Module: tf.compat.v1.data

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

[tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) API for input pipelines.

# See [Importing Data](https://tensorflow.org/guide/datasets) for an tf.compat.v1.data.experimental.StatsAggregator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#top_of_page)
* [Class StatsAggregator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#class_statsaggregator)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#methods)
  + [get\_summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#get_summary)

## Class StatsAggregator

A stateful resource that aggregates statistics from one or more iterators.

Defined in [python/data/experimental/ops/stats\_aggregator.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/stats_aggregator.py).

To record statistics, use one of the custom transformation functions defined in this module when defining your [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset). All statistics will be aggregated by the StatsAggregator that is associated with a particular iterator (see below). For example, to record the latency of producing each element by iterating over a dataset:

dataset = ...  
dataset = dataset.apply(tf.data.experimental.latency\_stats("total\_bytes"))

To associate a StatsAggregator with a [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) object, use the following pattern:

aggregator = tf.data.experimental.StatsAggregator()  
dataset = ...  
  
# Apply `StatsOptions` to associate `dataset` with `aggregator`.  
options = tf.data.Options()  
options.experimental\_stats.aggregator = aggregator  
dataset = dataset.with\_options(options)

To get a protocol buffer summary of the currently aggregated statistics, use the StatsAggregator.get\_summary() tensor. The easiest way to do this is to add the returned tensor to the tf.GraphKeys.SUMMARIES collection, so that the summaries will be included with any existing summaries.

aggregator = tf.data.experimental.StatsAggregator()  
# ...  
stats\_summary = aggregator.get\_summary()  
tf.compat.v1.add\_to\_collection(tf.GraphKeys.SUMMARIES, stats\_summary)

**Note:** This interface is experimental and expected to change. In particular, we expect to add other implementations of **StatsAggregator** that provide different ways of exporting statistics, and add more types of statistics.

## \_\_init\_\_

\_\_init\_\_()

Creates a StatsAggregator.

## Methods

### get\_summary

get\_summary()

Returns a string [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) that summarizes the aggregated statistics.

The returned tensor will contain a serialized [tf.compat.v1.summary.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffer, which can be used with the standard TensorBoard logging facilities.

#### Returns:

A scalar string [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) that summarizes the aggregated statistics.

overview.

Modules

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental) module: Experimental API for building input pipelines.

Classes

[class Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset): Represents a potentially large set of elements.

[class FixedLengthRecordDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset): A Dataset of fixed-length records from one or more binary files.

[class Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator): Represents the state of iterating through a Dataset.

[class Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options): Represents options for tf.data.Dataset.

[class TFRecordDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset): A Dataset comprising records from one or more TFRecord files.

[class TextLineDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset): A Dataset comprising lines from one or more text files.

Functions

[get\_output\_classes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/get_output_classes): Returns the output classes of a Dataset or Iterator.

[get\_output\_shapes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/get_output_shapes): Returns the output shapes of a Dataset or Iterator.

[get\_output\_types(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/get_output_types): Returns the output shapes of a Dataset or Iterator.

[make\_initializable\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/make_initializable_iterator): Creates a [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) for enumerating the elements of a dataset.

[make\_one\_shot\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/make_one_shot_iterator): Creates a [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) for enumerating the elements of a dataset.

# tf.compat.v1.data.Dataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset#top_of_page)
* [Class Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset#class_dataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Dataset#output_classes)

## Class Dataset

Represents a potentially large set of elements.

Inherits From: [Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset)

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

A Dataset can be used to represent an input pipeline as a collection of elements (nested structures of tensors) and a "logical plan" of transformations that act on those elements.

## \_\_init\_\_

\_\_init\_\_()

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

Creates an Iterator for enumerating the elements of this dataset.

The returned iterator implements the Python iterator protocol and therefore can only be used in eager mode.

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If not inside of tf.function and not executing eagerly.

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

@staticmethod  
from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

@staticmethod  
from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

@staticmethod  
from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

@staticmethod  
from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

@staticmethod  
list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

Returns the options for this dataset and its inputs.

#### Returns:

A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) object representing the dataset options.

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

@staticmethod  
range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

@staticmethod  
zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.FixedLengthRecordDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset#top_of_page)
* [Class FixedLengthRecordDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset#class_fixedlengthrecorddataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/FixedLengthRecordDataset#output_classes)

## Class FixedLengthRecordDataset

A Dataset of fixed-length records from one or more binary files.

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

## \_\_init\_\_

\_\_init\_\_(  
    filenames,  
    record\_bytes,  
    header\_bytes=None,  
    footer\_bytes=None,  
    buffer\_size=None,  
    compression\_type=None,  
    num\_parallel\_reads=None  
)

Creates a FixedLengthRecordDataset.

#### Args:

* **filenames**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor or [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) containing one or more filenames.
* **record\_bytes**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of bytes in each record.
* **header\_bytes**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of bytes to skip at the start of a file.
* **footer\_bytes**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of bytes to ignore at the end of a file.
* **buffer\_size**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of bytes to buffer when reading.
* **compression\_type**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".
* **num\_parallel\_reads**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of files to read in parallel. If greater than one, the records of files read in parallel are outputted in an interleaved order. If your input pipeline is I/O bottlenecked, consider setting this parameter to a value greater than one to parallelize the I/O. If None, files will be read sequentially.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.get\_output\_classes

Returns the output classes of a Dataset or Iterator.

tf.compat.v1.data.get\_output\_classes(dataset\_or\_iterator)

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

This utility method replaces the deprecated-in-V2 tf.compat.v1.Dataset.output\_classesproperty.

#### Args:

* **dataset\_or\_iterator**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset), [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator), orIteratorV2.

#### Returns:

A nested structure of Python type or [tf.data.experimental.Structure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/Structure) objects corresponding to each component of an element of this dataset.

# tf.compat.v1.data.get\_output\_shapes

Returns the output shapes of a Dataset or Iterator.

tf.compat.v1.data.get\_output\_shapes(dataset\_or\_iterator)

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

This utility method replaces the deprecated-in-V2 tf.compat.v1.Dataset.output\_shapes property.

#### Args:

* **dataset\_or\_iterator**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset), [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator), orIteratorV2.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of the given dataset or iterator.

# tf.compat.v1.data.get\_output\_types

Returns the output shapes of a Dataset or Iterator.

tf.compat.v1.data.get\_output\_types(dataset\_or\_iterator)

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

This utility method replaces the deprecated-in-V2 tf.compat.v1.Dataset.output\_types property.

#### Args:

* **dataset\_or\_iterator**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset), [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator), orIteratorV2.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

# tf.compat.v1.data.Iterator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator#top_of_page)
* [Class Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator#class_iterator)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator#properties)
  + [initializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator#initializer)

## Class Iterator

Represents the state of iterating through a Dataset.

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

## \_\_init\_\_

\_\_init\_\_(  
    iterator\_resource,  
    initializer,  
    output\_types,  
    output\_shapes,  
    output\_classes  
)

Creates a new iterator from the given iterator resource.

**Note:** Most users will not call this initializer directly, and will instead use **Dataset.make\_initializable\_iterator()** or **Dataset.make\_one\_shot\_iterator()**.

#### Args:

* **iterator\_resource**: A [tf.resource](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#resource) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) representing the iterator.
* **initializer**: A [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation) that should be run to initialize this iterator.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this iterator.
* **output\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this iterator.
* **output\_classes**: A nested structure of Python type objects corresponding to each component of an element of this iterator.

## Properties

### initializer

A [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation) that should be run to initialize this iterator.

#### Returns:

A [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation) that should be run to initialize this iterator

#### Raises:

* **ValueError**: If this iterator initializes itself automatically.

### output\_classes

Returns the class of each component of an element of this iterator. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(iterator)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this iterator. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(iterator)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this iterator. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(iterator)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### from\_string\_handle

@staticmethod  
from\_string\_handle(  
    string\_handle,  
    output\_types,  
    output\_shapes=None,  
    output\_classes=None  
)

Creates a new, uninitialized Iterator based on the given handle.

This method allows you to define a "feedable" iterator where you can choose between concrete iterators by feeding a value in a tf.Session.run call. In that case, string\_handle would be a [tf.compat.v1.placeholder](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/placeholder), and you would feed it with the value of tf.data.Iterator.string\_handle in each step.

For example, if you had two iterators that marked the current position in a training dataset and a test dataset, you could choose which to use in each step as follows:

train\_iterator = tf.data.Dataset(...).make\_one\_shot\_iterator()  
train\_iterator\_handle = sess.run(train\_iterator.string\_handle())  
  
test\_iterator = tf.data.Dataset(...).make\_one\_shot\_iterator()  
test\_iterator\_handle = sess.run(test\_iterator.string\_handle())  
  
handle = tf.compat.v1.placeholder(tf.string, shape=[])  
iterator = tf.data.Iterator.from\_string\_handle(  
    handle, train\_iterator.output\_types)  
  
next\_element = iterator.get\_next()  
loss = f(next\_element)  
  
train\_loss = sess.run(loss, feed\_dict={handle: train\_iterator\_handle})  
test\_loss = sess.run(loss, feed\_dict={handle: test\_iterator\_handle})

#### Args:

* **string\_handle**: A scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of type [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) that evaluates to a handle produced by the Iterator.string\_handle() method.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset. If omitted, each component will have an unconstrainted shape.
* **output\_classes**: (Optional.) A nested structure of Python type objects corresponding to each component of an element of this iterator. If omitted, each component is assumed to be of type [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor).

#### Returns:

An Iterator.

### from\_structure

@staticmethod  
from\_structure(  
    output\_types,  
    output\_shapes=None,  
    shared\_name=None,  
    output\_classes=None  
)

Creates a new, uninitialized Iterator with the given structure.

This iterator-constructing method can be used to create an iterator that is reusable with many different datasets.

The returned iterator is not bound to a particular dataset, and it has no initializer. To initialize the iterator, run the operation returned by Iterator.make\_initializer(dataset).

The following is an example

iterator = Iterator.from\_structure(tf.int64, tf.TensorShape([]))  
  
dataset\_range = Dataset.range(10)  
range\_initializer = iterator.make\_initializer(dataset\_range)  
  
dataset\_evens = dataset\_range.filter(lambda x: x % 2 == 0)  
evens\_initializer = iterator.make\_initializer(dataset\_evens)  
  
# Define a model based on the iterator; in this example, the model\_fn  
# is expected to take scalar tf.int64 Tensors as input (see  
# the definition of 'iterator' above).  
prediction, loss = model\_fn(iterator.get\_next())  
  
# Train for `num\_epochs`, where for each epoch, we first iterate over  
# dataset\_range, and then iterate over dataset\_evens.  
for \_ in range(num\_epochs):  
  # Initialize the iterator to `dataset\_range`  
  sess.run(range\_initializer)  
  while True:  
    try:  
      pred, loss\_val = sess.run([prediction, loss])  
    except tf.errors.OutOfRangeError:  
      break  
  
  # Initialize the iterator to `dataset\_evens`  
  sess.run(evens\_initializer)  
  while True:  
    try:  
      pred, loss\_val = sess.run([prediction, loss])  
    except tf.errors.OutOfRangeError:  
      break

#### Args:

* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset. If omitted, each component will have an unconstrainted shape.
* **shared\_name**: (Optional.) If non-empty, this iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).
* **output\_classes**: (Optional.) A nested structure of Python type objects corresponding to each component of an element of this iterator. If omitted, each component is assumed to be of type [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor).

#### Returns:

An Iterator.

#### Raises:

* **TypeError**: If the structures of output\_shapes and output\_types are not the same.

### get\_next

get\_next(name=None)

Returns a nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor)s representing the next element.

In graph mode, you should typically call this method once and use its result as the input to another computation. A typical loop will then call tf.Session.run on the result of that computation. The loop will terminate when the Iterator.get\_next() operation raises [tf.errors.OutOfRangeError](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/errors/OutOfRangeError). The following skeleton shows how to use this method when building a training loop:

dataset = ...  # A <a href="../../../../tf/data/Dataset"><code>tf.data.Dataset</code></a> object.  
iterator = dataset.make\_initializable\_iterator()  
next\_element = iterator.get\_next()  
  
# Build a TensorFlow graph that does something with each element.  
loss = model\_function(next\_element)  
optimizer = ...  # A <a href="../../../../tf/compat/v1/train/Optimizer"><code>tf.compat.v1.train.Optimizer</code></a> object.  
train\_op = optimizer.minimize(loss)  
  
with tf.compat.v1.Session() as sess:  
  try:  
    while True:  
      sess.run(train\_op)  
  except tf.errors.OutOfRangeError:  
    pass

NOTE: It is legitimate to call Iterator.get\_next() multiple times, e.g. when you are distributing different elements to multiple devices in a single step. However, a common pitfall arises when users call Iterator.get\_next() in each iteration of their training loop. Iterator.get\_next() adds ops to the graph, and executing each op allocates resources (including threads); as a consequence, invoking it in every iteration of a training loop causes slowdown and eventual resource exhaustion. To guard against this outcome, we log a warning when the number of uses crosses a fixed threshold of suspiciousness.

#### Args:

* **name**: (Optional.) A name for the created operation.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects.

### make\_initializer

make\_initializer(  
    dataset,  
    name=None  
)

Returns a [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation) that initializes this iterator on dataset.

#### Args:

* **dataset**: A Dataset with compatible structure to this iterator.
* **name**: (Optional.) A name for the created operation.

#### Returns:

A [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation) that can be run to initialize this iterator on the given dataset.

#### Raises:

* **TypeError**: If dataset and this iterator do not have a compatible element structure.

### string\_handle

string\_handle(name=None)

Returns a string-valued [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) that represents this iterator.

#### Args:

* **name**: (Optional.) A name for the created operation.

#### Returns:

A scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of type [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string).

# tf.compat.v1.data.make\_initializable\_iterator

Creates a [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) for enumerating the elements of a dataset.

tf.compat.v1.data.make\_initializable\_iterator(  
    dataset,  
    shared\_name=None  
)

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

**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = tf.compat.v1.data.make\_initializable\_iterator(dataset)  
# ...  
sess.run(iterator.initializer)

#### Args:

* **dataset**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset).
* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

A [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) over the elements of dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

# tf.compat.v1.data.make\_one\_shot\_iterator

Creates a [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) for enumerating the elements of a dataset.

tf.compat.v1.data.make\_one\_shot\_iterator(dataset)

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

**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not support re-initialization.

#### Args:

* **dataset**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset).

#### Returns:

A [tf.compat.v1.data.Iterator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/Iterator) over the elements of this dataset.

# tf.compat.v1.data.TextLineDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset#top_of_page)
* [Class TextLineDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset#class_textlinedataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TextLineDataset#output_classes)

## Class TextLineDataset

A Dataset comprising lines from one or more text files.

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

## \_\_init\_\_

\_\_init\_\_(  
    filenames,  
    compression\_type=None,  
    buffer\_size=None,  
    num\_parallel\_reads=None  
)

Creates a TextLineDataset.

#### Args:

* **filenames**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor or [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) containing one or more filenames.
* **compression\_type**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".
* **buffer\_size**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar denoting the number of bytes to buffer. A value of 0 results in the default buffering values chosen based on the compression type.
* **num\_parallel\_reads**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of files to read in parallel. If greater than one, the records of files read in parallel are outputted in an interleaved order. If your input pipeline is I/O bottlenecked, consider setting this parameter to a value greater than one to parallelize the I/O. If None, files will be read sequentially.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.TFRecordDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset#top_of_page)
* [Class TFRecordDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset#class_tfrecorddataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/TFRecordDataset#output_classes)

## Class TFRecordDataset

A Dataset comprising records from one or more TFRecord files.

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

## \_\_init\_\_

\_\_init\_\_(  
    filenames,  
    compression\_type=None,  
    buffer\_size=None,  
    num\_parallel\_reads=None  
)

Creates a TFRecordDataset to read one or more TFRecord files.

#### Args:

* **filenames**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor or [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) containing one or more filenames.
* **compression\_type**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".
* **buffer\_size**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of bytes in the read buffer. If your input pipeline is I/O bottlenecked, consider setting this parameter to a value 1-100 MBs. If None, a sensible default for both local and remote file systems is used.
* **num\_parallel\_reads**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar representing the number of files to read in parallel. If greater than one, the records of files read in parallel are outputted in an interleaved order. If your input pipeline is I/O bottlenecked, consider setting this parameter to a value greater than one to parallelize the I/O. If None, files will be read sequentially.

#### Raises:

* **TypeError**: If any argument does not have the expected type.
* **ValueError**: If any argument does not have the expected shape.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

Module: tf.compat.v1.data.experimental

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

Experimental API for building input pipelines.

This module contains experimental Dataset sources and transformations that can be used in conjunction with the [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) API. Note that the [tf.data.experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental) API is not subject to the same backwards compatibility guarantees as [tf.data](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data), but we will provide deprecation advice in advance of removing existing functionality.

See [Importing Data](https://tensorflow.org/guide/datasets) for an overview.

Classes

[class CheckpointInputPipelineHook](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/CheckpointInputPipelineHook): Checkpoints input pipeline state every N steps or seconds.

[class CsvDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset): A Dataset comprising lines from one or more CSV files.

[class DatasetStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/DatasetStructure): Represents a Dataset of structured values.

[class DistributeOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/DistributeOptions): Represents options for distributed data processing.

[class MapVectorizationOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/MapVectorizationOptions): Represents options for the MapVectorization optimization.

[class NestedStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/NestedStructure): Represents a nested structure in which each leaf is a Structure.

[class OptimizationOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/OptimizationOptions): Represents options for dataset optimizations.

[class Optional](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/Optional): Wraps a nested structure of tensors that may/may not be present at runtime.

[class OptionalStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/OptionalStructure): Represents an optional potentially containing a structured value.

[class RaggedTensorStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/RaggedTensorStructure): Represents structural information about a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor).

[class RandomDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset): A Dataset of pseudorandom values.

[class Reducer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/Reducer): A reducer is used for reducing a set of elements.

[class SparseTensorStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/SparseTensorStructure): Represents structural information about a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

[class SqlDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset): A Dataset consisting of the results from a SQL query.

[class StatsAggregator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator): A stateful resource that aggregates statistics from one or more iterators.

[class StatsOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/StatsOptions): Represents options for collecting dataset stats using StatsAggregator.

[class Structure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/Structure): Represents structural information, such as type and shape, about a value.

[class TFRecordWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/TFRecordWriter): Writes data to a TFRecord file.

[class TensorArrayStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/TensorArrayStructure): Represents structural information about a [tf.TensorArray](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorArray).

[class TensorStructure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/TensorStructure): Represents structural information about a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor).

[class ThreadingOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/ThreadingOptions): Represents options for dataset threading.

Functions

[Counter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/Counter): Creates a Dataset that counts from start in steps of size step.

[bucket\_by\_sequence\_length(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/bucket_by_sequence_length): A transformation that buckets elements in a Dataset by length.

[bytes\_produced\_stats(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/bytes_produced_stats): Records the number of bytes produced by each element of the input dataset.

[cardinality(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/cardinality): Returns the cardinality of dataset, if known.

[choose\_from\_datasets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/choose_from_datasets): Creates a dataset that deterministically chooses elements from datasets.

[copy\_to\_device(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/copy_to_device): A transformation that copies dataset elements to the given target\_device.

[dense\_to\_sparse\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch): A transformation that batches ragged elements into [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)s.

[enumerate\_dataset(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/enumerate_dataset): A transformation that enumerates the elements of a dataset. (deprecated)

[from\_variant(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/from_variant): Constructs a dataset from the given variant and structure.

[get\_next\_as\_optional(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/get_next_as_optional): Returns an Optional that contains the next value from the iterator.

[get\_single\_element(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/get_single_element): Returns the single element in dataset as a nested structure of tensors.

[get\_structure(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/get_structure): Returns the [tf.data.experimental.Structure](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/Structure) of a Dataset or Iterator.

[group\_by\_reducer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/group_by_reducer): A transformation that groups elements and performs a reduction.

[group\_by\_window(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/group_by_window): A transformation that groups windows of elements by key and reduces them.

[ignore\_errors(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/ignore_errors): Creates a Dataset from another Dataset and silently ignores any errors.

[latency\_stats(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/latency_stats): Records the latency of producing each element of the input dataset.

[make\_batched\_features\_dataset(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/make_batched_features_dataset)

[make\_csv\_dataset(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/make_csv_dataset): Reads CSV files into a dataset.

[make\_saveable\_from\_iterator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/make_saveable_from_iterator): Returns a SaveableObject for saving/restore iterator state using Saver.

[map\_and\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/map_and_batch): Fused implementation of map and batch. (deprecated)

[map\_and\_batch\_with\_legacy\_function(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/map_and_batch_with_legacy_function): Fused implementation of map and batch. (deprecated)

[parallel\_interleave(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/parallel_interleave): A parallel version of the Dataset.interleave() transformation. (deprecated)

[parse\_example\_dataset(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/parse_example_dataset): A transformation that parses Example protos into a dict of tensors.

[prefetch\_to\_device(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/prefetch_to_device): A transformation that prefetches dataset values to the given device.

[rejection\_resample(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/rejection_resample): A transformation that resamples a dataset to achieve a target distribution.

[sample\_from\_datasets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/sample_from_datasets): Samples elements at random from the datasets in datasets.

[scan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/scan): A transformation that scans a function across an input dataset.

[shuffle\_and\_repeat(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/shuffle_and_repeat): Shuffles and repeats a Dataset returning a new permutation for each epoch. (deprecated)

[take\_while(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/take_while): A transformation that stops dataset iteration based on a predicate.

[to\_variant(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/to_variant): Returns a variant representing the given dataset.

[unbatch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/unbatch): Splits elements of a dataset into multiple elements on the batch dimension. (deprecated)

[unique(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/unique): Creates a Dataset from another Dataset, discarding duplicates.

Other Members

* AUTOTUNE = -1
* INFINITE\_CARDINALITY = -1
* UNKNOWN\_CARDINALITY = -2

# tf.compat.v1.data.experimental.choose\_from\_datasets

Creates a dataset that deterministically chooses elements from datasets.

tf.compat.v1.data.experimental.choose\_from\_datasets(  
    datasets,  
    choice\_dataset  
)

Defined in [python/data/experimental/ops/interleave\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/interleave_ops.py).

For example, given the following datasets:

datasets = [tf.data.Dataset.from\_tensors("foo").repeat(),  
            tf.data.Dataset.from\_tensors("bar").repeat(),  
            tf.data.Dataset.from\_tensors("baz").repeat()]  
  
# Define a dataset containing `[0, 1, 2, 0, 1, 2, 0, 1, 2]`.  
choice\_dataset = tf.data.Dataset.range(3).repeat(3)  
  
result = tf.data.experimental.choose\_from\_datasets(datasets, choice\_dataset)

The elements of result will be:

"foo", "bar", "baz", "foo", "bar", "baz", "foo", "bar", "baz"

#### Args:

* **datasets**: A list of [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) objects with compatible structure.
* **choice\_dataset**: A [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) of scalar [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) tensors between 0 and len(datasets) - 1.

#### Returns:

A dataset that interleaves elements from datasets according to the values of choice\_dataset.

#### Raises:

* **TypeError**: If the datasets or choice\_dataset arguments have the wrong type.

# tf.compat.v1.data.experimental.Counter

Creates a Dataset that counts from start in steps of size step.

tf.compat.v1.data.experimental.Counter(  
    start=0,  
    step=1,  
    dtype=tf.dtypes.int64  
)

Defined in [python/data/experimental/ops/counter.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/counter.py).

#### For example:

Dataset.count() == [0, 1, 2, ...)  
Dataset.count(2) == [2, 3, ...)  
Dataset.count(2, 5) == [2, 7, 12, ...)  
Dataset.count(0, -1) == [0, -1, -2, ...)  
Dataset.count(10, -1) == [10, 9, ...)

#### Args:

* **start**: (Optional.) The starting value for the counter. Defaults to 0.
* **step**: (Optional.) The step size for the counter. Defaults to 1.
* **dtype**: (Optional.) The data type for counter elements. Defaults to [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).

#### Returns:

A Dataset of scalar dtype elements.

# tf.compat.v1.data.experimental.CsvDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset#top_of_page)
* [Class CsvDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset#class_csvdataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/CsvDataset#output_classes)

## Class CsvDataset

A Dataset comprising lines from one or more CSV files.

Defined in [python/data/experimental/ops/readers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/readers.py).

## \_\_init\_\_

\_\_init\_\_(  
    filenames,  
    record\_defaults,  
    compression\_type=None,  
    buffer\_size=None,  
    header=False,  
    field\_delim=',',  
    use\_quote\_delim=True,  
    na\_value='',  
    select\_cols=None  
)

Creates a CsvDataset by reading and decoding CSV files.

The elements of this dataset correspond to records from the file(s). RFC 4180 format is expected for CSV files (https://tools.ietf.org/html/rfc4180) Note that we allow leading and trailing spaces with int or float field.

For example, suppose we have a file 'my\_file0.csv' with four CSV columns of different data types:

abcdefg,4.28E10,5.55E6,12  
hijklmn,-5.3E14,,2

We can construct a CsvDataset from it as follows:

tf.compat.v1.enable\_eager\_execution()  
  
 dataset = tf.data.experimental.CsvDataset(  
    "my\_file\*.csv",  
    [tf.float32,  # Required field, use dtype or empty tensor  
     tf.constant([0.0], dtype=tf.float32),  # Optional field, default to 0.0  
     tf.int32,  # Required field, use dtype or empty tensor  
     ],  
    select\_cols=[1,2,3]  # Only parse last three columns  
)

The expected output of its iterations is:

for element in dataset:  
  print(element)  
  
>> (4.28e10, 5.55e6, 12)  
>> (-5.3e14, 0.0, 2)

#### Args:

* **filenames**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor containing one or more filenames.
* **record\_defaults**: A list of default values for the CSV fields. Each item in the list is either a valid CSV DType (float32, float64, int32, int64, string), or a Tensor object with one of the above types. One per column of CSV data, with either a scalar Tensor default value for the column if it is optional, or DType or empty Tensor if required. If both this and select\_columns are specified, these must have the same lengths, and column\_defaults is assumed to be sorted in order of increasing column index.
* **compression\_type**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP". Defaults to no compression.
* **buffer\_size**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar denoting the number of bytes to buffer while reading files. Defaults to 4MB.
* **header**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar indicating whether the CSV file(s) have header line(s) that should be skipped when parsing. Defaults to False.
* **field\_delim**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar containing the delimiter character that separates fields in a record. Defaults to ",".
* **use\_quote\_delim**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar. If False, treats double quotation marks as regular characters inside of string fields (ignoring RFC 4180, Section 2, Bullet 5). Defaults to True.
* **na\_value**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar indicating a value that will be treated as NA/NaN.
* **select\_cols**: (Optional.) A sorted list of column indices to select from the input data. If specified, only this subset of columns will be parsed. Defaults to parsing all columns.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.experimental.make\_batched\_features\_dataset

tf.compat.v1.data.experimental.make\_batched\_features\_dataset(  
    file\_pattern,  
    batch\_size,  
    features,  
    reader=tf.compat.v1.data.TFRecordDataset,  
    label\_key=None,  
    reader\_args=None,  
    num\_epochs=None,  
    shuffle=True,  
    shuffle\_buffer\_size=10000,  
    shuffle\_seed=None,  
    prefetch\_buffer\_size=dataset\_ops.AUTOTUNE,  
    reader\_num\_threads=1,  
    parser\_num\_threads=2,  
    sloppy\_ordering=False,  
    drop\_final\_batch=False  
)

Defined in [python/data/experimental/ops/readers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/readers.py).

# tf.compat.v1.data.experimental.make\_csv\_dataset

Reads CSV files into a dataset.

tf.compat.v1.data.experimental.make\_csv\_dataset(  
    file\_pattern,  
    batch\_size,  
    column\_names=None,  
    column\_defaults=None,  
    label\_name=None,  
    select\_columns=None,  
    field\_delim=',',  
    use\_quote\_delim=True,  
    na\_value='',  
    header=True,  
    num\_epochs=None,  
    shuffle=True,  
    shuffle\_buffer\_size=10000,  
    shuffle\_seed=None,  
    prefetch\_buffer\_size=dataset\_ops.AUTOTUNE,  
    num\_parallel\_reads=1,  
    sloppy=False,  
    num\_rows\_for\_inference=100,  
    compression\_type=None,  
    ignore\_errors=False  
)

Defined in [python/data/experimental/ops/readers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/readers.py).

Reads CSV files into a dataset, where each element is a (features, labels) tuple that corresponds to a batch of CSV rows. The features dictionary maps feature column names to Tensors containing the corresponding feature data, and labels is a Tensor containing the batch's label data.

#### Args:

* **file\_pattern**: List of files or patterns of file paths containing CSV records. See [tf.io.gfile.glob](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/glob) for pattern rules.
* **batch\_size**: An int representing the number of records to combine in a single batch.
* **column\_names**: An optional list of strings that corresponds to the CSV columns, in order. One per column of the input record. If this is not provided, infers the column names from the first row of the records. These names will be the keys of the features dict of each dataset element.
* **column\_defaults**: A optional list of default values for the CSV fields. One item per selected column of the input record. Each item in the list is either a valid CSV dtype (float32, float64, int32, int64, or string), or a Tensor with one of the aforementioned types. The tensor can either be a scalar default value (if the column is optional), or an empty tensor (if the column is required). If a dtype is provided instead of a tensor, the column is also treated as required. If this list is not provided, tries to infer types based on reading the first num\_rows\_for\_inference rows of files specified, and assumes all columns are optional, defaulting to 0 for numeric values and "" for string values. If both this and select\_columns are specified, these must have the same lengths, and column\_defaults is assumed to be sorted in order of increasing column index.
* **label\_name**: A optional string corresponding to the label column. If provided, the data for this column is returned as a separate Tensor from the features dictionary, so that the dataset complies with the format expected by a tf.Estimator.train or tf.Estimator.evaluateinput function.
* **select\_columns**: An optional list of integer indices or string column names, that specifies a subset of columns of CSV data to select. If column names are provided, these must correspond to names provided in column\_names or inferred from the file header lines. When this argument is specified, only a subset of CSV columns will be parsed and returned, corresponding to the columns specified. Using this results in faster parsing and lower memory usage. If both this and column\_defaults are specified, these must have the same lengths, and column\_defaults is assumed to be sorted in order of increasing column index.
* **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.
* **na\_value**: Additional string to recognize as NA/NaN.
* **header**: A bool that indicates whether the first rows of provided CSV files correspond to header lines with column names, and should not be included in the data.
* **num\_epochs**: An int specifying the number of times this dataset is repeated. If None, cycles through the dataset forever.
* **shuffle**: A bool that indicates whether the input should be shuffled.
* **shuffle\_buffer\_size**: Buffer size to use for shuffling. A large buffer size ensures better shuffling, but increases memory usage and startup time.
* **shuffle\_seed**: Randomization seed to use for shuffling.
* **prefetch\_buffer\_size**: An int specifying the number of feature batches to prefetch for performance improvement. Recommended value is the number of batches consumed per training step. Defaults to auto-tune.
* **num\_parallel\_reads**: Number of threads used to read CSV records from files. If >1, the results will be interleaved.
* **sloppy**: If True, reading performance will be improved at the cost of non-deterministic ordering. If False, the order of elements produced is deterministic prior to shuffling (elements are still randomized if shuffle=True. Note that if the seed is set, then order of elements after shuffling is deterministic). Defaults to False.
* **num\_rows\_for\_inference**: Number of rows of a file to use for type inference if record\_defaults is not provided. If None, reads all the rows of all the files. Defaults to 100.
* **compression\_type**: (Optional.) A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP". Defaults to no compression.
* **ignore\_errors**: (Optional.) If True, ignores errors with CSV file parsing, such as malformed data or empty lines, and moves on to the next valid CSV record. Otherwise, the dataset raises an error and stops processing when encountering any invalid records. Defaults to False.

#### Returns:

A dataset, where each element is a (features, labels) tuple that corresponds to a batch of batch\_sizeCSV rows. The features dictionary maps feature column names to Tensors containing the corresponding column data, and labels is a Tensor containing the column data for the label column specified by label\_name.

#### Raises:

* **ValueError**: If any of the arguments is malformed.

# tf.compat.v1.data.experimental.map\_and\_batch\_with\_legacy\_function

Fused implementation of map and batch. (deprecated)

tf.compat.v1.data.experimental.map\_and\_batch\_with\_legacy\_function(  
    map\_func,  
    batch\_size,  
    num\_parallel\_batches=None,  
    drop\_remainder=False,  
    num\_parallel\_calls=None  
)

Defined in [python/data/experimental/ops/batching.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/batching.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.experimental.map\_and\_batch()

NOTE: This is an escape hatch for existing uses of map\_and\_batch that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map\_and\_batch as this method will not be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors to another nested structure of tensors.
* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **num\_parallel\_batches**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of batches to create in parallel. On one hand, higher values can help mitigate the effect of stragglers. On the other hand, higher values can increase contention if CPU is scarce.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in case its size is smaller than desired; the default behavior is not to drop the smaller batch.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements to process in parallel. If not specified, batch\_size \* num\_parallel\_batcheselements will be processed in parallel. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

#### Raises:

* **ValueError**: If both num\_parallel\_batches and num\_parallel\_calls are specified

# tf.compat.v1.data.experimental.RandomDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset#top_of_page)
* [Class RandomDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset#class_randomdataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/RandomDataset#output_classes)

## Class RandomDataset

A Dataset of pseudorandom values.

Defined in [python/data/experimental/ops/random\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/random_ops.py).

## \_\_init\_\_

\_\_init\_\_(seed=None)

A Dataset of pseudorandom values.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.experimental.sample\_from\_datasets

Samples elements at random from the datasets in datasets.

tf.compat.v1.data.experimental.sample\_from\_datasets(  
    datasets,  
    weights=None,  
    seed=None  
)

Defined in [python/data/experimental/ops/interleave\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/interleave_ops.py).

#### Args:

* **datasets**: A list of [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) objects with compatible structure.
* **weights**: (Optional.) A list of len(datasets) floating-point values where weights[i]represents the probability with which an element should be sampled from datasets[i], or a [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) object where each element is such a list. Defaults to a uniform distribution across datasets.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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 dataset that interleaves elements from datasets at random, according to weights if provided, otherwise with uniform probability.

#### Raises:

* **TypeError**: If the datasets or weights arguments have the wrong type.
* **ValueError**: If the weights argument is specified and does not match the length of the datasets element.

# tf.compat.v1.data.experimental.SqlDataset

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset#top_of_page)
* [Class SqlDataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset#class_sqldataset)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset#properties)
  + [output\_classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/SqlDataset#output_classes)

## Class SqlDataset

A Dataset consisting of the results from a SQL query.

Defined in [python/data/experimental/ops/readers.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/readers.py).

## \_\_init\_\_

\_\_init\_\_(  
    driver\_name,  
    data\_source\_name,  
    query,  
    output\_types  
)

Creates a SqlDataset.

SqlDataset allows a user to read data from the result set of a SQL query. For example:

tf.compat.v1.enable\_eager\_execution()  
  
dataset = tf.data.experimental.SqlDataset("sqlite", "/foo/bar.sqlite3",  
                                          "SELECT name, age FROM people",  
                                          (tf.string, tf.int32))  
# Prints the rows of the result set of the above query.  
for element in dataset:  
  print(element)

#### Args:

* **driver\_name**: A 0-D [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor containing the database type. Currently, the only supported value is 'sqlite'.
* **data\_source\_name**: A 0-D [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor containing a connection string to connect to the database.
* **query**: A 0-D [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) tensor containing the SQL query to execute.
* **output\_types**: A tuple of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects representing the types of the columns returned by query.

## Properties

### output\_classes

Returns the class of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_classes(dataset)**.

The expected values are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) and [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

A nested structure of Python type objects corresponding to each component of an element of this dataset.

### output\_shapes

Returns the shape of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_shapes(dataset)**.

#### Returns:

A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element of this dataset.

### output\_types

Returns the type of each component of an element of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.compat.v1.data.get\_output\_types(dataset)**.

#### Returns:

A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element of this dataset.

## Methods

### \_\_iter\_\_

\_\_iter\_\_()

### apply

apply(transformation\_func)

Applies a transformation function to this dataset.

apply enables chaining of custom Dataset transformations, which are represented as functions that take one Dataset argument and return a transformed Dataset.

#### For example:

dataset = (dataset.map(lambda x: x \*\* 2)  
           .apply(group\_by\_window(key\_func, reduce\_func, window\_size))  
           .map(lambda x: x \*\* 3))

#### Args:

* **transformation\_func**: A function that takes one Dataset argument and returns a Dataset.

#### Returns:

* **Dataset**: The Dataset returned by applying transformation\_func to this dataset.

### batch

batch(  
    batch\_size,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into batches.

The tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainder argument to True to prevent the smaller batch from being produced.

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### cache

cache(filename='')

Caches the elements in this dataset.

#### Args:

* **filename**: A [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the name of a directory on the filesystem to use for caching tensors in this Dataset. If a filename is not provided, the dataset will be cached in memory.

#### Returns:

* **Dataset**: A Dataset.

### concatenate

concatenate(dataset)

Creates a Dataset by concatenating given dataset with this dataset.

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 8)  # ==> [ 4, 5, 6, 7 ]  
  
# Input dataset and dataset to be concatenated should have same  
# nested structures and output types.  
# c = Dataset.range(8, 14).batch(2)  # ==> [ [8, 9], [10, 11], [12, 13] ]  
# d = Dataset.from\_tensor\_slices([14.0, 15.0, 16.0])  
# a.concatenate(c) and a.concatenate(d) would result in error.  
  
a.concatenate(b)  # ==> [ 1, 2, 3, 4, 5, 6, 7 ]

#### Args:

* **dataset**: Dataset to be concatenated.

#### Returns:

* **Dataset**: A Dataset.

### enumerate

enumerate(start=0)

Enumerates the elements of this dataset.

It is similar to python's enumerate.

#### For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
a = { 1, 2, 3 }  
b = { (7, 8), (9, 10) }  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
a.enumerate(start=5)) == { (5, 1), (6, 2), (7, 3) }  
b.enumerate() == { (0, (7, 8)), (1, (9, 10)) }

#### Args:

* **start**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the start value for enumeration.

#### Returns:

* **Dataset**: A Dataset.

### filter

filter(predicate)

Filters this dataset according to predicate.

d = tf.data.Dataset.from\_tensor\_slices([1, 2, 3])  
  
d = d.filter(lambda x: x < 3)  # ==> [1, 2]  
  
# `tf.math.equal(x, y)` is required for equality comparison  
def filter\_fn(x):  
  return tf.math.equal(x, 1)  
  
d = d.filter(filter\_fn)  # ==> [1]

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### filter\_with\_legacy\_function

filter\_with\_legacy\_function(predicate)

Filters this dataset according to predicate. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.filter()

NOTE: This is an escape hatch for existing uses of filter that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to filter as this method will be removed in V2.

#### Args:

* **predicate**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a scalar [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) tensor.

#### Returns:

* **Dataset**: The Dataset containing the elements of this dataset for which predicate is True.

### flat\_map

flat\_map(map\_func)

Maps map\_func across this dataset and flattens the result.

Use flat\_map if you want to make sure that the order of your dataset stays the same. For example, to flatten a dataset of batches into a dataset of their elements:

a = Dataset.from\_tensor\_slices([ [1, 2, 3], [4, 5, 6], [7, 8, 9] ])  
  
a.flat\_map(lambda x: Dataset.from\_tensor\_slices(x + 1)) # ==>  
#  [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]

tf.data.Dataset.interleave() is a generalization of flat\_map, since flat\_map produces the same output as tf.data.Dataset.interleave(cycle\_length=1)

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.

#### Returns:

* **Dataset**: A Dataset.

### from\_generator

from\_generator(  
    generator,  
    output\_types,  
    output\_shapes=None,  
    args=None  
)

Creates a Dataset whose elements are generated by generator.

The generator argument must be a callable object that returns an object that support the iter()protocol (e.g. a generator function). The elements generated by generator must be compatible with the given output\_types and (optional) output\_shapes arguments.

#### For example:

import itertools  
tf.compat.v1.enable\_eager\_execution()  
  
def gen():  
  for i in itertools.count(1):  
    yield (i, [1] \* i)  
  
ds = tf.data.Dataset.from\_generator(  
    gen, (tf.int64, tf.int64), (tf.TensorShape([]), tf.TensorShape([None])))  
  
for value in ds.take(2):  
  print value  
# (1, array([1]))  
# (2, array([1, 1]))

NOTE: The current implementation of Dataset.from\_generator() uses [tf.compat.v1.py\_func](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/py_func)and inherits the same constraints. In particular, it requires the Dataset- and Iterator-related operations to be placed on a device in the same process as the Python program that calledDataset.from\_generator(). The body of generator will not be serialized in a GraphDef, and you should not use this method if you need to serialize your model and restore it in a different environment.

NOTE: If generator depends on mutable global variables or other external state, be aware that the runtime may invoke generator multiple times (in order to support repeating the Dataset) and at any time between the call to Dataset.from\_generator() and the production of the first element from the generator. Mutating global variables or external state can cause undefined behavior, and we recommend that you explicitly cache any external state in generator before callingDataset.from\_generator().

#### Args:

* **generator**: A callable object that returns an object that supports the iter() protocol. If argsis not specified, generator must take no arguments; otherwise it must take as many arguments as there are values in args.
* **output\_types**: A nested structure of [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) objects corresponding to each component of an element yielded by generator.
* **output\_shapes**: (Optional.) A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) objects corresponding to each component of an element yielded by generator.
* **args**: (Optional.) A tuple of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects that will be evaluated and passed to generator as NumPy-array arguments.

#### Returns:

* **Dataset**: A Dataset.

### from\_sparse\_tensor\_slices

from\_sparse\_tensor\_slices(sparse\_tensor)

Splits each rank-N [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor) in this dataset row-wise. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **tf.data.Dataset.from\_tensor\_slices()**.

#### Args:

* **sparse\_tensor**: A [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Returns:

* **Dataset**: A Dataset of rank-(N-1) sparse tensors.

### from\_tensor\_slices

from\_tensor\_slices(tensors)

Creates a Dataset whose elements are slices of the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors, each having the same size in the 0th dimension.

#### Returns:

* **Dataset**: A Dataset.

### from\_tensors

from\_tensors(tensors)

Creates a Dataset with a single element, comprising the given tensors.

Note that if tensors contains a NumPy array, and eager execution is not enabled, the values will be embedded in the graph as one or more [tf.constant](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/constant) operations. For large datasets (> 1 GB), this can waste memory and run into byte limits of graph serialization. If tensors contains one or more large NumPy arrays, consider the alternative described in [this guide](https://tensorflow.org/guide/datasets#consuming_numpy_arrays).

#### Args:

* **tensors**: A nested structure of tensors.

#### Returns:

* **Dataset**: A Dataset.

### interleave

interleave(  
    map\_func,  
    cycle\_length=AUTOTUNE,  
    block\_length=1,  
    num\_parallel\_calls=None  
)

Maps map\_func across this dataset, and interleaves the results.

For example, you can use Dataset.interleave() to process many input files concurrently:

# Preprocess 4 files concurrently, and interleave blocks of 16 records from  
# each file.  
filenames = ["/var/data/file1.txt", "/var/data/file2.txt", ...]  
dataset = (Dataset.from\_tensor\_slices(filenames)  
           .interleave(lambda x:  
               TextLineDataset(x).map(parse\_fn, num\_parallel\_calls=1),  
               cycle\_length=4, block\_length=16))

The cycle\_length and block\_length arguments control the order in which elements are produced. cycle\_length controls the number of input elements that are processed concurrently. If you set cycle\_length to 1, this transformation will handle one input element at a time, and will produce identical results to [tf.data.Dataset.flat\_map](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#flat_map). In general, this transformation will apply map\_functo cycle\_length input elements, open iterators on the returned Dataset objects, and cycle through them producing block\_length consecutive elements from each iterator, and consuming the next input element each time it reaches the end of an iterator.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
# NOTE: New lines indicate "block" boundaries.  
a.interleave(lambda x: Dataset.from\_tensors(x).repeat(6),  
            cycle\_length=2, block\_length=4)  # ==> [1, 1, 1, 1,  
                                             #      2, 2, 2, 2,  
                                             #      1, 1,  
                                             #      2, 2,  
                                             #      3, 3, 3, 3,  
                                             #      4, 4, 4, 4,  
                                             #      3, 3,  
                                             #      4, 4,  
                                             #      5, 5, 5, 5,  
                                             #      5, 5]

NOTE: The order of elements yielded by this transformation is deterministic, as long as map\_func is a pure function. If map\_func contains any stateful operations, the order in which that state is accessed is undefined.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to a Dataset.
* **cycle\_length**: (Optional.) The number of input elements that will be processed concurrently. If not specified, the value will be derived from the number of available CPU cores. If the num\_parallel\_calls argument is set to [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE), the cycle\_length argument also identifies the maximum degree of parallelism.
* **block\_length**: (Optional.) The number of consecutive elements to produce from each input element before cycling to another input element.
* **num\_parallel\_calls**: (Optional.) If specified, the implementation creates a threadpool, which is used to fetch inputs from cycle elements asynchronously and in parallel. The default behavior is to fetch inputs from cycle elements synchronously with no parallelism. If the value[tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### list\_files

list\_files(  
    file\_pattern,  
    shuffle=None,  
    seed=None  
)

A dataset of all files matching one or more glob patterns.

NOTE: The default behavior of this method is to return filenames in a non-deterministic random shuffled order. Pass a seed or shuffle=False to get results in a deterministic order.

#### Example:

If we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.py - /path/to/dir/c.py If we pass "/path/to/dir/\*.py" as the directory, the dataset would produce: - /path/to/dir/b.py - /path/to/dir/c.py

#### Args:

* **file\_pattern**: A string, a list of strings, or a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) of string type (scalar or vector), representing the filename glob (i.e. shell wildcard) pattern(s) that will be matched.
* **shuffle**: (Optional.) If True, the file names will be shuffled randomly. Defaults to True.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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:

* **Dataset**: A Dataset of strings corresponding to file names.

### make\_initializable\_iterator

make\_initializable\_iterator(shared\_name=None)

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_initializable\_iterator(dataset)**.**Note:** The returned iterator will be in an uninitialized state, and you must run the **iterator.initializer**operation before using it:

dataset = ...  
iterator = dataset.make\_initializable\_iterator()  
# ...  
sess.run(iterator.initializer)

#### Args:

* **shared\_name**: (Optional.) If non-empty, the returned iterator will be shared under the given name across multiple sessions that share the same devices (e.g. when using a remote server).

#### Returns:

An Iterator over the elements of this dataset.

#### Raises:

* **RuntimeError**: If eager execution is enabled.

### make\_one\_shot\_iterator

make\_one\_shot\_iterator()

Creates an Iterator for enumerating the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **for ... in dataset:** to iterate over a dataset. If using [**tf.estimator**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator), return the **Dataset** object directly from your input function. As a last resort, you can use **tf.compat.v1.data.make\_one\_shot\_iterator(dataset)**.**Note:** The returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

#### Returns:

An Iterator over the elements of this dataset.

### map

map(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset.

This transformation applies map\_func to each element of this dataset, and returns a new dataset containing the transformed elements, in the same order as they appeared in the input.

#### For example:

a = Dataset.range(1, 6)  # ==> [ 1, 2, 3, 4, 5 ]  
  
a.map(lambda x: x + 1)  # ==> [ 2, 3, 4, 5, 6 ]

The input signature of map\_func is determined by the structure of each element in this dataset. For example:

# NOTE: The following examples use `{ ... }` to represent the  
# contents of a dataset.  
# Each element is a `tf.Tensor` object.  
a = { 1, 2, 3, 4, 5 }  
# `map\_func` takes a single argument of type `tf.Tensor` with the same  
# shape and dtype.  
result = a.map(lambda x: ...)  
  
# Each element is a tuple containing two `tf.Tensor` objects.  
b = { (1, "foo"), (2, "bar"), (3, "baz") }  
# `map\_func` takes two arguments of type `tf.Tensor`.  
result = b.map(lambda x\_int, y\_str: ...)  
  
# Each element is a dictionary mapping strings to `tf.Tensor` objects.  
c = { {"a": 1, "b": "foo"}, {"a": 2, "b": "bar"}, {"a": 3, "b": "baz"} }  
# `map\_func` takes a single argument of type `dict` with the same keys as  
# the elements.  
result = c.map(lambda d: ...)

The value or values returned by map\_func determine the structure of each element in the returned dataset.

# `map\_func` returns a scalar `tf.Tensor` of type `tf.float32`.  
def f(...):  
  return tf.constant(37.0)  
result = dataset.map(f)  
result.output\_classes == tf.Tensor  
result.output\_types == tf.float32  
result.output\_shapes == []  # scalar  
  
# `map\_func` returns two `tf.Tensor` objects.  
def g(...):  
  return tf.constant(37.0), tf.constant(["Foo", "Bar", "Baz"])  
result = dataset.map(g)  
result.output\_classes == (tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string)  
result.output\_shapes == ([], [3])  
  
# Python primitives, lists, and NumPy arrays are implicitly converted to  
# `tf.Tensor`.  
def h(...):  
  return 37.0, ["Foo", "Bar", "Baz"], np.array([1.0, 2.0] dtype=np.float64)  
result = dataset.map(h)  
result.output\_classes == (tf.Tensor, tf.Tensor, tf.Tensor)  
result.output\_types == (tf.float32, tf.string, tf.float64)  
result.output\_shapes == ([], [3], [2])  
  
# `map\_func` can return nested structures.  
def i(...):  
  return {"a": 37.0, "b": [42, 16]}, "foo"  
result.output\_classes == ({"a": tf.Tensor, "b": tf.Tensor}, tf.Tensor)  
result.output\_types == ({"a": tf.float32, "b": tf.int32}, tf.string)  
result.output\_shapes == ({"a": [], "b": [2]}, [])

In addition to [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, map\_func can accept as arguments and return [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor)objects.

Note that irrespective of the context in which map\_func is defined (eager vs. graph), tf.data traces the function and executes it as a graph. To use Python code inside of the function you have two options:

1) Rely on AutoGraph to convert Python code into an equivalent graph computation. The downside of this approach is that AutoGraph can convert some but not all Python code.

2) Use [tf.py\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/py_function), which allows you to write arbitrary Python code but will generally result in worse performance than 1).

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### map\_with\_legacy\_function

map\_with\_legacy\_function(  
    map\_func,  
    num\_parallel\_calls=None  
)

Maps map\_func across the elements of this dataset. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use `tf.data.Dataset.map()

NOTE: This is an escape hatch for existing uses of map that do not work with V2 functions. New uses are strongly discouraged and existing uses should migrate to map as this method will be removed in V2.

#### Args:

* **map\_func**: A function mapping a nested structure of tensors (having shapes and types defined by self.output\_shapes and self.output\_types) to another nested structure of tensors.
* **num\_parallel\_calls**: (Optional.) A [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number elements to process asynchronously in parallel. If not specified, elements will be processed sequentially. If the value [tf.data.experimental.AUTOTUNE](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental#AUTOTUNE) is used, then the number of parallel calls is set dynamically based on available CPU.

#### Returns:

* **Dataset**: A Dataset.

### options

options()

### padded\_batch

padded\_batch(  
    batch\_size,  
    padded\_shapes,  
    padding\_values=None,  
    drop\_remainder=False  
)

Combines consecutive elements of this dataset into padded batches.

This transformation combines multiple consecutive elements of the input dataset into a single element.

Like [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the tensors in the resulting element will have an additional outer dimension, which will be batch\_size (or N % batch\_size for the last element if batch\_size does not divide the number of input elements N evenly and drop\_remainder is False). If your program depends on the batches having the same outer dimension, you should set the drop\_remainderargument to True to prevent the smaller batch from being produced.

Unlike [tf.data.Dataset.batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#batch), the input elements to be batched may have different shapes, and this transformation will pad each component to the respective shape in padding\_shapes. The padding\_shapes argument determines the resulting shape for each dimension of each component in an output element:

* If the dimension is a constant (e.g. tf.compat.v1.Dimension(37)), the component will be padded out to that length in that dimension.
* If the dimension is unknown (e.g. tf.compat.v1.Dimension(None)), the component will be padded out to the maximum length of all elements in that dimension.

See also [tf.data.experimental.dense\_to\_sparse\_batch](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/experimental/dense_to_sparse_batch), which combines elements that may have different shapes into a [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

#### Args:

* **batch\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of consecutive elements of this dataset to combine in a single batch.
* **padded\_shapes**: A nested structure of [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None)in a [tf.TensorShape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorShape) or -1 in a tensor-like object) will be padded to the maximum size of that dimension in each batch.
* **padding\_values**: (Optional.) A nested structure of scalar-shaped [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the padding values to use for the respective components. Defaults are 0 for numeric types and the empty string for string types.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether the last batch should be dropped in the case it has fewer than batch\_size elements; the default behavior is not to drop the smaller batch.

#### Returns:

* **Dataset**: A Dataset.

### prefetch

prefetch(buffer\_size)

Creates a Dataset that prefetches elements from this dataset.

**Note:** Like other **Dataset** methods, prefetch operates on the elements of the input dataset. It has no concept of examples vs. batches. **examples.prefetch(2)** will prefetch two elements (2 examples), while **examples.batch(20).prefetch(2)** will prefetch 2 elements (2 batches, of 20 examples each).

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the maximum number of elements that will be buffered when prefetching.

#### Returns:

* **Dataset**: A Dataset.

### range

range(\*args)

Creates a Dataset of a step-separated range of values.

#### For example:

Dataset.range(5) == [0, 1, 2, 3, 4]  
Dataset.range(2, 5) == [2, 3, 4]  
Dataset.range(1, 5, 2) == [1, 3]  
Dataset.range(1, 5, -2) == []  
Dataset.range(5, 1) == []  
Dataset.range(5, 1, -2) == [5, 3]

#### Args:

* **\*args**: follows the same semantics as python's xrange. len(args) == 1 -> start = 0, stop = args[0], step = 1 len(args) == 2 -> start = args[0], stop = args[1], step = 1 len(args) == 3 -> start = args[0], stop = args[1, stop = args[2]

#### Returns:

* **Dataset**: A RangeDataset.

#### Raises:

* **ValueError**: if len(args) == 0.

### reduce

reduce(  
    initial\_state,  
    reduce\_func  
)

Reduces the input dataset to a single element.

The transformation calls reduce\_func successively on every element of the input dataset until the dataset is exhausted, aggregating information in its internal state. The initial\_state argument is used for the initial state and the final state is returned as the result.

#### For example:

* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, \_: x + 1) produces 5
* tf.data.Dataset.range(5).reduce(np.int64(0), lambda x, y: x + y) produces 10

#### Args:

* **initial\_state**: A nested structure of tensors, representing the initial state of the transformation.
* **reduce\_func**: A function that maps (old\_state, input\_element) to new\_state. It must take two arguments and return a nested structure of tensors. The structure of new\_state must match the structure of initial\_state.

#### Returns:

A nested structure of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) objects, corresponding to the final state of the transformation.

### repeat

repeat(count=None)

Repeats this dataset count times.

NOTE: If this dataset is a function of global state (e.g. a random number generator), then different repetitions may produce different elements.

#### Args:

* **count**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of times the dataset should be repeated. The default behavior (if count is None or -1) is for the dataset be repeated indefinitely.

#### Returns:

* **Dataset**: A Dataset.

### shard

shard(  
    num\_shards,  
    index  
)

Creates a Dataset that includes only 1/num\_shards of this dataset.

This dataset operator is very useful when running distributed training, as it allows each worker to read a unique subset.

When reading a single input file, you can skip elements as follows:

d = tf.data.TFRecordDataset(input\_file)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Important caveats:

* Be sure to shard before you use any randomizing operator (such as shuffle).
* Generally it is best if the shard operator is used early in the dataset pipeline. For example, when reading from a set of TFRecord files, shard before converting the dataset to input samples. This avoids reading every file on every worker. The following is an example of an efficient sharding strategy within a complete pipeline:

d = Dataset.list\_files(pattern)  
d = d.shard(num\_workers, worker\_index)  
d = d.repeat(num\_epochs)  
d = d.shuffle(shuffle\_buffer\_size)  
d = d.interleave(tf.data.TFRecordDataset,  
                 cycle\_length=num\_readers, block\_length=1)  
d = d.map(parser\_fn, num\_parallel\_calls=num\_map\_threads)

#### Args:

* **num\_shards**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of shards operating in parallel.
* **index**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the worker index.

#### Returns:

* **Dataset**: A Dataset.

#### Raises:

* **InvalidArgumentError**: if num\_shards or index are illegal values. Note: error checking is done on a best-effort basis, and errors aren't guaranteed to be caught upon dataset creation. (e.g. providing in a placeholder tensor bypasses the early checking, and will instead result in an error during a session.run call.)

### shuffle

shuffle(  
    buffer\_size,  
    seed=None,  
    reshuffle\_each\_iteration=None  
)

Randomly shuffles the elements of this dataset.

This dataset fills a buffer with buffer\_size elements, then randomly samples elements from this buffer, replacing the selected elements with new elements. For perfect shuffling, a buffer size greater than or equal to the full size of the dataset is required.

For instance, if your dataset contains 10,000 elements but buffer\_size is set to 1,000, then shuffle will initially select a random element from only the first 1,000 elements in the buffer. Once an element is selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element, maintaining the 1,000 element buffer.

#### Args:

* **buffer\_size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements from this dataset from which the new dataset will sample.
* **seed**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the random seed that will be used to create the distribution. 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.
* **reshuffle\_each\_iteration**: (Optional.) A boolean, which if true indicates that the dataset should be pseudorandomly reshuffled each time it is iterated over. (Defaults to True.)

#### Returns:

* **Dataset**: A Dataset.

### skip

skip(count)

Creates a Dataset that skips count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be skipped to form the new dataset. If count is greater than the size of this dataset, the new dataset will contain no elements. If count is -1, skips the entire dataset.

#### Returns:

* **Dataset**: A Dataset.

### take

take(count)

Creates a Dataset with at most count elements from this dataset.

#### Args:

* **count**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of this dataset that should be taken to form the new dataset. If count is -1, or if count is greater than the size of this dataset, the new dataset will contain all elements of this dataset.

#### Returns:

* **Dataset**: A Dataset.

### unbatch

unbatch()

Splits elements of a dataset into multiple elements.

For example, if elements of the dataset are shaped [B, a0, a1, ...], where B may vary for each input element, then for each element in the dataset, the unbatched dataset will contain B consecutive elements of shape [a0, a1, ...].

# NOTE: The following example uses `{ ... }` to represent the contents  
# of a dataset.  
ds = { ['a', 'b', 'c'], ['a', 'b'], ['a', 'b', 'c', 'd'] }  
  
ds.unbatch() == {'a', 'b', 'c', 'a', 'b', 'a', 'b', 'c', 'd'}

#### Returns:

A Dataset transformation function, which can be passed to [tf.data.Dataset.apply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset#apply).

### window

window(  
    size,  
    shift=None,  
    stride=1,  
    drop\_remainder=False  
)

Combines (nests of) input elements into a dataset of (nests of) windows.

A "window" is a finite dataset of flat elements of size size (or possibly fewer if there are not enough input elements to fill the window and drop\_remainder evaluates to false).

The stride argument determines the stride of the input elements, and the shift argument determines the shift of the window.

For example, letting {...} to represent a Dataset:

* tf.data.Dataset.range(7).window(2) produces { {0, 1}, {2, 3}, {4, 5}, {6}}
* tf.data.Dataset.range(7).window(3, 2, 1, True) produces { {0, 1, 2}, {2, 3, 4}, {4, 5, 6}}
* tf.data.Dataset.range(7).window(3, 1, 2, True) produces { {0, 2, 4}, {1, 3, 5}, {2, 4, 6}}

Note that when the window transformation is applied to a dataset of nested elements, it produces a dataset of nested windows.

#### For example:

* tf.data.Dataset.from\_tensor\_slices((range(4), range(4)).window(2) produces {({0, 1}, {0, 1}), ({2, 3}, {2, 3})}
* tf.data.Dataset.from\_tensor\_slices({"a": range(4)}).window(2) produces { {"a": {0, 1}}, {"a": {2, 3}}}

#### Args:

* **size**: A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the number of elements of the input dataset to combine into a window.
* **shift**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the forward shift of the sliding window in each iteration. Defaults to size.
* **stride**: (Optional.) A [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing the stride of the input elements in the sliding window.
* **drop\_remainder**: (Optional.) A [tf.bool](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#bool) scalar [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor), representing whether a window should be dropped in case its size is smaller than window\_size.

#### Returns:

* **Dataset**: A Dataset of (nests of) windows -- a finite datasets of flat elements created from the (nests of) input elements.

### with\_options

with\_options(options)

Returns a new [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) with the given options set.

The options are "global" in the sense they apply to the entire dataset. If options are set multiple times, they are merged as long as different options do not use different non-default values.

#### Args:

* **options**: A [tf.data.Options](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Options) that identifies the options the use.

#### Returns:

* **Dataset**: A Dataset with the given options.

#### Raises:

* **ValueError**: when an option is set more than once to a non-default value

### zip

zip(datasets)

Creates a Dataset by zipping together the given datasets.

This method has similar semantics to the built-in zip() function in Python, with the main difference being that the datasets argument can be an arbitrary nested structure of Dataset objects. For example:

a = Dataset.range(1, 4)  # ==> [ 1, 2, 3 ]  
b = Dataset.range(4, 7)  # ==> [ 4, 5, 6 ]  
c = Dataset.range(7, 13).batch(2)  # ==> [ [7, 8], [9, 10], [11, 12] ]  
d = Dataset.range(13, 15)  # ==> [ 13, 14 ]  
  
# The nested structure of the `datasets` argument determines the  
# structure of elements in the resulting dataset.  
Dataset.zip((a, b))  # ==> [ (1, 4), (2, 5), (3, 6) ]  
Dataset.zip((b, a))  # ==> [ (4, 1), (5, 2), (6, 3) ]  
  
# The `datasets` argument may contain an arbitrary number of  
# datasets.  
Dataset.zip((a, b, c))  # ==> [ (1, 4, [7, 8]),  
                        #       (2, 5, [9, 10]),  
                        #       (3, 6, [11, 12]) ]  
  
# The number of elements in the resulting dataset is the same as  
# the size of the smallest dataset in `datasets`.  
Dataset.zip((a, d))  # ==> [ (1, 13), (2, 14) ]

#### Args:

* **datasets**: A nested structure of datasets.

#### Returns:

* **Dataset**: A Dataset.

# tf.compat.v1.data.experimental.StatsAggregator

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#top_of_page)
* [Class StatsAggregator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#class_statsaggregator)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#methods)
  + [get\_summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/data/experimental/StatsAggregator#get_summary)

## Class StatsAggregator

A stateful resource that aggregates statistics from one or more iterators.

Defined in [python/data/experimental/ops/stats\_aggregator.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/data/experimental/ops/stats_aggregator.py).

To record statistics, use one of the custom transformation functions defined in this module when defining your [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset). All statistics will be aggregated by the StatsAggregator that is associated with a particular iterator (see below). For example, to record the latency of producing each element by iterating over a dataset:

dataset = ...  
dataset = dataset.apply(tf.data.experimental.latency\_stats("total\_bytes"))

To associate a StatsAggregator with a [tf.data.Dataset](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/data/Dataset) object, use the following pattern:

aggregator = tf.data.experimental.StatsAggregator()  
dataset = ...  
  
# Apply `StatsOptions` to associate `dataset` with `aggregator`.  
options = tf.data.Options()  
options.experimental\_stats.aggregator = aggregator  
dataset = dataset.with\_options(options)

To get a protocol buffer summary of the currently aggregated statistics, use the StatsAggregator.get\_summary() tensor. The easiest way to do this is to add the returned tensor to the tf.GraphKeys.SUMMARIES collection, so that the summaries will be included with any existing summaries.

aggregator = tf.data.experimental.StatsAggregator()  
# ...  
stats\_summary = aggregator.get\_summary()  
tf.compat.v1.add\_to\_collection(tf.GraphKeys.SUMMARIES, stats\_summary)

**Note:** This interface is experimental and expected to change. In particular, we expect to add other implementations of **StatsAggregator** that provide different ways of exporting statistics, and add more types of statistics.

## \_\_init\_\_

\_\_init\_\_()

Creates a StatsAggregator.

## Methods

### get\_summary

get\_summary()

Returns a string [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) that summarizes the aggregated statistics.

The returned tensor will contain a serialized [tf.compat.v1.summary.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffer, which can be used with the standard TensorBoard logging facilities.

#### Returns:

A scalar string [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) that summarizes the aggregated statistics.