Package 'tfdatasets'

August 25, 2018

Type Package

Title Interface to 'TensorFlow' Datasets

Version 1.9

Description Interface to 'TensorFlow' Datasets, a high-level library for building complex input pipelines from simple, re-usable pieces. See <https://www.tensorflow.org/programmers_guide/datasets> for additional details.

License Apache License 2.0

URL https://github.com/rstudio/tfdatasets

BugReports https://github.com/rstudio/tfdatasets/issues

SystemRequirements TensorFlow >= 1.4 (https://www.tensorflow.org/)

Encoding UTF-8

LazyData true

Depends R (>= 3.1)

Imports reticulate (>= 1.10), tensorflow (>= 1.9), magrittr, rlang, tidyselect, stats

RoxygenNote 6.0.1

Suggests testthat, knitr, tfestimators, keras

VignetteBuilder knitr

NeedsCompilation no

Author JJ Allaire [aut, cre], Yuan Tang [aut] (<https://orcid.org/0000-0001-5243-233X>), Kevin Ushey [aut], RStudio [cph, fnd], Google Inc. [cph]

Maintainer JJ Allaire <jj@rstudio.com>

Repository CRAN

Date/Publication 2018-08-25 13:20:03 UTC

R topics documented:

dataset_batch *Combines consecutive elements of this dataset into batches.*

Description

Combines consecutive elements of this dataset into batches.

Usage

```
dataset_batch(dataset, batch_size, drop_remainder = FALSE)
```

Arguments

dataset	A dataset
batch_size	An integer, representing the number of consecutive elements of this dataset to combine in a single batch.
drop_remainder	Ensure that batches have a fixed size by omitting any final smaller batch if it's present. Note that this is required for use with the Keras tensor inputs to fit/evaluate/etc.

Value

A dataset

See Also

Other dataset methods: dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_cache *Caches the elements in this dataset.*

Description

Caches the elements in this dataset.

Usage

```
dataset_cache(dataset, filename = NULL)
```

Arguments

dataset	A dataset
filename	String with 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.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

---

dataset_concatenate *Creates a dataset by concatenating given dataset with this dataset.*

Description

Creates a dataset by concatenating given dataset with this dataset.

Usage

```
dataset_concatenate(dataset, other)
```

Arguments

dataset	A dataset
other	Dataset to be concatenated

Value

A dataset

Note

Input dataset and dataset to be concatenated should have same nested structures and output types.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_decode_delim *Transform a dataset with delimted text lines into a dataset with named columns*

Description

Transform a dataset with delimted text lines into a dataset with named columns

Usage

```
dataset_decode_delim(dataset, record_spec, parallel_records = NULL)
```

Arguments

dataset	Dataset containing delimited text lines (e.g. a CSV)
record_spec	Specification of column names and types (see delim_record_spec()).
parallel_records	(Optional) An integer, representing the number of records to decode in parallel. If not specified, records will be processed sequentially.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_filter *Filter a dataset by a predicate*

Description

Filter a dataset by a predicate

Usage

```
dataset_filter(dataset, predicate)
```

Arguments

dataset	A dataset
predicate	A function mapping a nested structure of tensors (having shapes and types defined by output_shapes() and output_types() to a scalar tf$bool tensor.

Details

Note that the functions used inside the predicate must be tensor operations (e.g. `tf$not_equal`, `tf$less`, etc.). R generic methods for relational operators (e.g. <, >, <=, etc.) and logical operators (e.g. !, &, |, etc.) are provided so you can use shorthand syntax for most common comparisions (this is illustrated by the example below).

Value

A dataset composed of records that matched the predicate.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

Examples

```
## Not run:

dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_filter(function(record) {
    record$mpg >= 20
})

dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_filter(function(record) {
    record$mpg >= 20 & record$cyl >= 6L
  })


## End(Not run)
```

---

dataset_flat_map *Maps map_func across this dataset and flattens the result.*

Description

Maps map_func across this dataset and flattens the result.

Usage

```
dataset_flat_map(dataset, map_func)
```

Arguments

dataset	A dataset
map_func	A function mapping a nested structure of tensors (having shapes and types defined by output_shapes() and output_types() to a dataset.

Value

A dataset

---

dataset_interleave *Maps map_func across this dataset, and interleaves the results*

Description

Maps map_func across this dataset, and interleaves the results

Usage

```
dataset_interleave(dataset, map_func, cycle_length, block_length = 1)
```

Arguments

dataset	A dataset
map_func	A function mapping a nested structure of tensors (having shapes and types defined by output_shapes() and output_types() to a dataset.
cycle_length	The number of elements from this dataset that will be processed concurrently.
block_length	The number of consecutive elements to produce from each input element before cycling to another input element.

Details

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. In general, this transformation will apply map_func to 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.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

Examples

```
## Not run:

dataset <- tensor_slices_dataset(c(1,2,3,4,5)) %>%
 dataset_interleave(cycle_length = 2, block_length = 4, function(x) {
   tensors_dataset(x) %>%
     dataset_repeat(6)
 })

# resulting dataset (newlines indicate "block" boundaries):
c(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,
)


## End(Not run)
```

`dataset_map` *Map a function across a dataset.*

Description

Map a function across a dataset.

Usage

```
dataset_map(dataset, map_func, num_parallel_calls = NULL)
```

Arguments

`dataset`	A dataset
`map_func`	A function mapping a nested structure of tensors (having shapes and types defined by `output_shapes()` and `output_types()` to another nested structure of tensors.
`num_parallel_calls`	(Optional) An integer, representing the number of elements to process in parallel If not specified, elements will be processed sequentially.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_padded_batch, dataset_prefetch_to_dev dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

---

`dataset_map_and_batch` *Fused implementation of dataset_map() and dataset_batch()*

Description

Maps 'map_func" across batch_size consecutive elements of this dataset and then combines them into a batch. Functionally, it is equivalent to map followed by batch. However, by fusing the two transformations together, the implementation can be more efficient.

Usage

```
dataset_map_and_batch(dataset, map_func, batch_size,
  num_parallel_batches = NULL, drop_remainder = FALSE,
  num_parallel_calls = NULL)
```

Arguments

`dataset`	A dataset
`map_func`	A function mapping a nested structure of tensors (having shapes and types defined by `output_shapes()` and `output_types()` to another nested structure of tensors.
`batch_size`	An integer, representing the number of consecutive elements of this dataset to combine in a single batch.
`num_parallel_batches`	(Optional) An integer, 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`	Ensure that batches have a fixed size by omitting any final smaller batch if it's present. Note that this is required for use with the Keras tensor inputs to fit/evaluate/etc.
`num_parallel_calls`	(Optional) An integer, representing the number of elements to process in parallel If not specified, elements will be processed sequentially.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

---

`dataset_padded_batch` *Combines consecutive elements of this dataset into padded batches*

Description

This method combines multiple consecutive elements of this dataset, which might have different shapes, into a single element. The tensors in the resulting element have an additional outer dimension, and are padded to the respective shape in `padded_shapes`.

Usage

```
dataset_padded_batch(dataset, batch_size, padded_shapes,
  padding_values = NULL, drop_remainder = FALSE)
```

Arguments

`dataset`	A dataset
`batch_size`	An integer, representing the number of consecutive elements of this dataset to combine in a single batch.
`padded_shapes`	A nested structure of tf$TensorShape or integer vector tensor-like objects representing the shape to which the respective component of each input element should be padded prior to batching. Any unknown dimensions (e.g. `tf$Dimension(NULL)` in a `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, 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`	Ensure that batches have a fixed size by omitting any final smaller batch if it's present. Note that this is required for use with the Keras tensor inputs to fit/evaluate/etc.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_prefetch *Creates a Dataset that prefetches elements from this dataset.*

Description

Creates a Dataset that prefetches elements from this dataset.

Usage

```
dataset_prefetch(dataset, buffer_size)
```

Arguments

dataset	A dataset
buffer_size	An integer, representing the maximum number elements that will be buffered when prefetching.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_prefetch_to_device *A transformation that prefetches dataset values to the given* device

Description

A transformation that prefetches dataset values to the given device

Usage

```
dataset_prefetch_to_device(dataset, device, buffer_size = NULL)
```

Arguments

dataset	A dataset
device	A string. The name of a device to which elements will be prefetched (e.g. "/gpu:0").
buffer_size	(Optional.) The number of elements to buffer on device. Defaults to an automatically chosen value.

Value

A dataset

Note

Although the transformation creates a dataset, the transformation must be the final dataset in the input pipeline.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip, dataset_take

---

`dataset_prepare` *Prepare a dataset for analysis*

Description

Transform a dataset with named columns into a list with features (`x`) and response (`y`) elements.

Usage

```
dataset_prepare(dataset, x, y = NULL, named = TRUE,
  named_features = FALSE, parallel_records = NULL, batch_size = NULL,
  num_parallel_batches = NULL, drop_remainder = FALSE)
```

Arguments

`dataset`	A dataset
`x`	Features to include. When `named_features` is `FALSE` all features will be stacked into a single tensor so must have an identical data type.
`y`	(Optional). Response variable.
`named`	`TRUE` to name the dataset elements "x" and "y", `FALSE` to not name the dataset elements.
`named_features`	`TRUE` to yield features as a named list; `FALSE` to stack features into a single array. Note that in the case of `FALSE` (the default) all features will be stacked into a single 2D tensor so need to have the same underlying data type.
`parallel_records`	(Optional) An integer, representing the number of records to decode in parallel. If not specified, records will be processed sequentially.
`batch_size`	(Optional). Batch size if you would like to fuse the `dataset_prepare()` operation together with a `dataset_batch()` (fusing generally improves overall training performance).

num_parallel_batches
: (Optional) An integer, 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
: Ensure that batches have a fixed size by omitting any final smaller batch if it's present. Note that this is required for use with the Keras tensor inputs to fit/evaluate/etc.

Value

A dataset. The dataset will have a structure of either:

- When `named_features` is `TRUE`: `list(x = list(feature_name = feature_values, ...), y = response_values`
- When `named_features` is `FALSE`: `list(x = features_array, y = response_values)`, where `features_array` is a Rank 2 array of `(batch_size, num_features)`.

Note that the `y` element will be omitted when `y` is `NULL`.

See Also

input_fn() for use with **tfestimators**.

`dataset_repeat` *Repeats a dataset count times.*

Description

Repeats a dataset count times.

Usage

```
dataset_repeat(dataset, count = NULL)
```

Arguments

dataset
: A dataset

count
: (Optional.) An integer value representing the number of times the elements of this dataset should be repeated. The default behavior (if `count` is `NULL` or `-1`) is for the elements to be repeated indefinitely.

Value

A dataset

See Also

Other dataset methods: `dataset_batch`, `dataset_cache`, `dataset_concatenate`, `dataset_decode_delim`, `dataset_filter`, `dataset_interleave`, `dataset_map_and_batch`, `dataset_map`, `dataset_padded_batch`, `dataset_prefetch_to_device`, `dataset_prefetch`, `dataset_shuffle_and_repeat`, `dataset_shuffle`, `dataset_skip`, `dataset_take`

dataset_shard *Creates a dataset that includes only 1 / num_shards of this dataset.*

Description

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

Usage

```
dataset_shard(dataset, num_shards, index)
```

Arguments

dataset	A dataset
num_shards	A integer representing the number of shards operating in parallel.
index	A integer, representing the worker index.

Value

A dataset

dataset_shuffle *Randomly shuffles the elements of this dataset.*

Description

Randomly shuffles the elements of this dataset.

Usage

```
dataset_shuffle(dataset, buffer_size, seed = NULL)
```

Arguments

dataset	A dataset
buffer_size	An integer, representing the number of elements from this dataset from which the new dataset will sample.
seed	(Optional) An integer, representing the random seed that will be used to create the distribution.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_skip, dataset_take

---

`dataset_shuffle_and_repeat` *Shuffles and repeats a dataset returning a new permutation for each epoch.*

Description

Shuffles and repeats a dataset returning a new permutation for each epoch.

Usage

```
dataset_shuffle_and_repeat(dataset, buffer_size, count = NULL, seed = NULL)
```

Arguments

`dataset`	A dataset
`buffer_size`	An integer, representing the number of elements from this dataset from which the new dataset will sample.
`count`	(Optional.) An integer value representing the number of times the elements of this dataset should be repeated. The default behavior (if `count` is `NULL` or `-1`) is for the elements to be repeated indefinitely.
`seed`	(Optional) An integer, representing the random seed that will be used to create the distribution.

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle, dataset_skip, dataset_take

dataset_skip *Creates a dataset that skips count elements from this dataset*

Description

Creates a dataset that skips count elements from this dataset

Usage

```
dataset_skip(dataset, count)
```

Arguments

dataset	A dataset
count	An integer, 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.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_take

dataset_take *Creates a dataset with at most count elements from this dataset*

Description

Creates a dataset with at most count elements from this dataset

Usage

```
dataset_take(dataset, count)
```

Arguments

dataset	A dataset
count	Integer 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.

Value

A dataset

See Also

Other dataset methods: dataset_batch, dataset_cache, dataset_concatenate, dataset_decode_delim, dataset_filter, dataset_interleave, dataset_map_and_batch, dataset_map, dataset_padded_batch, dataset_prefetch_to_device, dataset_prefetch, dataset_repeat, dataset_shuffle_and_repeat, dataset_shuffle, dataset_skip

---

delim_record_spec *Specification for reading a record from a text file with delimited values*

Description

Specification for reading a record from a text file with delimited values

Usage

```
delim_record_spec(example_file, delim = ",", skip = 0, names = NULL,
  types = NULL, defaults = NULL)

csv_record_spec(example_file, skip = 0, names = NULL, types = NULL,
  defaults = NULL)

tsv_record_spec(example_file, skip = 0, names = NULL, types = NULL,
  defaults = NULL)
```

Arguments

example_file	File that provides an example of the records to be read. If you don't explicitly specify names and types (or defaults) then this file will be read to generate default values.
delim	Character delimiter to separate fields in a record (defaults to ",")
skip	Number of lines to skip before reading data. Note that if names is explicitly provided and there are column names witin the file then skip should be set to 1 to ensure that the column names are bypassed.
names	Character vector with column names (or NULL to automatically detect the column names from the first row of example_file). If names is a character vector, the values will be used as the names of the columns, and the first row of the input will be read into the first row of the datset. Note that if the underlying text file also includes column names in it's first row, this row should be skipped explicitly with skip = 1. If NULL, the first row of the example_file will be used as the column names, and will be skipped when reading the dataset.

types	Column types. If NULL and defaults is specified then types will be imputed from the defaults. Otherwise, all column types will be imputed from the first 1000 rows of the example_file. This is convenient (and fast), but not robust. If the imputation fails, you'll need to supply the correct types yourself. Types can be explicitliy specified in a character vector as "integer", "double", and "character" (e.g. col_types = c("double", "double", "integer"). Alternatively, you can use a compact string representation where each character represents one column: c = character, i = integer, d = double (e.g. types =ddi‘).
defaults	List of default values which are used when data is missing from a record (e.g. list(0, 0, 0L). If NULL then defaults will be automatically provided based on types (0 for numeric columns and "" for character columns).

---

file_list_dataset *A dataset of all files matching a pattern*

Description

A dataset of all files matching a pattern

Usage

```
file_list_dataset(file_pattern, shuffle = NULL, seed = NULL)
```

Arguments

file_pattern	A string, representing the filename pattern that will be matched.
shuffle	(Optional) If TRUE``, the file names will be shuffled randomly. Defaults toTRUE‘
seed	(Optional) An integer, representing the random seed that will be used to create the distribution.

Details

For example, if we had the following files on our filesystem: - /path/to/dir/a.txt - /path/to/dir/b.csv - /path/to/dir/c.csv

If we pass "/path/to/dir/*.csv" as the file_pattern, the dataset would produce: - /path/to/dir/b.csv - /path/to/dir/c.csv

Value

A dataset of string correponding to file names

Note

The shuffle and seed arguments only apply for TensorFlow >= v1.8

fixed_length_record_dataset *A dataset of fixed-length records from one or more binary files.*

Description

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

Usage

```
fixed_length_record_dataset(filenames, record_bytes, header_bytes = NULL,
  footer_bytes = NULL, buffer_size = NULL)
```

Arguments

filenames	A string tensor containing one or more filenames.
record_bytes	An integer representing the number of bytes in each record.
header_bytes	(Optional) An integer scalar representing the number of bytes to skip at the start of a file.
footer_bytes	(Optional) A integer scalar representing the number of bytes to ignore at the end of a file.
buffer_size	(Optional) A integer scalar representing the number of bytes to buffer when reading.

Value

A dataset

input_fn.tf_dataset *Construct a tfestimators input function from a dataset*

Description

Construct a tfestimators input function from a dataset

Usage

```
input_fn.tf_dataset(dataset, features, response = NULL)
```

Arguments

dataset	A dataset
features	The names of feature variables to be used.
response	The name of the response variable.

Details

Creating an input_fn from a dataset requires that the dataset consist of a set of named output tensors (e.g. like the dataset produced by the `tfrecord_dataset()` or `text_line_dataset()` function).

Value

An input_fn suitable for use with tfestimators train, evaluate, and predict methods

---

iterator_get_next *Get next element from iterator*

Description

Returns a nested list of tensors that when evaluated will yield the next element(s) in the dataset.

Usage

```
iterator_get_next(iterator, name = NULL)
```

Arguments

`iterator`	An iterator
`name`	(Optional) A name for the created operation.

Value

A nested list of tensors

See Also

Other iterator functions: `iterator_initializer`, `iterator_make_initializer`, `iterator_string_handle`, `make-iterator`

---

iterator_initializer *An operation that should be run to initialize this iterator.*

Description

An operation that should be run to initialize this iterator.

Usage

```
iterator_initializer(iterator)
```

Arguments

iterator	An iterator

See Also

Other iterator functions: iterator_get_next, iterator_make_initializer, iterator_string_handle, make-iterator

---

iterator_make_initializer

Create an operation that can be run to initialize this iterator

---

Description

Create an operation that can be run to initialize this iterator

Usage

```
iterator_make_initializer(iterator, dataset, name = NULL)
```

Arguments

iterator	An iterator
dataset	A dataset
name	(Optional) A name for the created operation.

Value

A tf$Operation that can be run to initialize this iterator on the given dataset.

See Also

Other iterator functions: iterator_get_next, iterator_initializer, iterator_string_handle, make-iterator

iterator_string_handle *String-valued tensor that represents this iterator*

Description

String-valued tensor that represents this iterator

Usage

```
iterator_string_handle(iterator, name = NULL)
```

Arguments

iterator	An iterator
name	(Optional) A name for the created operation.

Value

Scalar tensor of type string

See Also

Other iterator functions: iterator_get_next, iterator_initializer, iterator_make_initializer, make-iterator

make-iterator *Creates an iterator for enumerating the elements of this dataset.*

Description

Creates an iterator for enumerating the elements of this dataset.

Usage

```
make_iterator_one_shot(dataset)

make_iterator_initializable(dataset, shared_name = NULL)

make_iterator_from_structure(output_types, output_shapes = NULL,
  shared_name = NULL)

make_iterator_from_string_handle(string_handle, output_types,
  output_shapes = NULL)
```

Arguments

dataset	A 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).
output_types	A nested structure of tf$DType objects corresponding to each component of an element of this iterator.
output_shapes	(Optional) A nested structure of tf$TensorShape objects corresponding to each component of an element of this dataset. If omitted, each component will have an unconstrainted shape.
string_handle	A scalar tensor of type string that evaluates to a handle produced by the iterator_string_handle() method.

Value

An Iterator over the elements of this dataset.

Initialization

For make_iterator_one_shot(), the returned iterator will be initialized automatically. A "one-shot" iterator does not currently support re-initialization.

For make_iterator_initializable(), the returned iterator will be in an uninitialized state, and you must run the object returned from iterator_initializer() before using it.

For make_iterator_from_structure(), 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().

See Also

Other iterator functions: iterator_get_next, iterator_initializer, iterator_make_initializer, iterator_string_handle

---

make_csv_dataset *Reads CSV files into a batched dataset*

Description

Reads CSV files into a dataset, where each element is a (features, labels) list 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.

Usage

```
make_csv_dataset(file_pattern, batch_size, column_names = NULL,
  column_defaults = NULL, label_name = NULL, select_columns = NULL,
  field_delim = ",", use_quote_delim = TRUE, na_value = "",
  header = TRUE, num_epochs = NULL, shuffle = TRUE,
  shuffle_buffer_size = 10000, shuffle_seed = NULL,
  prefetch_buffer_size = 1, num_parallel_reads = 1,
  num_parallel_parser_calls = 2, sloppy = FALSE,
  num_rows_for_inference = 100)
```

Arguments

file_pattern	List of files or glob patterns of file paths containing CSV records.
batch_size	An integer 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 (integer, numeric, 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 Estiamtors and Keras.
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 integer specifying the number of times this dataset is repeated. If NULL, 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.

num_parallel_reads
Number of threads used to read CSV records from files. If >1, the results will be interleaved.

num_parallel_parser_calls
Number of parallel invocations of the CSV parsing function on CSV records.

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 NULL, reads all the rows of all the files. Defaults to 100.

Value

A dataset, where each element is a (features, labels) list that corresponds to a batch of batch_size CSV 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.

---

next_batch *Tensor(s) for retreiving the next batch from a dataset*

Description

Tensor(s) for retreiving the next batch from a dataset

Usage

```
next_batch(dataset)
```

Arguments

dataset	A dataset

Details

To access the underlying data within the dataset you iteratively evaluate the tensor(s) to read batches of data.

Note that in many cases you won't need to explicitly evaluate the tensors. Rather, you will pass the tensors to another function that will perform the evaluation (e.g. the Keras layer_input() and compile() functions).

If you do need to perform iteration manually by evaluating the tensors, there are a couple of possible approaches to controlling/detecting when iteration should end.

One approach is to create a dataset that yields batches infinitely (traversing the dataset multiple times with different batches randomly drawn). In this case you'd use another mechanism like a global step counter or detecting a learning plateau.

Another approach is to detect when all batches have been yielded from the dataset. When the tensor reaches the end of iteration a runtime error will occur. You can catch and ignore the error when it occurs by wrapping your iteration code in the `with_dataset()` function.

See the examples below for a demonstration of each of these methods of iteration.

Value

Tensor(s) that can be evaluated to yield the next batch of training data.

Examples

```
## Not run:

# iteration with 'infinite' dataset and explicit step counter

library(tfdatasets)
dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_prepare(x = c(mpg, disp), y = cyl) %>%
  dataset_shuffle(5000) %>%
  dataset_batch(128) %>%
  dataset_repeat() # repeat infinitely
batch <- next_batch(dataset)
steps <- 200
for (i in 1:steps) {
  # use batch$x and batch$y tensors
}

# iteration that detects and ignores end of iteration error

library(tfdatasets)
dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_prepare(x = c(mpg, disp), y = cyl) %>%
  dataset_batch(128) %>%
  dataset_repeat(10)
```

```
batch <- next_batch(dataset)
with_dataset({
  while(TRUE) {
    # use batch$x and batch$y tensors
  }
})

## End(Not run)
```

`output_types` *Output types and shapes*

Description

Output types and shapes

Usage

```
output_types(object)

output_shapes(object)
```

Arguments

`object`	A dataset or iterator

Value

`output_types()` returns the type of each component of an element of this object; `output_shapes()` returns the shape of each component of an element of this object

---

`range_dataset` *Creates a dataset of a step-separated range of values.*

Description

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

Usage

```
range_dataset(from = 0, to = 0, by = 1)
```

Arguments

`from`	Range start
`to`	Range end (exclusive)
`by`	Increment of the sequence

read_files *Read a dataset from a set of files*

Description

Read files into a dataset, optionally processing them in parallel.

Usage

```
read_files(files, reader, ..., parallel_files = 1, parallel_interleave = 1,
  num_shards = NULL, shard_index = NULL)
```

Arguments

files	List of filenames or glob pattern for files (e.g. "*.csv")
reader	Function that maps a file into a dataset (e.g. text_line_dataset() or tfrecord_dataset()).
...	Additional arguments to pass to reader function
parallel_files	An integer, number of files to process in parallel
parallel_interleave	An integer, number of consecutive records to produce from each file before cycling to another file.
num_shards	An integer representing the number of shards operating in parallel.
shard_index	An integer, representing the worker index. Shared indexes are 0 based so for e.g. 8 shards valid indexes would be 0-7.

Value

A dataset

sample_from_datasets *Samples elements at random from the datasets in* datasets.

Description

Samples elements at random from the datasets in datasets.

Usage

```
sample_from_datasets(datasets, weights = NULL, seed = NULL)
```

Arguments

datasets	A list ofobjects with compatible structure.
weights	(Optional.) A list of length(datasets) floating-point values where weights[[i]] represents the probability with which an element should be sampled from datasets[[i]], or a dataset object where each element is such a list. Defaults to a uniform distribution across datasets. [[i]: R:[i [[i]: R:[i
seed	(Optional.) An integer, representing the random seed that will be used to create the distribution.

Value

A dataset that interleaves elements from datasets at random, according to weights if provided, otherwise with uniform probability.

---

sparse_tensor_slices_dataset

Splits each rank-N tf$SparseTensor *in this dataset row-wise.*

Description

Splits each rank-N tf$SparseTensor in this dataset row-wise.

Usage

```
sparse_tensor_slices_dataset(sparse_tensor)
```

Arguments

sparse_tensor	A tf$SparseTensor.

Value

A dataset of rank-(N-1) sparse tensors.

See Also

Other tensor datasets: tensor_slices_dataset, tensors_dataset

sql_record_spec *A dataset consisting of the results from a SQL query*

Description

A dataset consisting of the results from a SQL query

Usage

```
sql_record_spec(names, types)

sql_dataset(driver_name, data_source_name, query, record_spec)

sqlite_dataset(filename, query, record_spec)
```

Arguments

names	Names of columns returned from the query
types	List of tf$DType objects (e.g. tf$int32, tf$double, tf$string) representing the types of the columns returned by the query.
driver_name	String containing the database type. Currently, the only supported value is 'sqlite'.
data_source_name	String containing a connection string to connect to the database.
query	String containing the SQL query to execute.
record_spec	Names and types of database columns
filename	Filename for the database

Value

A dataset

tensors_dataset *Creates a dataset with a single element, comprising the given tensors.*

Description

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

Usage

```
tensors_dataset(tensors)
```

Arguments

tensors	A nested structure of tensors.

Value

A dataset.

See Also

Other tensor datasets: sparse_tensor_slices_dataset, tensor_slices_dataset

---

tensor_slices_dataset *Creates a dataset whose elements are slices of the given tensors.*

Description

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

Usage

```
tensor_slices_dataset(tensors)
```

Arguments

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

Value

A dataset.

See Also

Other tensor datasets: sparse_tensor_slices_dataset, tensors_dataset

text_line_dataset *A dataset comprising lines from one or more text files.*

Description

A dataset comprising lines from one or more text files.

Usage

```
text_line_dataset(filenames, compression_type = NULL, record_spec = NULL,
  parallel_records = NULL)
```

Arguments

filenames	String(s) specifying one or more filenames
compression_type	A string, one of: NULL (no compression), "ZLIB", or "GZIP".
record_spec	(Optional) Specification used to decode delimimted text lines into records (see delim_record_spec()).
parallel_records	(Optional) An integer, representing the number of records to decode in parallel. If not specified, records will be processed sequentially.

Value

A dataset

tfrecord_dataset *A dataset comprising records from one or more TFRecord files.*

Description

A dataset comprising records from one or more TFRecord files.

Usage

```
tfrecord_dataset(filenames, compression_type = NULL, buffer_size = NULL,
  num_parallel_reads = NULL)
```

Arguments

filenames	String(s) specifying one or more filenames
compression_type	A string, one of: NULL (no compression), "ZLIB", or "GZIP".
buffer_size	An integer representing the number of bytes in the read buffer. (0 means no buffering).
num_parallel_reads	An integer representing the number of files to read in parallel. Defaults to reading files sequentially.

Details

If the dataset encodes a set of TFExample instances, then they can be decoded into named records using the dataset_map() function (see example below).

Examples

```
## Not run:

# Creates a dataset that reads all of the examples from two files, and extracts
# the image and label features.
filenames <- c("/var/data/file1.tfrecord", "/var/data/file2.tfrecord")
dataset <- tfrecord_dataset(filenames) %>%
  dataset_map(function(example_proto) {
    features <- list(
      image = tf$FixedLenFeature(shape(), tf$string, default_value = ""),
      label = tf$FixedLenFeature(shape(), tf$int32, default_value = 0L)
    )
    tf$parse_single_example(example_proto, features)
  })

## End(Not run)
```

---

until_out_of_range *Execute code that traverses a dataset until an out of range condition occurs*

Description

Execute code that traverses a dataset until an out of range condition occurs

Usage

```
until_out_of_range(expr)

out_of_range_handler(e)
```

Arguments

expr	Expression to execute (will be executed multiple times until the condition occurs)
e	Error object

Details

When a dataset iterator reaches the end, an out of range runtime error will occur. This function will catch and ignore the error when it occurs.

Examples

```
## Not run:
library(tfdatasets)
dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_batch(128) %>%
  dataset_repeat(10) %>%
  dataset_prepare(x = c(mpg, disp), y = cyl)

iter <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iter)

until_out_of_range({
  batch <- sess$run(next_batch)
  # use batch$x and batch$y tensors
})

## End(Not run)
```

---

with_dataset *Execute code that traverses a dataset*

Description

Execute code that traverses a dataset

Usage

```
with_dataset(expr)
```

Arguments

expr	Expression to execute

Details

When a dataset iterator reaches the end, an out of range runtime error will occur. You can catch and ignore the error when it occurs by wrapping your iteration code in a call to `with_dataset()` (see the example below for an illustration).

Examples

```
## Not run:
library(tfdatasets)
dataset <- text_line_dataset("mtcars.csv", record_spec = mtcars_spec) %>%
  dataset_prepare(x = c(mpg, disp), y = cyl) %>%
  dataset_batch(128) %>%
  dataset_repeat(10)

iter <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iter)

with_dataset({
  while(TRUE) {
    batch <- sess$run(next_batch)
    # use batch$x and batch$y tensors
  }
})

## End(Not run)
```

---

zip_datasets *Creates a dataset by zipping together the given datasets.*

Description

Merges datasets together into pairs or tuples that contain an element from each dataset.

Usage

```
zip_datasets(...)
```

Arguments

...	Datasets to zip (or a single argument with a list or list of lists of datasets).

Value

A dataset

Index