a string matrix of shape [N x 3] where N is the minibatch size and the rows correspond to packed outputs of serialize\_sparse. The ranks of the original SparseTensor objects must all match. When the final SparseTensor is created, it has rank one higher than the ranks of the incoming SparseTensor objects (they have been concatenated along a new row dimension).

The output SparseTensor object's shape values for all dimensions but the first are the max across the input SparseTensor objects' shape values for the corresponding dimensions. Its first shape value is N, the minibatch size.

The input SparseTensor objects' indices are assumed ordered in standard lexicographic order. If this is not the case, after this step run sparse.reorder to restore index ordering.

For example, if the serialized input is a [2, 3] matrix representing two original SparseTensorobjects:

index = [ 0]

        [10]

        [20]

values = [1, 2, 3]

shape = [50]

and

index = [ 2]

        [10]

values = [4, 5]

shape = [30]

then the final deserialized SparseTensor will be:

index = [0  0]

        [0 10]

        [0 20]

        [1  2]

        [1 10]

values = [1, 2, 3, 4, 5]

shape = [2 50]

#### Args:

* **serialized\_sparse**: 2-D Tensor of type string of shape [N, 3]. The serialized and packed SparseTensor objects.
* **dtype**: The dtype of the serialized SparseTensor objects.
* **rank**: (optional) Python int, the rank of the SparseTensor objects.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A SparseTensor representing the deserialized SparseTensors, concatenated along the SparseTensors' first dimension.

All of the serialized SparseTensors must have had the same rank and type.

# tf.io.encode\_base64

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64#aliases)

Encode strings into web-safe base64 format.

### Aliases:

* tf.compat.v1.encode\_base64
* tf.compat.v1.io.encode\_base64
* tf.compat.v2.io.encode\_base64
* tf.io.encode\_base64

tf.io.encode\_base64(

    input,

    pad=False,

    name=None

)

Defined in generated file: python/ops/gen\_string\_ops.py.

Refer to the following article for more information on base64 format: en.wikipedia.org/wiki/Base64. Base64 strings may have padding with '=' at the end so that the encoded has length multiple of 4. See Padding section of the link above.

Web-safe means that the encoder uses - and \_ instead of + and /.

#### Args:

* **input**: A Tensor of type string. Strings to be encoded.
* **pad**: An optional bool. Defaults to False. Bool whether padding is applied at the ends.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.encode\_jpeg

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

JPEG-encode an image.

### Aliases:

* tf.compat.v1.image.encode\_jpeg
* tf.compat.v1.io.encode\_jpeg
* tf.compat.v2.image.encode\_jpeg
* tf.compat.v2.io.encode\_jpeg
* tf.image.encode\_jpeg
* tf.io.encode\_jpeg

tf.io.encode\_jpeg(

    image,

    format='',

    quality=95,

    progressive=False,

    optimize\_size=False,

    chroma\_downsampling=True,

    density\_unit='in',

    x\_density=300,

    y\_density=300,

    xmp\_metadata='',

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

image is a 3-D uint8 Tensor of shape [height, width, channels].

The attr format can be used to override the color format of the encoded output. Values can be:

* '': Use a default format based on the number of channels in the image.
* grayscale: Output a grayscale JPEG image. The channels dimension of image must be 1.
* rgb: Output an RGB JPEG image. The channels dimension of image must be 3.

If format is not specified or is the empty string, a default format is picked in function of the number of channels in image:

* 1: Output a grayscale image.
* 3: Output an RGB image.

#### Args:

* **image**: A Tensor of type uint8. 3-D with shape [height, width, channels].
* **format**: An optional string from: "", "grayscale", "rgb". Defaults to "". Per pixel image format.
* **quality**: An optional int. Defaults to 95. Quality of the compression from 0 to 100 (higher is better and slower).
* **progressive**: An optional bool. Defaults to False. If True, create a JPEG that loads progressively (coarse to fine).
* **optimize\_size**: An optional bool. Defaults to False. If True, spend CPU/RAM to reduce size with no quality change.
* **chroma\_downsampling**: An optional bool. Defaults to True. See http://en.wikipedia.org/wiki/Chroma\_subsampling.
* **density\_unit**: An optional string from: "in", "cm". Defaults to "in". Unit used to specify x\_density and y\_density: pixels per inch ('in') or centimeter ('cm').
* **x\_density**: An optional int. Defaults to 300. Horizontal pixels per density unit.
* **y\_density**: An optional int. Defaults to 300. Vertical pixels per density unit.
* **xmp\_metadata**: An optional string. Defaults to "". If not empty, embed this XMP metadata in the image header.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.encode\_proto

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

The op serializes protobuf messages provided in the input tensors.

### Aliases:

* tf.compat.v1.io.encode\_proto
* tf.compat.v2.io.encode\_proto
* tf.io.encode\_proto

tf.io.encode\_proto(

    sizes,

    values,

    field\_names,

    message\_type,

    descriptor\_source='local://',

    name=None

)

Defined in generated file: python/ops/gen\_encode\_proto\_ops.py.

The types of the tensors in values must match the schema for the fields specified in field\_names. All the tensors in values must have a common shape prefix, batch\_shape.

The sizes tensor specifies repeat counts for each field. The repeat count (last dimension) of a each tensor in values must be greater than or equal to corresponding repeat count in sizes.

A message\_type name must be provided to give context for the field names. The actual message descriptor can be looked up either in the linked-in descriptor pool or a filename provided by the caller using the descriptor\_source attribute.

The descriptor\_source attribute selects a source of protocol descriptors to consult when looking up message\_type. This may be a filename containing a serialized FileDescriptorSet message, or the special value local://, in which case only descriptors linked into the code will be searched; the filename can be on any filesystem accessible to TensorFlow.

You can build a descriptor\_source file using the --descriptor\_set\_out and --include\_imports options to the protocol compiler protoc.

The local:// database only covers descriptors linked into the code via C++ libraries, not Python imports. You can link in a proto descriptor by creating a cc\_library target with alwayslink=1.

There are a few special cases in the value mapping:

Submessage and group fields must be pre-serialized as TensorFlow strings.

TensorFlow lacks support for unsigned int64s, so they must be represented as [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) with the same twos-complement bit pattern (the obvious way).

Unsigned int32 values can be represented exactly with [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64), or with sign wrapping if the input is of type [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32).

#### Args:

* **sizes**: A Tensor of type int32. Tensor of int32 with shape [batch\_shape, len(field\_names)].
* **values**: A list of Tensor objects. List of tensors containing values for the corresponding field.
* **field\_names**: A list of strings. List of strings containing proto field names.
* **message\_type**: A string. Name of the proto message type to decode.
* **descriptor\_source**: An optional string. Defaults to "local://".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.extract\_jpeg\_shape

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

Extract the shape information of a JPEG-encoded image.

### Aliases:

* tf.compat.v1.image.extract\_jpeg\_shape
* tf.compat.v1.io.extract\_jpeg\_shape
* tf.compat.v2.image.extract\_jpeg\_shape
* tf.compat.v2.io.extract\_jpeg\_shape
* tf.image.extract\_jpeg\_shape
* tf.io.extract\_jpeg\_shape

tf.io.extract\_jpeg\_shape(

    contents,

    output\_type=tf.dtypes.int32,

    name=None

)

Defined in generated file: python/ops/gen\_image\_ops.py.

This op only parses the image header, so it is much faster than DecodeJpeg.

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **output\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.int32, tf.int64. Defaults to [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32). (Optional) The output type of the operation (int32 or int64). Defaults to int32.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

# tf.io.FixedLenFeature

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#top_of_page)
* [Class FixedLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#class_fixedlenfeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#properties)

## Class FixedLenFeature

Configuration for parsing a fixed-length input feature.

### Aliases:

* Class tf.compat.v1.FixedLenFeature
* Class tf.compat.v1.io.FixedLenFeature
* Class tf.compat.v2.io.FixedLenFeature
* Class tf.io.FixedLenFeature

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

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

To treat sparse input as dense, provide a default\_value; otherwise, the parse functions will fail on any examples missing this feature.

#### Fields:

* **shape**: Shape of input data.
* **dtype**: Data type of input.
* **default\_value**: Value to be used if an example is missing this feature. It must be compatible with dtype and of the specified shape.

## Properties

### shape

### dtype

### default\_value

# tf.io.FixedLenSequenceFeature

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#top_of_page)
* [Class FixedLenSequenceFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#class_fixedlensequencefeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#properties)
  + [shape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#shape)

## Class FixedLenSequenceFeature

Configuration for parsing a variable-length input feature into a Tensor.

### Aliases:

* Class tf.compat.v1.FixedLenSequenceFeature
* Class tf.compat.v1.io.FixedLenSequenceFeature
* Class tf.compat.v2.io.FixedLenSequenceFeature
* Class tf.io.FixedLenSequenceFeature

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

The resulting Tensor of parsing a single SequenceExample or Example has a static shape of [None] + shape and the specified dtype. The resulting Tensor of parsing a batch\_size many Examples has a static shape of [batch\_size, None] + shape and the specified dtype. The entries in the batch from different Examples will be padded with default\_value to the maximum length present in the batch.

To treat a sparse input as dense, provide allow\_missing=True; otherwise, the parse functions will fail on any examples missing this feature.

#### Fields:

* **shape**: Shape of input data for dimension 2 and higher. First dimension is of variable length None.
* **dtype**: Data type of input.
* **allow\_missing**: Whether to allow this feature to be missing from a feature list item. Is available only for parsing SequenceExample not for parsing Examples.
* **default\_value**: Scalar value to be used to pad multiple Examples to their maximum length. Irrelevant for parsing a single Example or SequenceExample. Defaults to "" for dtype string and 0 otherwise (optional).

## Properties

### shape

### dtype

### allow\_missing

### default\_value

# tf.io.is\_jpeg

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

Convenience function to check if the 'contents' encodes a JPEG image.

### Aliases:

* tf.compat.v1.image.is\_jpeg
* tf.compat.v1.io.is\_jpeg
* tf.compat.v2.image.is\_jpeg
* tf.compat.v2.io.is\_jpeg
* tf.image.is\_jpeg
* tf.io.is\_jpeg

tf.io.is\_jpeg(

    contents,

    name=None

)

Defined in [python/ops/image\_ops\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/image_ops_impl.py).

#### Args:

* **contents**: 0-D string. The encoded image bytes.
* **name**: A name for the operation (optional)

#### Returns:

A scalar boolean tensor indicating if 'contents' may be a JPEG image. is\_jpeg is susceptible to false positives.

# tf.io.matching\_files

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

Returns the set of files matching one or more glob patterns.

### Aliases:

* tf.compat.v1.io.matching\_files
* tf.compat.v1.matching\_files
* tf.compat.v2.io.matching\_files
* tf.io.matching\_files

tf.io.matching\_files(

    pattern,

    name=None

)

Defined in generated file: python/ops/gen\_io\_ops.py.

Note that this routine only supports wildcard characters in the basename portion of the pattern, not in the directory portion. Note also that the order of filenames returned can be non-deterministic.

#### Args:

* **pattern**: A Tensor of type string. Shell wildcard pattern(s). Scalar or vector of type string.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.match\_filenames\_once

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

Save the list of files matching pattern, so it is only computed once.

### Aliases:

* tf.compat.v1.io.match\_filenames\_once
* tf.compat.v1.train.match\_filenames\_once
* tf.compat.v2.io.match\_filenames\_once
* tf.io.match\_filenames\_once

tf.io.match\_filenames\_once(

    pattern,

    name=None

)

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

NOTE: The order of the files returned can be non-deterministic.

#### Args:

* **pattern**: A file pattern (glob), or 1D tensor of file patterns.
* **name**: A name for the operations (optional).

#### Returns:

A variable that is initialized to the list of files matching the pattern(s).

# tf.io.parse\_example

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

Parses Example protos into a dict of tensors.

### Aliases:

* tf.compat.v2.io.parse\_example
* tf.io.parse\_example

tf.io.parse\_example(

    serialized,

    features,

    example\_names=None,

    name=None

)

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

Parses a number of serialized [Example](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) protos given in serialized. We refer to serialized as a batch with batch\_size many entries of individual Example protos.

example\_names may contain descriptive names for the corresponding serialized protos. These may be useful for debugging purposes, but they have no effect on the output. If not None, example\_namesmust be the same length as serialized.

This op parses serialized examples into a dictionary mapping keys to Tensor and SparseTensorobjects. features is a dict from keys to VarLenFeature, SparseFeature, and FixedLenFeatureobjects. Each VarLenFeature and SparseFeature is mapped to a SparseTensor, and eachFixedLenFeature is mapped to a Tensor.

Each VarLenFeature maps to a SparseTensor of the specified type representing a ragged matrix. Its indices are [batch, index] where batch identifies the example in serialized, and index is the value's index in the list of values associated with that feature and example.

Each SparseFeature maps to a SparseTensor of the specified type representing a Tensor of dense\_shape [batch\_size] + SparseFeature.size. Its values come from the feature in the examples with key value\_key. A values[i] comes from a position k in the feature of an example at batch entry batch. This positional information is recorded in indices[i] as [batch, index\_0, index\_1, ...] where index\_j is the k-th value of the feature in the example at with key SparseFeature.index\_key[j]. In other words, we split the indices (except the first index indicating the batch entry) of a SparseTensor by dimension into different features of the Example. Due to its complexity a VarLenFeature should be preferred over a SparseFeature whenever possible.

Each FixedLenFeature df maps to a Tensor of the specified type (or [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) if not specified) and shape (serialized.size(),) + df.shape.

FixedLenFeature entries with a default\_value are optional. With no default value, we will fail if that Feature is missing from any example in serialized.

Each FixedLenSequenceFeature df maps to a Tensor of the specified type (or [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) if not specified) and shape (serialized.size(), None) + df.shape. All examples in serialized will be padded with default\_value along the second dimension.

#### Examples:

For example, if one expects a [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) VarLenFeature ft and three serialized Examples are provided:

serialized = [

  features

    { feature { key: "ft" value { float\_list { value: [1.0, 2.0] } } } },

  features

    { feature []},

  features

    { feature { key: "ft" value { float\_list { value: [3.0] } } }

]

then the output will look like:

{"ft": SparseTensor(indices=[[0, 0], [0, 1], [2, 0]],

                    values=[1.0, 2.0, 3.0],

                    dense\_shape=(3, 2)) }

If instead a FixedLenSequenceFeature with default\_value = -1.0 and shape=[] is used then the output will look like:

{"ft": [[1.0, 2.0], [3.0, -1.0]]}

Given two Example input protos in serialized:

[

  features {

    feature { key: "kw" value { bytes\_list { value: [ "knit", "big" ] } } }

    feature { key: "gps" value { float\_list { value: [] } } }

  },

  features {

    feature { key: "kw" value { bytes\_list { value: [ "emmy" ] } } }

    feature { key: "dank" value { int64\_list { value: [ 42 ] } } }

    feature { key: "gps" value { } }

  }

]

And arguments

example\_names: ["input0", "input1"],

features: {

    "kw": VarLenFeature(tf.string),

    "dank": VarLenFeature(tf.int64),

    "gps": VarLenFeature(tf.float32),

}

Then the output is a dictionary:

{

  "kw": SparseTensor(

      indices=[[0, 0], [0, 1], [1, 0]],

      values=["knit", "big", "emmy"]

      dense\_shape=[2, 2]),

  "dank": SparseTensor(

      indices=[[1, 0]],

      values=[42],

      dense\_shape=[2, 1]),

  "gps": SparseTensor(

      indices=[],

      values=[],

      dense\_shape=[2, 0]),

}

For dense results in two serialized Examples:

[

  features {

    feature { key: "age" value { int64\_list { value: [ 0 ] } } }

    feature { key: "gender" value { bytes\_list { value: [ "f" ] } } }

   },

   features {

    feature { key: "age" value { int64\_list { value: [] } } }

    feature { key: "gender" value { bytes\_list { value: [ "f" ] } } }

  }

]

#### We can use arguments:

example\_names: ["input0", "input1"],

features: {

    "age": FixedLenFeature([], dtype=tf.int64, default\_value=-1),

    "gender": FixedLenFeature([], dtype=tf.string),

}

And the expected output is:

{

  "age": [[0], [-1]],

  "gender": [["f"], ["f"]],

}

An alternative to VarLenFeature to obtain a SparseTensor is SparseFeature. For example, given two Example input protos in serialized:

[

  features {

    feature { key: "val" value { float\_list { value: [ 0.5, -1.0 ] } } }

    feature { key: "ix" value { int64\_list { value: [ 3, 20 ] } } }

  },

  features {

    feature { key: "val" value { float\_list { value: [ 0.0 ] } } }

    feature { key: "ix" value { int64\_list { value: [ 42 ] } } }

  }

]

And arguments

example\_names: ["input0", "input1"],

features: {

    "sparse": SparseFeature(

        index\_key="ix", value\_key="val", dtype=tf.float32, size=100),

}

Then the output is a dictionary:

{

  "sparse": SparseTensor(

      indices=[[0, 3], [0, 20], [1, 42]],

      values=[0.5, -1.0, 0.0]

      dense\_shape=[2, 100]),

}

#### Args:

* **serialized**: A vector (1-D Tensor) of strings, a batch of binary serialized Example protos.
* **features**: A dict mapping feature keys to FixedLenFeature, VarLenFeature, and SparseFeature values.
* **example\_names**: A vector (1-D Tensor) of strings (optional), the names of the serialized protos in the batch.
* **name**: A name for this operation (optional).

#### Returns:

A dict mapping feature keys to Tensor and SparseTensor values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_sequence\_example

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

Parses a batch of SequenceExample protos.

### Aliases:

* tf.compat.v1.io.parse\_sequence\_example
* tf.compat.v2.io.parse\_sequence\_example
* tf.io.parse\_sequence\_example

tf.io.parse\_sequence\_example(

    serialized,

    context\_features=None,

    sequence\_features=None,

    example\_names=None,

    name=None

)

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

Parses a vector of serialized [SequenceExample](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) protos given in serialized.

This op parses serialized sequence examples into a tuple of dictionaries mapping keys to Tensor and SparseTensor objects respectively. The first dictionary contains mappings for keys appearing incontext\_features, and the second dictionary contains mappings for keys appearing in sequence\_features.

At least one of context\_features and sequence\_features must be provided and non-empty.

The context\_features keys are associated with a SequenceExample as a whole, independent of time / frame. In contrast, the sequence\_features keys provide a way to access variable-length data within the FeatureList section of the SequenceExample proto. While the shapes of context\_features values are fixed with respect to frame, the frame dimension (the first dimension) of sequence\_features values may vary between SequenceExample protos, and even between feature\_list keys within the same SequenceExample.

context\_features contains VarLenFeature and FixedLenFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenFeature is mapped to a Tensor, of the specified type, shape, and default value.

sequence\_features contains VarLenFeature and FixedLenSequenceFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenSequenceFeature is mapped to a Tensor, each of the specified type. The shape will be (B,T,) + df.dense\_shape for FixedLenSequenceFeature df, where B is the batch size, and T is the length of the associatedFeatureList in the SequenceExample. For instance, FixedLenSequenceFeature([]) yields a scalar 2-D Tensor of static shape [None, None] and dynamic shape [B, T], whileFixedLenSequenceFeature([k]) (for int k >= 1) yields a 3-D matrix Tensor of static shape [None, None, k] and dynamic shape [B, T, k].

Like the input, the resulting output tensors have a batch dimension. This means that the original per-example shapes of VarLenFeatures and FixedLenSequenceFeatures can be lost. To handle that situation, this op also provides dicts of shape tensors as part of the output. There is one dict for the context features, and one for the feature\_list features. Context features of type FixedLenFeatures will not be present, since their shapes are already known by the caller. In situations where the input 'FixedLenFeature`s are of different lengths across examples, the shorter examples will be padded with default datatype values: 0 for numeric types, and the empty string for string types.

Each SparseTensor corresponding to sequence\_features represents a ragged vector. Its indices are [time, index], where time is the FeatureList entry and index is the value's index in the list of values associated with that time.

FixedLenFeature entries with a default\_value and FixedLenSequenceFeature entries with allow\_missing=True are optional; otherwise, we will fail if that Feature or FeatureList is missing from any example in serialized.

example\_name may contain a descriptive name for the corresponding serialized proto. This may be useful for debugging purposes, but it has no effect on the output. If not None, example\_name must be a scalar.

#### Args:

* **serialized**: A vector (1-D Tensor) of type string containing binary serialized SequenceExample protos.
* **context\_features**: A dict mapping feature keys to FixedLenFeature or VarLenFeaturevalues. These features are associated with a SequenceExample as a whole.
* **sequence\_features**: A dict mapping feature keys to FixedLenSequenceFeature or VarLenFeature values. These features are associated with data within the FeatureListsection of the SequenceExample proto.
* **example\_names**: A vector (1-D Tensor) of strings (optional), the name of the serialized protos.
* **name**: A name for this operation (optional).

#### Returns:

A tuple of three dicts, each mapping keys to Tensors and SparseTensors. The first dict contains the context key/values, the second dict contains the feature\_list key/values, and the final dict contains the lengths of any dense feature\_list features.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_single\_example

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#used_in_the_tutorials)

Parses a single Example proto.

### Aliases:

* tf.compat.v2.io.parse\_single\_example
* tf.io.parse\_single\_example

tf.io.parse\_single\_example(

    serialized,

    features,

    example\_names=None,

    name=None

)

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

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

Similar to parse\_example, except:

For dense tensors, the returned Tensor is identical to the output of parse\_example, except there is no batch dimension, the output shape is the same as the shape given in dense\_shape.

For SparseTensors, the first (batch) column of the indices matrix is removed (the indices matrix is a column vector), the values vector is unchanged, and the first (batch\_size) entry of the shape vector is removed (it is now a single element vector).

One might see performance advantages by batching Example protos with parse\_example instead of using this function directly.

#### Args:

* **serialized**: A scalar string Tensor, a single serialized Example. See \_parse\_single\_example\_raw documentation for more details.
* **features**: A dict mapping feature keys to FixedLenFeature or VarLenFeature values.
* **example\_names**: (Optional) A scalar string Tensor, the associated name. See \_parse\_single\_example\_raw documentation for more details.
* **name**: A name for this operation (optional).

#### Returns:

A dict mapping feature keys to Tensor and SparseTensor values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_single\_sequence\_example

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

Parses a single SequenceExample proto.

### Aliases:

* tf.compat.v1.io.parse\_single\_sequence\_example
* tf.compat.v1.parse\_single\_sequence\_example
* tf.compat.v2.io.parse\_single\_sequence\_example
* tf.io.parse\_single\_sequence\_example

tf.io.parse\_single\_sequence\_example(

    serialized,

    context\_features=None,

    sequence\_features=None,

    example\_name=None,

    name=None

)

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

Parses a single serialized [SequenceExample](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) proto given in serialized.

This op parses a serialized sequence example into a tuple of dictionaries mapping keys to Tensorand SparseTensor objects respectively. The first dictionary contains mappings for keys appearing incontext\_features, and the second dictionary contains mappings for keys appearing in sequence\_features.

At least one of context\_features and sequence\_features must be provided and non-empty.

The context\_features keys are associated with a SequenceExample as a whole, independent of time / frame. In contrast, the sequence\_features keys provide a way to access variable-length data within the FeatureList section of the SequenceExample proto. While the shapes of context\_features values are fixed with respect to frame, the frame dimension (the first dimension) of sequence\_features values may vary between SequenceExample protos, and even between feature\_list keys within the same SequenceExample.

context\_features contains VarLenFeature and FixedLenFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenFeature is mapped to a Tensor, of the specified type, shape, and default value.

sequence\_features contains VarLenFeature and FixedLenSequenceFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenSequenceFeature is mapped to a Tensor, each of the specified type. The shape will be (T,) + df.dense\_shape for FixedLenSequenceFeature df, where T is the length of the associated FeatureList in the SequenceExample. For instance, FixedLenSequenceFeature([]) yields a scalar 1-D Tensor of static shape [None] and dynamic shape [T], while FixedLenSequenceFeature([k]) (for int k >= 1) yields a 2-D matrix Tensor of static shape [None, k] and dynamic shape [T, k].

Each SparseTensor corresponding to sequence\_features represents a ragged vector. Its indices are [time, index], where time is the FeatureList entry and index is the value's index in the list of values associated with that time.

FixedLenFeature entries with a default\_value and FixedLenSequenceFeature entries with allow\_missing=True are optional; otherwise, we will fail if that Feature or FeatureList is missing from any example in serialized.

example\_name may contain a descriptive name for the corresponding serialized proto. This may be useful for debugging purposes, but it has no effect on the output. If not None, example\_name must be a scalar.

#### Args:

* **serialized**: A scalar (0-D Tensor) of type string, a single binary serialized SequenceExampleproto.
* **context\_features**: A dict mapping feature keys to FixedLenFeature or VarLenFeaturevalues. These features are associated with a SequenceExample as a whole.
* **sequence\_features**: A dict mapping feature keys to FixedLenSequenceFeature or VarLenFeature values. These features are associated with data within the FeatureListsection of the SequenceExample proto.
* **example\_name**: A scalar (0-D Tensor) of strings (optional), the name of the serialized proto.
* **name**: A name for this operation (optional).

#### Returns:

A tuple of two dicts, each mapping keys to Tensors and SparseTensors. The first dict contains the context key/values. The second dict contains the feature\_list key/values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_tensor

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#used_in_the_tutorials)

Transforms a serialized tensorflow.TensorProto proto into a Tensor.

### Aliases:

* tf.compat.v1.io.parse\_tensor
* tf.compat.v1.parse\_tensor
* tf.compat.v2.io.parse\_tensor
* tf.io.parse\_tensor

tf.io.parse\_tensor(

    serialized,

    out\_type,

    name=None

)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)

#### Args:

* **serialized**: A Tensor of type string. A scalar string containing a serialized TensorProto proto.
* **out\_type**: A [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType). The type of the serialized tensor. The provided type must match the type of the serialized tensor and no implicit conversion will take place.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type out\_type.

# tf.io.read\_file

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#used_in_the_tutorials)

Reads and outputs the entire contents of the input filename.

### Aliases:

* tf.compat.v1.io.read\_file
* tf.compat.v1.read\_file
* tf.compat.v2.io.read\_file
* tf.io.read\_file

tf.io.read\_file(

    filename,

    name=None

)

Defined in generated file: python/ops/gen\_io\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

#### Args:

* **filename**: A Tensor of type string.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.serialize\_many\_sparse

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

Serialize N-minibatch SparseTensor into an [N, 3] Tensor.

### Aliases:

* tf.compat.v2.io.serialize\_many\_sparse
* tf.io.serialize\_many\_sparse

tf.io.serialize\_many\_sparse(

    sp\_input,

    out\_type=tf.dtypes.string,

    name=None

)

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

The SparseTensor must have rank R greater than 1, and the first dimension is treated as the minibatch dimension. Elements of the SparseTensor must be sorted in increasing order of this first dimension. The serialized SparseTensor objects going into each row of the output Tensor will have rank R-1.

The minibatch size N is extracted from sparse\_shape[0].

#### Args:

* **sp\_input**: The input rank R SparseTensor.
* **out\_type**: The dtype to use for serialization.
* **name**: A name prefix for the returned tensors (optional).

#### Returns:

A matrix (2-D Tensor) with N rows and 3 columns. Each column represents serialized SparseTensor's indices, values, and shape (respectively).

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.

# tf.io.serialize\_sparse

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

Serialize a SparseTensor into a 3-vector (1-D Tensor) object.

### Aliases:

* tf.compat.v2.io.serialize\_sparse
* tf.io.serialize\_sparse

tf.io.serialize\_sparse(

    sp\_input,

    out\_type=tf.dtypes.string,

    name=None

)

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

#### Args:

* **sp\_input**: The input SparseTensor.
* **out\_type**: The dtype to use for serialization.
* **name**: A name prefix for the returned tensors (optional).

#### Returns:

A 3-vector (1-D Tensor), with each column representing the serialized SparseTensor's indices, values, and shape (respectively).

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.

# tf.io.serialize\_tensor

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

Transforms a Tensor into a serialized TensorProto proto.

### Aliases:

* tf.compat.v1.io.serialize\_tensor
* tf.compat.v1.serialize\_tensor
* tf.compat.v2.io.serialize\_tensor
* tf.io.serialize\_tensor

tf.io.serialize\_tensor(

    tensor,

    name=None

)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

#### Args:

* **tensor**: A Tensor. A Tensor of type T.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.SparseFeature

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#top_of_page)
* [Class SparseFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#class_sparsefeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#properties)
  + [index\_key](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#index_key)

## Class SparseFeature

Configuration for parsing a sparse input feature from an Example.

### Aliases:

* Class tf.compat.v1.SparseFeature
* Class tf.compat.v1.io.SparseFeature
* Class tf.compat.v2.io.SparseFeature
* Class tf.io.SparseFeature

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

Note, preferably use VarLenFeature (possibly in combination with a SequenceExample) in order to parse out SparseTensors instead of SparseFeature due to its simplicity.

Closely mimicking the SparseTensor that will be obtained by parsing an Example with a SparseFeature config, a SparseFeature contains a

* value\_key: The name of key for a Feature in the Example whose parsed Tensor will be the resulting SparseTensor.values.
* index\_key: A list of names - one for each dimension in the resulting SparseTensor whose indices[i][dim] indicating the position of the i-th value in the dim dimension will be equal to the i-th value in the Feature with key named index\_key[dim] in the Example.
* size: A list of ints for the resulting SparseTensor.dense\_shape.

For example, we can represent the following 2D SparseTensor

SparseTensor(indices=[[3, 1], [20, 0]],

             values=[0.5, -1.0]

             dense\_shape=[100, 3])

with an Example input proto

features {

  feature { key: "val" value { float\_list { value: [ 0.5, -1.0 ] } } }

  feature { key: "ix0" value { int64\_list { value: [ 3, 20 ] } } }

  feature { key: "ix1" value { int64\_list { value: [ 1, 0 ] } } }

}

and SparseFeature config with 2 index\_keys

SparseFeature(index\_key=["ix0", "ix1"],

              value\_key="val",

              dtype=tf.float32,

              size=[100, 3])

#### Fields:

* **index\_key**: A single string name or a list of string names of index features. For each key the underlying feature's type must be int64 and its length must always match that of the value\_key feature. To represent SparseTensors with a dense\_shape of rank higher than 1 a list of length rank should be used.
* **value\_key**: Name of value feature. The underlying feature's type must be dtype and its length must always match that of all the index\_keys' features.
* **dtype**: Data type of the value\_key feature.
* **size**: A Python int or list thereof specifying the dense shape. Should be a list if and only if index\_key is a list. In that case the list must be equal to the length of index\_key. Each for each entry i all values in the index\_key[i] feature must be in [0, size[i]).
* **already\_sorted**: A Python boolean to specify whether the values in value\_key are already sorted by their index position. If so skip sorting. False by default (optional).

## Properties

### index\_key

### value\_key

### dtype

### size

### already\_sorted

# tf.io.TFRecordOptions

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#top_of_page)
* [Class TFRecordOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#class_tfrecordoptions)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#methods)
  + [get\_compression\_type\_string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#get_compression_type_string)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#class_members)

## Class TFRecordOptions

Options used for manipulating TFRecord files.

### Aliases:

* Class tf.compat.v1.io.TFRecordOptions
* Class tf.compat.v1.python\_io.TFRecordOptions
* Class tf.compat.v2.io.TFRecordOptions
* Class tf.io.TFRecordOptions

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

## \_\_init\_\_

\_\_init\_\_(

    compression\_type=None,

    flush\_mode=None,

    input\_buffer\_size=None,

    output\_buffer\_size=None,

    window\_bits=None,

    compression\_level=None,

    compression\_method=None,

    mem\_level=None,

    compression\_strategy=None

)

Creates a TFRecordOptions instance.

Options only effect TFRecordWriter when compression\_type is not None. Documentation, details, and defaults can be found in [zlib\_compression\_options.h](https://www.tensorflow.org/code/tensorflow/core/lib/io/zlib_compression_options.h) and in the [zlib manual](http://www.zlib.net/manual.html). Leaving an option as None allows C++ to set a reasonable default.

#### Args:

* **compression\_type**: "GZIP", "ZLIB", or "" (no compression).
* **flush\_mode**: flush mode or None, Default: Z\_NO\_FLUSH.
* **input\_buffer\_size**: int or None.
* **output\_buffer\_size**: int or None.
* **window\_bits**: int or None.
* **compression\_level**: 0 to 9, or None.
* **compression\_method**: compression method or None.
* **mem\_level**: 1 to 9, or None.
* **compression\_strategy**: strategy or None. Default: Z\_DEFAULT\_STRATEGY.

#### Returns:

A TFRecordOptions object.

#### Raises:

* **ValueError**: If compression\_type is invalid.

## Methods

### get\_compression\_type\_string

@classmethod

get\_compression\_type\_string(

    cls,

    options

)

Convert various option types to a unified string.

#### Args:

* **options**: TFRecordOption, TFRecordCompressionType, or string.

#### Returns:

Compression type as string (e.g. 'ZLIB', 'GZIP', or '').

#### Raises:

* **ValueError**: If compression\_type is invalid.

## Class Members

* compression\_type\_map

# tf.io.TFRecordWriter

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#top_of_page)
* [Class TFRecordWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#class_tfrecordwriter)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#__init__)

## Class TFRecordWriter

A class to write records to a TFRecords file.

### Aliases:

* Class tf.compat.v1.io.TFRecordWriter
* Class tf.compat.v1.python\_io.TFRecordWriter
* Class tf.compat.v2.io.TFRecordWriter
* Class tf.io.TFRecordWriter

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

This class implements \_\_enter\_\_ and \_\_exit\_\_, and can be used in with blocks like a normal file.

## \_\_init\_\_

\_\_init\_\_(

    path,

    options=None

)

Opens file path and creates a TFRecordWriter writing to it.

#### Args:

* **path**: The path to the TFRecords file.
* **options**: (optional) String specifying compression type, TFRecordCompressionType, or TFRecordOptions object.

#### Raises:

* **IOError**: If path cannot be opened for writing.
* **ValueError**: If valid compression\_type can't be determined from options.

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

Enter a with block.

### \_\_exit\_\_

\_\_exit\_\_(

    unused\_type,

    unused\_value,

    unused\_traceback

)

Exit a with block, closing the file.

### close

close()

Close the file.

### flush

flush()

Flush the file.

### write

write(record)

Write a string record to the file.

#### Args:

* **record**: str

# tf.io.VarLenFeature

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#top_of_page)
* [Class VarLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#class_varlenfeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#properties)
  + [dtype](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#dtype)

## Class VarLenFeature

Configuration for parsing a variable-length input feature.

### Aliases:

* Class tf.compat.v1.VarLenFeature
* Class tf.compat.v1.io.VarLenFeature
* Class tf.compat.v2.io.VarLenFeature
* Class tf.io.VarLenFeature

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

#### Fields:

* **dtype**: Data type of input.

## Properties

### dtype

# tf.io.write\_file

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

Writes contents to the file at input filename. Creates file and recursively

### Aliases:

* tf.compat.v1.io.write\_file
* tf.compat.v1.write\_file
* tf.compat.v2.io.write\_file
* tf.io.write\_file

tf.io.write\_file(

    filename,

    contents,

    name=None

)

Defined in generated file: python/ops/gen\_io\_ops.py.

creates directory if not existing.

#### Args:

* **filename**: A Tensor of type string. scalar. The name of the file to which we write the contents.
* **contents**: A Tensor of type string. scalar. The content to be written to the output file.
* **name**: A name for the operation (optional).

#### Returns:

The created Operation.

# tf.io.write\_graph

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

Writes a graph proto to a file.

### Aliases:

* tf.compat.v1.io.write\_graph
* tf.compat.v1.train.write\_graph
* tf.compat.v2.io.write\_graph
* tf.io.write\_graph

tf.io.write\_graph(

    graph\_or\_graph\_def,

    logdir,

    name,

    as\_text=True

)

Defined in [python/framework/graph\_io.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/framework/graph_io.py).

The graph is written as a text proto unless as\_text is False.

v = tf.Variable(0, name='my\_variable')

sess = tf.compat.v1.Session()

tf.io.write\_graph(sess.graph\_def, '/tmp/my-model', 'train.pbtxt')

or

v = tf.Variable(0, name='my\_variable')

sess = tf.compat.v1.Session()

tf.io.write\_graph(sess.graph, '/tmp/my-model', 'train.pbtxt')

#### Args:

* **graph\_or\_graph\_def**: A Graph or a GraphDef protocol buffer.
* **logdir**: Directory where to write the graph. This can refer to remote filesystems, such as Google Cloud Storage (GCS).
* **name**: Filename for the graph.
* **as\_text**: If True, writes the graph as an ASCII proto.

#### Returns:

The path of the output proto file.

Module: tf.io.gfile

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

Public API for tf.io.gfile namespace.

Classes

[class GFile](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/GFile): File I/O wrappers without thread locking.

Functions

[copy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/copy): Copies data from src to dst.

[exists(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/exists): Determines whether a path exists or not.

[glob(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/glob): Returns a list of files that match the given pattern(s).

[isdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/isdir): Returns whether the path is a directory or not.

[listdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/listdir): Returns a list of entries contained within a directory.

[makedirs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/makedirs): Creates a directory and all parent/intermediate directories.

[mkdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/mkdir): Creates a directory with the name given by 'path'.

[remove(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/remove): Deletes the path located at 'path'.

[rename(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/rename): Rename or move a file / directory.

[rmtree(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/rmtree): Deletes everything under path recursively.

[stat(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/stat): Returns file statistics for a given path.

[walk(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/walk): Recursive directory tree generator for directories.

Module: tf.nest

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

Public API for tf.nest namespace.

Functions

[assert\_same\_structure(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nest/assert_same_structure): Asserts that two structures are nested in the same way.

[flatten(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nest/flatten): Returns a flat list from a given nested structure.

[is\_nested(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nest/is_nested): Returns true if its input is a collections.Sequence (except strings).

[map\_structure(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nest/map_structure): Applies func to each entry in structure and returns a new structure.

[pack\_sequence\_as(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nest/pack_sequence_as): Returns a given flattened sequence packed into a given structure.

# tf.nest.assert\_same\_structure

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

Asserts that two structures are nested in the same way.

### Aliases:

* tf.compat.v1.nest.assert\_same\_structure
* tf.compat.v2.nest.assert\_same\_structure
* tf.nest.assert\_same\_structure

tf.nest.assert\_same\_structure(

    nest1,

    nest2,

    check\_types=True,

    expand\_composites=False

)

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

Note that namedtuples with identical name and fields are always considered to have the same shallow structure (even with check\_types=True). For instance, this code will print True:

def nt(a, b):

  return collections.namedtuple('foo', 'a b')(a, b)

print(assert\_same\_structure(nt(0, 1), nt(2, 3)))

#### Args:

* **nest1**: an arbitrarily nested structure.
* **nest2**: an arbitrarily nested structure.
* **check\_types**: if True (default) types of sequences are checked as well, including the keys of dictionaries. If set to False, for example a list and a tuple of objects will look the same if they have the same size. Note that namedtuples with identical name and fields are always considered to have the same shallow structure. Two types will also be considered the same if they are both list subtypes (which allows "list" and "\_ListWrapper" from trackable dependency tracking to compare equal).
* **expand\_composites**: If true, then composite tensors such as [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) and [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) are expanded into their component tensors.

#### Raises:

* **ValueError**: If the two structures do not have the same number of elements or if the two structures are not nested in the same way.
* **TypeError**: If the two structures differ in the type of sequence in any of their substructures. Only possible if check\_types is True.

# tf.nest.flatten

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

Returns a flat list from a given nested structure.

### Aliases:

* tf.compat.v1.nest.flatten
* tf.compat.v2.nest.flatten
* tf.nest.flatten

tf.nest.flatten(

    structure,

    expand\_composites=False

)

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

If nest is not a sequence, tuple, or dict, then returns a single-element list: [nest].

In the case of dict instances, the sequence consists of the values, sorted by key to ensure deterministic behavior. This is true also for OrderedDict instances: their sequence order is ignored, the sorting order of keys is used instead. The same convention is followed in pack\_sequence\_as. This correctly repacks dicts and OrderedDicts after they have been flattened, and also allows flattening an OrderedDict and then repacking it back using a corresponding plain dict, or vice-versa. Dictionaries with non-sortable keys cannot be flattened.

Users must not modify any collections used in nest while this function is running.

#### Args:

* **structure**: an arbitrarily nested structure or a scalar object. Note, numpy arrays are considered scalars.
* **expand\_composites**: If true, then composite tensors such as tf.SparseTensor and tf.RaggedTensor are expanded into their component tensors.

#### Returns:

A Python list, the flattened version of the input.

#### Raises:

* **TypeError**: The nest is or contains a dict with non-sortable keys.

# tf.nest.is\_nested

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

Returns true if its input is a collections.Sequence (except strings).

### Aliases:

* tf.compat.v1.nest.is\_nested
* tf.compat.v2.nest.is\_nested
* tf.nest.is\_nested

tf.nest.is\_nested(seq)

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

#### Args:

* **seq**: an input sequence.

#### Returns:

True if the sequence is a not a string and is a collections.Sequence or a dict.

# tf.nest.map\_structure

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

Applies func to each entry in structure and returns a new structure.

### Aliases:

* tf.compat.v1.nest.map\_structure
* tf.compat.v2.nest.map\_structure
* tf.nest.map\_structure

tf.nest.map\_structure(

    func,

    \*structure,

    \*\*kwargs

)

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

Applies func(x[0], x[1], ...) where x[i] is an entry in structure[i]. All structures in structure must have the same arity, and the return value will contain results with the same structure layout.

#### Args:

* **func**: A callable that accepts as many arguments as there are structures.
* **\*structure**: scalar, or tuple or list of constructed scalars and/or other tuples/lists, or scalars. Note: numpy arrays are considered as scalars.
* **\*\*kwargs**: Valid keyword args are:
  + check\_types: If set to True (default) the types of iterables within the structures have to be same (e.g. map\_structure(func, [1], (1,)) raises a TypeError exception). To allow this set this argument to False. Note that namedtuples with identical name and fields are always considered to have the same shallow structure.
  + expand\_composites: If set to True, then composite tensors such as [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)and [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) are expanded into their component tensors. If False (the default), then composite tensors are not expanded.

#### Returns:

A new structure with the same arity as structure, whose values correspond to func(x[0], x[1], ...) where x[i] is a value in the corresponding location in structure[i]. If there are different sequence types and check\_types is False the sequence types of the first structure will be used.

#### Raises:

* **TypeError**: If func is not callable or if the structures do not match each other by depth tree.
* **ValueError**: If no structure is provided or if the structures do not match each other by type.
* **ValueError**: If wrong keyword arguments are provided.

# tf.nest.pack\_sequence\_as

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

Returns a given flattened sequence packed into a given structure.

### Aliases:

* tf.compat.v1.nest.pack\_sequence\_as
* tf.compat.v2.nest.pack\_sequence\_as
* tf.nest.pack\_sequence\_as

tf.nest.pack\_sequence\_as(

    structure,

    flat\_sequence,

    expand\_composites=False

)

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

If structure is a scalar, flat\_sequence must be a single-element list; in this case the return value is flat\_sequence[0].

If structure is or contains a dict instance, the keys will be sorted to pack the flat sequence in deterministic order. This is true also for OrderedDict instances: their sequence order is ignored, the sorting order of keys is used instead. The same convention is followed in flatten. This correctly repacks dicts and OrderedDicts after they have been flattened, and also allows flattening an OrderedDict and then repacking it back using a corresponding plain dict, or vice-versa. Dictionaries with non-sortable keys cannot be flattened.

#### Args:

* **structure**: Nested structure, whose structure is given by nested lists, tuples, and dicts. Note: numpy arrays and strings are considered scalars.
* **flat\_sequence**: flat sequence to pack.
* **expand\_composites**: If true, then composite tensors such as [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) and [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) are expanded into their component tensors.

#### Returns:

* **packed**: flat\_sequence converted to have the same recursive structure as structure.

#### Raises:

* **ValueError**: If flat\_sequence and structure have different element counts.
* **TypeError**: structure is or contains a dict with non-sortable keys.