Module: tf.io

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

Public API for tf.io namespace.

Modules

[gfile](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile) module: Public API for tf.io.gfile namespace.

Classes

[class FixedLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature): Configuration for parsing a fixed-length input feature.

[class FixedLenSequenceFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature): Configuration for parsing a variable-length input feature into a Tensor.

[class SparseFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature): Configuration for parsing a sparse input feature from an Example.

[class TFRecordOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions): Options used for manipulating TFRecord files.

[class TFRecordWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter): A class to write records to a TFRecords file.

[class VarLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature): Configuration for parsing a variable-length input feature.

Functions

[decode\_and\_crop\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_and_crop_jpeg): Decode and Crop a JPEG-encoded image to a uint8 tensor.

[decode\_base64(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64): Decode web-safe base64-encoded strings.

[decode\_bmp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_bmp): Decode the first frame of a BMP-encoded image to a uint8 tensor.

[decode\_compressed(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_compressed): Decompress strings.

[decode\_csv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_csv): Convert CSV records to tensors. Each column maps to one tensor.

[decode\_gif(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_gif): Decode the frame(s) of a GIF-encoded image to a uint8 tensor.

[decode\_image(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image): Function for decode\_bmp, decode\_gif, decode\_jpeg, and decode\_png.

[decode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg): Decode a JPEG-encoded image to a uint8 tensor.

[decode\_json\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_json_example): Convert JSON-encoded Example records to binary protocol buffer strings.

[decode\_png(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_png): Decode a PNG-encoded image to a uint8 or uint16 tensor.

[decode\_proto(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_proto): The op extracts fields from a serialized protocol buffers message into tensors.

[decode\_raw(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_raw): Convert raw byte strings into tensors.

[deserialize\_many\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/deserialize_many_sparse): Deserialize and concatenate SparseTensors from a serialized minibatch.

[encode\_base64(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64): Encode strings into web-safe base64 format.

[encode\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_jpeg): JPEG-encode an image.

[encode\_proto(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_proto): The op serializes protobuf messages provided in the input tensors.

[extract\_jpeg\_shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/extract_jpeg_shape): Extract the shape information of a JPEG-encoded image.

[is\_jpeg(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/is_jpeg): Convenience function to check if the 'contents' encodes a JPEG image.

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

[matching\_files(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/matching_files): Returns the set of files matching one or more glob patterns.

[parse\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_example): Parses Example protos into a dict of tensors.

[parse\_sequence\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_sequence_example): Parses a batch of SequenceExample protos.

[parse\_single\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example): Parses a single Example proto.

[parse\_single\_sequence\_example(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_sequence_example): Parses a single SequenceExample proto.

[parse\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor): Transforms a serialized tensorflow.TensorProto proto into a Tensor.

[read\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file): Reads and outputs the entire contents of the input filename.

[serialize\_many\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_many_sparse): Serialize N-minibatch SparseTensor into an [N, 3] Tensor.

[serialize\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_sparse): Serialize a SparseTensor into a 3-vector (1-D Tensor) object.

[serialize\_tensor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/serialize_tensor): Transforms a Tensor into a serialized TensorProto proto.

[write\_file(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/write_file): Writes contents to the file at input filename. Creates file and recursively

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

# tf.compat.v1.io.TFRecordCompressionType

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#top_of_page)
* [Class TFRecordCompressionType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#class_tfrecordcompressiontype)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/io/TFRecordCompressionType#class_members)

## Class TFRecordCompressionType

The type of compression for the record.

### Aliases:

* Class tf.compat.v1.io.TFRecordCompressionType
* Class tf.compat.v1.python\_io.TFRecordCompressionType

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

## Class Members

* GZIP = 2
* NONE = 0
* ZLIB = 1

# tf.compat.v1.io.tf\_record\_iterator

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

An iteraror that read the records from a TFRecords file. (deprecated)

**Aliases:**

tf.compat.v1.io.tf\_record\_iterator

tf.compat.v1.python\_io.tf\_record\_iterator

tf.compat.v1.io.tf\_record\_iterator(  
    path,  
    options=None  
)

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use eager execution and: **tf.data.TFRecordDataset(path)**

**Args:**

**path**: The path to the TFRecords file.

**options**: (optional) A TFRecordOptions object.

**Yields:** Strings.

**Raises: IOError**: If path cannot be opened for reading.

reset\_states(states=None)

# tf.io.decode\_and\_crop\_jpeg

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

Decode and Crop a JPEG-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_and\_crop\_jpeg
* tf.compat.v1.io.decode\_and\_crop\_jpeg
* tf.compat.v2.image.decode\_and\_crop\_jpeg
* tf.compat.v2.io.decode\_and\_crop\_jpeg
* tf.image.decode\_and\_crop\_jpeg
* tf.io.decode\_and\_crop\_jpeg

tf.io.decode\_and\_crop\_jpeg(  
    contents,  
    crop\_window,  
    channels=0,  
    ratio=1,  
    fancy\_upscaling=True,  
    try\_recover\_truncated=False,  
    acceptable\_fraction=1,  
    dct\_method='',  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the JPEG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.

If needed, the JPEG-encoded image is transformed to match the requested number of color channels.

The attr ratio allows downscaling the image by an integer factor during decoding. Allowed values are: 1, 2, 4, and 8. This is much faster than downscaling the image later.

It is equivalent to a combination of decode and crop, but much faster by only decoding partial jpeg image.

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **crop\_window**: A Tensor of type int32. 1-D. The crop window: [crop\_y, crop\_x, crop\_height, crop\_width].
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **ratio**: An optional int. Defaults to 1. Downscaling ratio.
* **fancy\_upscaling**: An optional bool. Defaults to True. If true use a slower but nicer upscaling of the chroma planes (yuv420/422 only).
* **try\_recover\_truncated**: An optional bool. Defaults to False. If true try to recover an image from truncated input.
* **acceptable\_fraction**: An optional float. Defaults to 1. The minimum required fraction of lines before a truncated input is accepted.
* **dct\_method**: An optional string. Defaults to "". string specifying a hint about the algorithm used for decompression. Defaults to "" which maps to a system-specific default. Currently valid values are ["INTEGER\_FAST", "INTEGER\_ACCURATE"]. The hint may be ignored (e.g., the internal jpeg library changes to a version that does not have that specific option.)
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_base64

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64#aliases)

Decode web-safe base64-encoded strings.

### Aliases:

* tf.compat.v1.decode\_base64
* tf.compat.v1.io.decode\_base64
* tf.compat.v2.io.decode\_base64
* tf.io.decode\_base64

tf.io.decode\_base64(  
    input,  
    name=None  
)

Defined in generated file: python/ops/gen\_string\_ops.py.

Input may or may not have padding at the end. See EncodeBase64 for padding. Web-safe means that input must use - and \_ instead of + and /.

#### Args:

* **input**: A Tensor of type string. Base64 strings to decode.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_bmp

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

Decode the first frame of a BMP-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_bmp
* tf.compat.v1.io.decode\_bmp
* tf.compat.v2.image.decode\_bmp
* tf.compat.v2.io.decode\_bmp
* tf.image.decode\_bmp
* tf.io.decode\_bmp

tf.io.decode\_bmp(  
    contents,  
    channels=0,  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the BMP-encoded image.
* 3: output an RGB image.
* 4: output an RGBA image.

#### Args:

* **contents**: A Tensor of type string. 0-D. The BMP-encoded image.
* **channels**: An optional int. Defaults to 0.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_compressed

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

Decompress strings.

### Aliases:

* tf.compat.v1.decode\_compressed
* tf.compat.v1.io.decode\_compressed
* tf.compat.v2.io.decode\_compressed
* tf.io.decode\_compressed

tf.io.decode\_compressed(  
    bytes,  
    compression\_type='',  
    name=None  
)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

This op decompresses each element of the bytes input Tensor, which is assumed to be compressed using the given compression\_type.

The output is a string Tensor of the same shape as bytes, each element containing the decompressed data from the corresponding element in bytes.

#### Args:

* **bytes**: A Tensor of type string. A Tensor of string which is compressed.
* **compression\_type**: An optional string. Defaults to "". A scalar containing either (i) the empty string (no compression), (ii) "ZLIB", or (iii) "GZIP".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_csv

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

Convert CSV records to tensors. Each column maps to one tensor.

### Aliases:

* tf.compat.v2.io.decode\_csv
* tf.io.decode\_csv

tf.io.decode\_csv(  
    records,  
    record\_defaults,  
    field\_delim=',',  
    use\_quote\_delim=True,  
    na\_value='',  
    select\_cols=None,  
    name=None  
)

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

RFC 4180 format is expected for the CSV records. (https://tools.ietf.org/html/rfc4180) Note that we allow leading and trailing spaces with int or float field.

#### Args:

* **records**: A Tensor of type string. Each string is a record/row in the csv and all records should have the same format.
* **record\_defaults**: A list of Tensor objects with specific types. Acceptable types are float32, float64, int32, int64, string. One tensor per column of the input record, with either a scalar default value for that column or an empty vector if the column is required.
* **field\_delim**: An optional string. Defaults to ",". char delimiter to separate fields in a record.
* **use\_quote\_delim**: An optional bool. Defaults to True. If false, treats double quotation marks as regular characters inside of the string fields (ignoring RFC 4180, Section 2, Bullet 5).
* **na\_value**: Additional string to recognize as NA/NaN.
* **select\_cols**: Optional sorted list of column indices to select. If specified, only this subset of columns will be parsed and returned.
* **name**: A name for the operation (optional).

#### Returns:

A list of Tensor objects. Has the same type as record\_defaults. Each tensor will have the same shape as records.

#### Raises:

* **ValueError**: If any of the arguments is malformed.

# tf.io.decode\_gif

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

Decode the frame(s) of a GIF-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_gif
* tf.compat.v1.io.decode\_gif
* tf.compat.v2.image.decode\_gif
* tf.compat.v2.io.decode\_gif
* tf.image.decode\_gif
* tf.io.decode\_gif

tf.io.decode\_gif(  
    contents,  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

GIF images with frame or transparency compression are not supported. On Linux and MacOS systems, convert animated GIFs from compressed to uncompressed by running:

convert $src.gif -coalesce $dst.gif

This op also supports decoding JPEGs and PNGs, though it is cleaner to use[tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The GIF-encoded image.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_image

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image#used_in_the_tutorials)

Function for decode\_bmp, decode\_gif, decode\_jpeg, and decode\_png.

### Aliases:

* tf.compat.v1.image.decode\_image
* tf.compat.v1.io.decode\_image
* tf.compat.v2.image.decode\_image
* tf.compat.v2.io.decode\_image
* tf.image.decode\_image
* tf.io.decode\_image

tf.io.decode\_image(  
    contents,  
    channels=None,  
    dtype=tf.dtypes.uint8,  
    name=None,  
    expand\_animations=True  
)

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

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Detects whether an image is a BMP, GIF, JPEG, or PNG, and performs the appropriate operation to convert the input bytes string into a Tensor of type dtype.

**Note:** **decode\_gif** returns a 4-D array **[num\_frames, height, width, 3]**, as opposed to **decode\_bmp**, **decode\_jpeg** and **decode\_png**, which return 3-D arrays **[height, width, num\_channels]**. Make sure to take this into account when constructing your graph if you are intermixing GIF files with BMP, JPEG, and/or PNG files. Alternately, set the **expand\_animations** argument of this function to **False**, in which case the op will return 3-dimensional tensors and will truncate animated GIF files to the first frame.

#### Args:

* **contents**: 0-D string. The encoded image bytes.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **dtype**: The desired DType of the returned Tensor.
* **name**: A name for the operation (optional)
* **expand\_animations**: Controls the shape of the returned op's output. If True, the returned op will produce a 3-D tensor for PNG, JPEG, and BMP files; and a 4-D tensor for all GIFs, whether animated or not. If, False, the returned op will produce a 3-D tensor for all file types and will truncate animated GIFs to the first frame.

#### Returns:

Tensor with type dtype and a 3- or 4-dimensional shape, depending on the file type and the value of the expand\_animations parameter.

#### Raises:

* **ValueError**: On incorrect number of channels.

# tf.io.decode\_jpeg

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_jpeg#used_in_the_tutorials)

Decode a JPEG-encoded image to a uint8 tensor.

### Aliases:

* tf.compat.v1.image.decode\_jpeg
* tf.compat.v1.io.decode\_jpeg
* tf.compat.v2.image.decode\_jpeg
* tf.compat.v2.io.decode\_jpeg
* tf.image.decode\_jpeg
* tf.io.decode\_jpeg

tf.io.decode\_jpeg(  
    contents,  
    channels=0,  
    ratio=1,  
    fancy\_upscaling=True,  
    try\_recover\_truncated=False,  
    acceptable\_fraction=1,  
    dct\_method='',  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the JPEG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.

If needed, the JPEG-encoded image is transformed to match the requested number of color channels.

The attr ratio allows downscaling the image by an integer factor during decoding. Allowed values are: 1, 2, 4, and 8. This is much faster than downscaling the image later.

This op also supports decoding PNGs and non-animated GIFs since the interface is the same, though it is cleaner to use [tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **ratio**: An optional int. Defaults to 1. Downscaling ratio.
* **fancy\_upscaling**: An optional bool. Defaults to True. If true use a slower but nicer upscaling of the chroma planes (yuv420/422 only).
* **try\_recover\_truncated**: An optional bool. Defaults to False. If true try to recover an image from truncated input.
* **acceptable\_fraction**: An optional float. Defaults to 1. The minimum required fraction of lines before a truncated input is accepted.
* **dct\_method**: An optional string. Defaults to "". string specifying a hint about the algorithm used for decompression. Defaults to "" which maps to a system-specific default. Currently valid values are ["INTEGER\_FAST", "INTEGER\_ACCURATE"]. The hint may be ignored (e.g., the internal jpeg library changes to a version that does not have that specific option.)
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type uint8.

# tf.io.decode\_json\_example

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

Convert JSON-encoded Example records to binary protocol buffer strings.

### Aliases:

* tf.compat.v1.decode\_json\_example
* tf.compat.v1.io.decode\_json\_example
* tf.compat.v2.io.decode\_json\_example
* tf.io.decode\_json\_example

tf.io.decode\_json\_example(  
    json\_examples,  
    name=None  
)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

This op translates a tensor containing Example records, encoded using the [standard JSON mapping](https://developers.google.com/protocol-buffers/docs/proto3#json), into a tensor containing the same records encoded as binary protocol buffers. The resulting tensor can then be fed to any of the other Example-parsing ops.

#### Args:

* **json\_examples**: A Tensor of type string. Each string is a JSON object serialized according to the JSON mapping of the Example proto.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.decode\_png

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

Decode a PNG-encoded image to a uint8 or uint16 tensor.

### Aliases:

* tf.compat.v1.image.decode\_png
* tf.compat.v1.io.decode\_png
* tf.compat.v2.image.decode\_png
* tf.compat.v2.io.decode\_png
* tf.image.decode\_png
* tf.io.decode\_png

tf.io.decode\_png(  
    contents,  
    channels=0,  
    dtype=tf.dtypes.uint8,  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

The attr channels indicates the desired number of color channels for the decoded image.

#### Accepted values are:

* 0: Use the number of channels in the PNG-encoded image.
* 1: output a grayscale image.
* 3: output an RGB image.
* 4: output an RGBA image.

If needed, the PNG-encoded image is transformed to match the requested number of color channels.

This op also supports decoding JPEGs and non-animated GIFs since the interface is the same, though it is cleaner to use [tf.image.decode\_image](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_image).

#### Args:

* **contents**: A Tensor of type string. 0-D. The PNG-encoded image.
* **channels**: An optional int. Defaults to 0. Number of color channels for the decoded image.
* **dtype**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.uint8, tf.uint16. Defaults to [tf.uint8](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#uint8).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type dtype.

# tf.io.decode\_proto

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

The op extracts fields from a serialized protocol buffers message into tensors.

### Aliases:

* tf.compat.v1.io.decode\_proto
* tf.compat.v2.io.decode\_proto
* tf.io.decode\_proto

tf.io.decode\_proto(  
    bytes,  
    message\_type,  
    field\_names,  
    output\_types,  
    descriptor\_source='local://',  
    message\_format='binary',  
    sanitize=False,  
    name=None  
)

Defined in generated file: python/ops/gen\_decode\_proto\_ops.py.

The decode\_proto op extracts fields from a serialized protocol buffers message into tensors. The fields in field\_names are decoded and converted to the corresponding output\_types if possible.

A message\_type name must be provided to give context for the field names. The actual message descriptor can be looked up either in the linked-in descriptor pool or a filename provided by the caller using the descriptor\_source attribute.

Each output tensor is a dense tensor. This means that it is padded to hold the largest number of repeated elements seen in the input minibatch. (The shape is also padded by one to prevent zero-sized dimensions). The actual repeat counts for each example in the minibatch can be found in the sizesoutput. In many cases the output of decode\_proto is fed immediately into tf.squeeze if missing values are not a concern. When using tf.squeeze, always pass the squeeze dimension explicitly to avoid surprises.

For the most part, the mapping between Proto field types and TensorFlow dtypes is straightforward. However, there are a few special cases:

* A proto field that contains a submessage or group can only be converted to DT\_STRING (the serialized submessage). This is to reduce the complexity of the API. The resulting string can be used as input to another instance of the decode\_proto op.
* TensorFlow lacks support for unsigned integers. The ops represent uint64 types as a DT\_INT64with the same twos-complement bit pattern (the obvious way). Unsigned int32 values can be represented exactly by specifying type DT\_INT64, or using twos-complement if the caller specifies DT\_INT32 in the output\_types attribute.

The descriptor\_source attribute selects a source of protocol descriptors to consult when looking up message\_type. This may be a filename containing a serialized FileDescriptorSet message, or the special value local://, in which case only descriptors linked into the code will be searched; the filename can be on any filesystem accessible to TensorFlow.

You can build a descriptor\_source file using the --descriptor\_set\_out and --include\_imports options to the protocol compiler protoc.

The local:// database only covers descriptors linked into the code via C++ libraries, not Python imports. You can link in a proto descriptor by creating a cc\_library target with alwayslink=1.

Both binary and text proto serializations are supported, and can be chosen using the format attribute.

#### Args:

* **bytes**: A Tensor of type string. Tensor of serialized protos with shape batch\_shape.
* **message\_type**: A string. Name of the proto message type to decode.
* **field\_names**: A list of strings. List of strings containing proto field names. An extension field can be decoded by using its full name, e.g. EXT\_PACKAGE.EXT\_FIELD\_NAME.
* **output\_types**: A list of tf.DTypes. List of TF types to use for the respective field in field\_names.
* **descriptor\_source**: An optional string. Defaults to "local://". Either the special value local:// or a path to a file containing a serialized FileDescriptorSet.
* **message\_format**: An optional string. Defaults to "binary". Either binary or text.
* **sanitize**: An optional bool. Defaults to False. Whether to sanitize the result or not.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (sizes, values).

* **sizes**: A Tensor of type int32.
* **values**: A list of Tensor objects of type output\_types.

# tf.io.decode\_raw

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

Convert raw byte strings into tensors.

### Aliases:

* tf.compat.v2.io.decode\_raw
* tf.io.decode\_raw

tf.io.decode\_raw(  
    input\_bytes,  
    out\_type,  
    little\_endian=True,  
    fixed\_length=None,  
    name=None  
)

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

#### Args:

* **input\_bytes**: Each element of the input Tensor is converted to an array of bytes.
* **out\_type**: DType of the output. Acceptable types are half, float, double, int32, uint16, uint8, int16, int8, int64.
* **little\_endian**: Whether the input\_bytes data is in little-endian format. Data will be converted into host byte order if necessary.
* **fixed\_length**: If set, the first fixed\_length bytes of each element will be converted. Data will be zero-padded or truncated to the specified length.

fixed\_length must be a multiple of the size of out\_type. fixed\_length must be specified if the elements of input\_bytes are of variable length.

* **name**: A name for the operation (optional).

#### Returns:

A Tensor object storing the decoded bytes.

# tf.io.deserialize\_many\_sparse

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

Deserialize and concatenate SparseTensors from a serialized minibatch.

### Aliases:

* tf.compat.v1.deserialize\_many\_sparse
* tf.compat.v1.io.deserialize\_many\_sparse
* tf.compat.v2.io.deserialize\_many\_sparse
* tf.io.deserialize\_many\_sparse

tf.io.deserialize\_many\_sparse(  
    serialized\_sparse,  
    dtype,  
    rank=None,  
    name=None  
)

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

The input serialized\_sparse must be a string matrix of shape [N x 3] where N is the minibatch size and the rows correspond to packed outputs of serialize\_sparse. The ranks of the original SparseTensor objects must all match. When the final SparseTensor is created, it has rank one higher than the ranks of the incoming SparseTensor objects (they have been concatenated along a new row dimension).

The output SparseTensor object's shape values for all dimensions but the first are the max across the input SparseTensor objects' shape values for the corresponding dimensions. Its first shape value is N, the minibatch size.

The input SparseTensor objects' indices are assumed ordered in standard lexicographic order. If this is not the case, after this step run sparse.reorder to restore index ordering.

For example, if the serialized input is a [2, 3] matrix representing two original SparseTensorobjects:

index = [ 0]  
        [10]  
        [20]  
values = [1, 2, 3]  
shape = [50]

and

index = [ 2]  
        [10]  
values = [4, 5]  
shape = [30]

then the final deserialized SparseTensor will be:

index = [0  0]  
        [0 10]  
        [0 20]  
        [1  2]  
        [1 10]  
values = [1, 2, 3, 4, 5]  
shape = [2 50]

#### Args:

* **serialized\_sparse**: 2-D Tensor of type string of shape [N, 3]. The serialized and packed SparseTensor objects.
* **dtype**: The dtype of the serialized SparseTensor objects.
* **rank**: (optional) Python int, the rank of the SparseTensor objects.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A SparseTensor representing the deserialized SparseTensors, concatenated along the SparseTensors' first dimension.

All of the serialized SparseTensors must have had the same rank and type.

# tf.io.encode\_base64

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64#aliases)

Encode strings into web-safe base64 format.

### Aliases:

* tf.compat.v1.encode\_base64
* tf.compat.v1.io.encode\_base64
* tf.compat.v2.io.encode\_base64
* tf.io.encode\_base64

tf.io.encode\_base64(  
    input,  
    pad=False,  
    name=None  
)

Defined in generated file: python/ops/gen\_string\_ops.py.

Refer to the following article for more information on base64 format: en.wikipedia.org/wiki/Base64. Base64 strings may have padding with '=' at the end so that the encoded has length multiple of 4. See Padding section of the link above.

Web-safe means that the encoder uses - and \_ instead of + and /.

#### Args:

* **input**: A Tensor of type string. Strings to be encoded.
* **pad**: An optional bool. Defaults to False. Bool whether padding is applied at the ends.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.encode\_jpeg

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

JPEG-encode an image.

### Aliases:

* tf.compat.v1.image.encode\_jpeg
* tf.compat.v1.io.encode\_jpeg
* tf.compat.v2.image.encode\_jpeg
* tf.compat.v2.io.encode\_jpeg
* tf.image.encode\_jpeg
* tf.io.encode\_jpeg

tf.io.encode\_jpeg(  
    image,  
    format='',  
    quality=95,  
    progressive=False,  
    optimize\_size=False,  
    chroma\_downsampling=True,  
    density\_unit='in',  
    x\_density=300,  
    y\_density=300,  
    xmp\_metadata='',  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

image is a 3-D uint8 Tensor of shape [height, width, channels].

The attr format can be used to override the color format of the encoded output. Values can be:

* '': Use a default format based on the number of channels in the image.
* grayscale: Output a grayscale JPEG image. The channels dimension of image must be 1.
* rgb: Output an RGB JPEG image. The channels dimension of image must be 3.

If format is not specified or is the empty string, a default format is picked in function of the number of channels in image:

* 1: Output a grayscale image.
* 3: Output an RGB image.

#### Args:

* **image**: A Tensor of type uint8. 3-D with shape [height, width, channels].
* **format**: An optional string from: "", "grayscale", "rgb". Defaults to "". Per pixel image format.
* **quality**: An optional int. Defaults to 95. Quality of the compression from 0 to 100 (higher is better and slower).
* **progressive**: An optional bool. Defaults to False. If True, create a JPEG that loads progressively (coarse to fine).
* **optimize\_size**: An optional bool. Defaults to False. If True, spend CPU/RAM to reduce size with no quality change.
* **chroma\_downsampling**: An optional bool. Defaults to True. See http://en.wikipedia.org/wiki/Chroma\_subsampling.
* **density\_unit**: An optional string from: "in", "cm". Defaults to "in". Unit used to specify x\_density and y\_density: pixels per inch ('in') or centimeter ('cm').
* **x\_density**: An optional int. Defaults to 300. Horizontal pixels per density unit.
* **y\_density**: An optional int. Defaults to 300. Vertical pixels per density unit.
* **xmp\_metadata**: An optional string. Defaults to "". If not empty, embed this XMP metadata in the image header.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.encode\_proto

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

The op serializes protobuf messages provided in the input tensors.

### Aliases:

* tf.compat.v1.io.encode\_proto
* tf.compat.v2.io.encode\_proto
* tf.io.encode\_proto

tf.io.encode\_proto(  
    sizes,  
    values,  
    field\_names,  
    message\_type,  
    descriptor\_source='local://',  
    name=None  
)

Defined in generated file: python/ops/gen\_encode\_proto\_ops.py.

The types of the tensors in values must match the schema for the fields specified in field\_names. All the tensors in values must have a common shape prefix, batch\_shape.

The sizes tensor specifies repeat counts for each field. The repeat count (last dimension) of a each tensor in values must be greater than or equal to corresponding repeat count in sizes.

A message\_type name must be provided to give context for the field names. The actual message descriptor can be looked up either in the linked-in descriptor pool or a filename provided by the caller using the descriptor\_source attribute.

The descriptor\_source attribute selects a source of protocol descriptors to consult when looking up message\_type. This may be a filename containing a serialized FileDescriptorSet message, or the special value local://, in which case only descriptors linked into the code will be searched; the filename can be on any filesystem accessible to TensorFlow.

You can build a descriptor\_source file using the --descriptor\_set\_out and --include\_imports options to the protocol compiler protoc.

The local:// database only covers descriptors linked into the code via C++ libraries, not Python imports. You can link in a proto descriptor by creating a cc\_library target with alwayslink=1.

There are a few special cases in the value mapping:

Submessage and group fields must be pre-serialized as TensorFlow strings.

TensorFlow lacks support for unsigned int64s, so they must be represented as [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) with the same twos-complement bit pattern (the obvious way).

Unsigned int32 values can be represented exactly with [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64), or with sign wrapping if the input is of type [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32).

#### Args:

* **sizes**: A Tensor of type int32. Tensor of int32 with shape [batch\_shape, len(field\_names)].
* **values**: A list of Tensor objects. List of tensors containing values for the corresponding field.
* **field\_names**: A list of strings. List of strings containing proto field names.
* **message\_type**: A string. Name of the proto message type to decode.
* **descriptor\_source**: An optional string. Defaults to "local://".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.extract\_jpeg\_shape

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

Extract the shape information of a JPEG-encoded image.

### Aliases:

* tf.compat.v1.image.extract\_jpeg\_shape
* tf.compat.v1.io.extract\_jpeg\_shape
* tf.compat.v2.image.extract\_jpeg\_shape
* tf.compat.v2.io.extract\_jpeg\_shape
* tf.image.extract\_jpeg\_shape
* tf.io.extract\_jpeg\_shape

tf.io.extract\_jpeg\_shape(  
    contents,  
    output\_type=tf.dtypes.int32,  
    name=None  
)

Defined in generated file: python/ops/gen\_image\_ops.py.

This op only parses the image header, so it is much faster than DecodeJpeg.

#### Args:

* **contents**: A Tensor of type string. 0-D. The JPEG-encoded image.
* **output\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.int32, tf.int64. Defaults to [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32). (Optional) The output type of the operation (int32 or int64). Defaults to int32.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

# tf.io.FixedLenFeature

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#top_of_page)
* [Class FixedLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#class_fixedlenfeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#used_in_the_tutorials)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenFeature#properties)

## Class FixedLenFeature

Configuration for parsing a fixed-length input feature.

### Aliases:

* Class tf.compat.v1.FixedLenFeature
* Class tf.compat.v1.io.FixedLenFeature
* Class tf.compat.v2.io.FixedLenFeature
* Class tf.io.FixedLenFeature

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

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

To treat sparse input as dense, provide a default\_value; otherwise, the parse functions will fail on any examples missing this feature.

#### Fields:

* **shape**: Shape of input data.
* **dtype**: Data type of input.
* **default\_value**: Value to be used if an example is missing this feature. It must be compatible with dtype and of the specified shape.

## Properties

### shape

### dtype

### default\_value

# tf.io.FixedLenSequenceFeature

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#top_of_page)
* [Class FixedLenSequenceFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#class_fixedlensequencefeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#properties)
  + [shape](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/FixedLenSequenceFeature#shape)

## Class FixedLenSequenceFeature

Configuration for parsing a variable-length input feature into a Tensor.

### Aliases:

* Class tf.compat.v1.FixedLenSequenceFeature
* Class tf.compat.v1.io.FixedLenSequenceFeature
* Class tf.compat.v2.io.FixedLenSequenceFeature
* Class tf.io.FixedLenSequenceFeature

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

The resulting Tensor of parsing a single SequenceExample or Example has a static shape of [None] + shape and the specified dtype. The resulting Tensor of parsing a batch\_size many Examples has a static shape of [batch\_size, None] + shape and the specified dtype. The entries in the batch from different Examples will be padded with default\_value to the maximum length present in the batch.

To treat a sparse input as dense, provide allow\_missing=True; otherwise, the parse functions will fail on any examples missing this feature.

#### Fields:

* **shape**: Shape of input data for dimension 2 and higher. First dimension is of variable length None.
* **dtype**: Data type of input.
* **allow\_missing**: Whether to allow this feature to be missing from a feature list item. Is available only for parsing SequenceExample not for parsing Examples.
* **default\_value**: Scalar value to be used to pad multiple Examples to their maximum length. Irrelevant for parsing a single Example or SequenceExample. Defaults to "" for dtype string and 0 otherwise (optional).

## Properties

### shape

### dtype

### allow\_missing

### default\_value

# tf.io.is\_jpeg

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

Convenience function to check if the 'contents' encodes a JPEG image.

### Aliases:

* tf.compat.v1.image.is\_jpeg
* tf.compat.v1.io.is\_jpeg
* tf.compat.v2.image.is\_jpeg
* tf.compat.v2.io.is\_jpeg
* tf.image.is\_jpeg
* tf.io.is\_jpeg

tf.io.is\_jpeg(  
    contents,  
    name=None  
)

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

#### Args:

* **contents**: 0-D string. The encoded image bytes.
* **name**: A name for the operation (optional)

#### Returns:

A scalar boolean tensor indicating if 'contents' may be a JPEG image. is\_jpeg is susceptible to false positives.

# tf.io.matching\_files

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

Returns the set of files matching one or more glob patterns.

### Aliases:

* tf.compat.v1.io.matching\_files
* tf.compat.v1.matching\_files
* tf.compat.v2.io.matching\_files
* tf.io.matching\_files

tf.io.matching\_files(  
    pattern,  
    name=None  
)

Defined in generated file: python/ops/gen\_io\_ops.py.

Note that this routine only supports wildcard characters in the basename portion of the pattern, not in the directory portion. Note also that the order of filenames returned can be non-deterministic.

#### Args:

* **pattern**: A Tensor of type string. Shell wildcard pattern(s). Scalar or vector of type string.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.match\_filenames\_once

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

Save the list of files matching pattern, so it is only computed once.

### Aliases:

* tf.compat.v1.io.match\_filenames\_once
* tf.compat.v1.train.match\_filenames\_once
* tf.compat.v2.io.match\_filenames\_once
* tf.io.match\_filenames\_once

tf.io.match\_filenames\_once(  
    pattern,  
    name=None  
)

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

NOTE: The order of the files returned can be non-deterministic.

#### Args:

* **pattern**: A file pattern (glob), or 1D tensor of file patterns.
* **name**: A name for the operations (optional).

#### Returns:

A variable that is initialized to the list of files matching the pattern(s).

# tf.io.parse\_example

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

Parses Example protos into a dict of tensors.

### Aliases:

* tf.compat.v2.io.parse\_example
* tf.io.parse\_example

tf.io.parse\_example(  
    serialized,  
    features,  
    example\_names=None,  
    name=None  
)

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

Parses a number of serialized [Example](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) protos given in serialized. We refer to serialized as a batch with batch\_size many entries of individual Example protos.

example\_names may contain descriptive names for the corresponding serialized protos. These may be useful for debugging purposes, but they have no effect on the output. If not None, example\_namesmust be the same length as serialized.

This op parses serialized examples into a dictionary mapping keys to Tensor and SparseTensorobjects. features is a dict from keys to VarLenFeature, SparseFeature, and FixedLenFeatureobjects. Each VarLenFeature and SparseFeature is mapped to a SparseTensor, and eachFixedLenFeature is mapped to a Tensor.

Each VarLenFeature maps to a SparseTensor of the specified type representing a ragged matrix. Its indices are [batch, index] where batch identifies the example in serialized, and index is the value's index in the list of values associated with that feature and example.

Each SparseFeature maps to a SparseTensor of the specified type representing a Tensor of dense\_shape [batch\_size] + SparseFeature.size. Its values come from the feature in the examples with key value\_key. A values[i] comes from a position k in the feature of an example at batch entry batch. This positional information is recorded in indices[i] as [batch, index\_0, index\_1, ...] where index\_j is the k-th value of the feature in the example at with key SparseFeature.index\_key[j]. In other words, we split the indices (except the first index indicating the batch entry) of a SparseTensor by dimension into different features of the Example. Due to its complexity a VarLenFeature should be preferred over a SparseFeature whenever possible.

Each FixedLenFeature df maps to a Tensor of the specified type (or [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) if not specified) and shape (serialized.size(),) + df.shape.

FixedLenFeature entries with a default\_value are optional. With no default value, we will fail if that Feature is missing from any example in serialized.

Each FixedLenSequenceFeature df maps to a Tensor of the specified type (or [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) if not specified) and shape (serialized.size(), None) + df.shape. All examples in serialized will be padded with default\_value along the second dimension.

#### Examples:

For example, if one expects a [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) VarLenFeature ft and three serialized Examples are provided:

serialized = [  
  features  
    { feature { key: "ft" value { float\_list { value: [1.0, 2.0] } } } },  
  features  
    { feature []},  
  features  
    { feature { key: "ft" value { float\_list { value: [3.0] } } }  
]

then the output will look like:

{"ft": SparseTensor(indices=[[0, 0], [0, 1], [2, 0]],  
                    values=[1.0, 2.0, 3.0],  
                    dense\_shape=(3, 2)) }

If instead a FixedLenSequenceFeature with default\_value = -1.0 and shape=[] is used then the output will look like:

{"ft": [[1.0, 2.0], [3.0, -1.0]]}

Given two Example input protos in serialized:

[  
  features {  
    feature { key: "kw" value { bytes\_list { value: [ "knit", "big" ] } } }  
    feature { key: "gps" value { float\_list { value: [] } } }  
  },  
  features {  
    feature { key: "kw" value { bytes\_list { value: [ "emmy" ] } } }  
    feature { key: "dank" value { int64\_list { value: [ 42 ] } } }  
    feature { key: "gps" value { } }  
  }  
]

And arguments

example\_names: ["input0", "input1"],  
features: {  
    "kw": VarLenFeature(tf.string),  
    "dank": VarLenFeature(tf.int64),  
    "gps": VarLenFeature(tf.float32),  
}

Then the output is a dictionary:

{  
  "kw": SparseTensor(  
      indices=[[0, 0], [0, 1], [1, 0]],  
      values=["knit", "big", "emmy"]  
      dense\_shape=[2, 2]),  
  "dank": SparseTensor(  
      indices=[[1, 0]],  
      values=[42],  
      dense\_shape=[2, 1]),  
  "gps": SparseTensor(  
      indices=[],  
      values=[],  
      dense\_shape=[2, 0]),  
}

For dense results in two serialized Examples:

[  
  features {  
    feature { key: "age" value { int64\_list { value: [ 0 ] } } }  
    feature { key: "gender" value { bytes\_list { value: [ "f" ] } } }  
   },  
   features {  
    feature { key: "age" value { int64\_list { value: [] } } }  
    feature { key: "gender" value { bytes\_list { value: [ "f" ] } } }  
  }  
]

#### We can use arguments:

example\_names: ["input0", "input1"],  
features: {  
    "age": FixedLenFeature([], dtype=tf.int64, default\_value=-1),  
    "gender": FixedLenFeature([], dtype=tf.string),  
}

And the expected output is:

{  
  "age": [[0], [-1]],  
  "gender": [["f"], ["f"]],  
}

An alternative to VarLenFeature to obtain a SparseTensor is SparseFeature. For example, given two Example input protos in serialized:

[  
  features {  
    feature { key: "val" value { float\_list { value: [ 0.5, -1.0 ] } } }  
    feature { key: "ix" value { int64\_list { value: [ 3, 20 ] } } }  
  },  
  features {  
    feature { key: "val" value { float\_list { value: [ 0.0 ] } } }  
    feature { key: "ix" value { int64\_list { value: [ 42 ] } } }  
  }  
]

And arguments

example\_names: ["input0", "input1"],  
features: {  
    "sparse": SparseFeature(  
        index\_key="ix", value\_key="val", dtype=tf.float32, size=100),  
}

Then the output is a dictionary:

{  
  "sparse": SparseTensor(  
      indices=[[0, 3], [0, 20], [1, 42]],  
      values=[0.5, -1.0, 0.0]  
      dense\_shape=[2, 100]),  
}

#### Args:

* **serialized**: A vector (1-D Tensor) of strings, a batch of binary serialized Example protos.
* **features**: A dict mapping feature keys to FixedLenFeature, VarLenFeature, and SparseFeature values.
* **example\_names**: A vector (1-D Tensor) of strings (optional), the names of the serialized protos in the batch.
* **name**: A name for this operation (optional).

#### Returns:

A dict mapping feature keys to Tensor and SparseTensor values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_sequence\_example

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

Parses a batch of SequenceExample protos.

### Aliases:

* tf.compat.v1.io.parse\_sequence\_example
* tf.compat.v2.io.parse\_sequence\_example
* tf.io.parse\_sequence\_example

tf.io.parse\_sequence\_example(  
    serialized,  
    context\_features=None,  
    sequence\_features=None,  
    example\_names=None,  
    name=None  
)

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

Parses a vector of serialized [SequenceExample](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) protos given in serialized.

This op parses serialized sequence examples into a tuple of dictionaries mapping keys to Tensor and SparseTensor objects respectively. The first dictionary contains mappings for keys appearing incontext\_features, and the second dictionary contains mappings for keys appearing in sequence\_features.

At least one of context\_features and sequence\_features must be provided and non-empty.

The context\_features keys are associated with a SequenceExample as a whole, independent of time / frame. In contrast, the sequence\_features keys provide a way to access variable-length data within the FeatureList section of the SequenceExample proto. While the shapes of context\_features values are fixed with respect to frame, the frame dimension (the first dimension) of sequence\_features values may vary between SequenceExample protos, and even between feature\_list keys within the same SequenceExample.

context\_features contains VarLenFeature and FixedLenFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenFeature is mapped to a Tensor, of the specified type, shape, and default value.

sequence\_features contains VarLenFeature and FixedLenSequenceFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenSequenceFeature is mapped to a Tensor, each of the specified type. The shape will be (B,T,) + df.dense\_shape for FixedLenSequenceFeature df, where B is the batch size, and T is the length of the associatedFeatureList in the SequenceExample. For instance, FixedLenSequenceFeature([]) yields a scalar 2-D Tensor of static shape [None, None] and dynamic shape [B, T], whileFixedLenSequenceFeature([k]) (for int k >= 1) yields a 3-D matrix Tensor of static shape [None, None, k] and dynamic shape [B, T, k].

Like the input, the resulting output tensors have a batch dimension. This means that the original per-example shapes of VarLenFeatures and FixedLenSequenceFeatures can be lost. To handle that situation, this op also provides dicts of shape tensors as part of the output. There is one dict for the context features, and one for the feature\_list features. Context features of type FixedLenFeatures will not be present, since their shapes are already known by the caller. In situations where the input 'FixedLenFeature`s are of different lengths across examples, the shorter examples will be padded with default datatype values: 0 for numeric types, and the empty string for string types.

Each SparseTensor corresponding to sequence\_features represents a ragged vector. Its indices are [time, index], where time is the FeatureList entry and index is the value's index in the list of values associated with that time.

FixedLenFeature entries with a default\_value and FixedLenSequenceFeature entries with allow\_missing=True are optional; otherwise, we will fail if that Feature or FeatureList is missing from any example in serialized.

example\_name may contain a descriptive name for the corresponding serialized proto. This may be useful for debugging purposes, but it has no effect on the output. If not None, example\_name must be a scalar.

#### Args:

* **serialized**: A vector (1-D Tensor) of type string containing binary serialized SequenceExample protos.
* **context\_features**: A dict mapping feature keys to FixedLenFeature or VarLenFeaturevalues. These features are associated with a SequenceExample as a whole.
* **sequence\_features**: A dict mapping feature keys to FixedLenSequenceFeature or VarLenFeature values. These features are associated with data within the FeatureListsection of the SequenceExample proto.
* **example\_names**: A vector (1-D Tensor) of strings (optional), the name of the serialized protos.
* **name**: A name for this operation (optional).

#### Returns:

A tuple of three dicts, each mapping keys to Tensors and SparseTensors. The first dict contains the context key/values, the second dict contains the feature\_list key/values, and the final dict contains the lengths of any dense feature\_list features.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_single\_example

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_single_example#used_in_the_tutorials)

Parses a single Example proto.

### Aliases:

* tf.compat.v2.io.parse\_single\_example
* tf.io.parse\_single\_example

tf.io.parse\_single\_example(  
    serialized,  
    features,  
    example\_names=None,  
    name=None  
)

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

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

Similar to parse\_example, except:

For dense tensors, the returned Tensor is identical to the output of parse\_example, except there is no batch dimension, the output shape is the same as the shape given in dense\_shape.

For SparseTensors, the first (batch) column of the indices matrix is removed (the indices matrix is a column vector), the values vector is unchanged, and the first (batch\_size) entry of the shape vector is removed (it is now a single element vector).

One might see performance advantages by batching Example protos with parse\_example instead of using this function directly.

#### Args:

* **serialized**: A scalar string Tensor, a single serialized Example. See \_parse\_single\_example\_raw documentation for more details.
* **features**: A dict mapping feature keys to FixedLenFeature or VarLenFeature values.
* **example\_names**: (Optional) A scalar string Tensor, the associated name. See \_parse\_single\_example\_raw documentation for more details.
* **name**: A name for this operation (optional).

#### Returns:

A dict mapping feature keys to Tensor and SparseTensor values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_single\_sequence\_example

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

Parses a single SequenceExample proto.

### Aliases:

* tf.compat.v1.io.parse\_single\_sequence\_example
* tf.compat.v1.parse\_single\_sequence\_example
* tf.compat.v2.io.parse\_single\_sequence\_example
* tf.io.parse\_single\_sequence\_example

tf.io.parse\_single\_sequence\_example(  
    serialized,  
    context\_features=None,  
    sequence\_features=None,  
    example\_name=None,  
    name=None  
)

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

Parses a single serialized [SequenceExample](https://www.tensorflow.org/code/tensorflow/core/example/example.proto) proto given in serialized.

This op parses a serialized sequence example into a tuple of dictionaries mapping keys to Tensorand SparseTensor objects respectively. The first dictionary contains mappings for keys appearing incontext\_features, and the second dictionary contains mappings for keys appearing in sequence\_features.

At least one of context\_features and sequence\_features must be provided and non-empty.

The context\_features keys are associated with a SequenceExample as a whole, independent of time / frame. In contrast, the sequence\_features keys provide a way to access variable-length data within the FeatureList section of the SequenceExample proto. While the shapes of context\_features values are fixed with respect to frame, the frame dimension (the first dimension) of sequence\_features values may vary between SequenceExample protos, and even between feature\_list keys within the same SequenceExample.

context\_features contains VarLenFeature and FixedLenFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenFeature is mapped to a Tensor, of the specified type, shape, and default value.

sequence\_features contains VarLenFeature and FixedLenSequenceFeature objects. Each VarLenFeature is mapped to a SparseTensor, and each FixedLenSequenceFeature is mapped to a Tensor, each of the specified type. The shape will be (T,) + df.dense\_shape for FixedLenSequenceFeature df, where T is the length of the associated FeatureList in the SequenceExample. For instance, FixedLenSequenceFeature([]) yields a scalar 1-D Tensor of static shape [None] and dynamic shape [T], while FixedLenSequenceFeature([k]) (for int k >= 1) yields a 2-D matrix Tensor of static shape [None, k] and dynamic shape [T, k].

Each SparseTensor corresponding to sequence\_features represents a ragged vector. Its indices are [time, index], where time is the FeatureList entry and index is the value's index in the list of values associated with that time.

FixedLenFeature entries with a default\_value and FixedLenSequenceFeature entries with allow\_missing=True are optional; otherwise, we will fail if that Feature or FeatureList is missing from any example in serialized.

example\_name may contain a descriptive name for the corresponding serialized proto. This may be useful for debugging purposes, but it has no effect on the output. If not None, example\_name must be a scalar.

#### Args:

* **serialized**: A scalar (0-D Tensor) of type string, a single binary serialized SequenceExampleproto.
* **context\_features**: A dict mapping feature keys to FixedLenFeature or VarLenFeaturevalues. These features are associated with a SequenceExample as a whole.
* **sequence\_features**: A dict mapping feature keys to FixedLenSequenceFeature or VarLenFeature values. These features are associated with data within the FeatureListsection of the SequenceExample proto.
* **example\_name**: A scalar (0-D Tensor) of strings (optional), the name of the serialized proto.
* **name**: A name for this operation (optional).

#### Returns:

A tuple of two dicts, each mapping keys to Tensors and SparseTensors. The first dict contains the context key/values. The second dict contains the feature\_list key/values.

#### Raises:

* **ValueError**: if any feature is invalid.

# tf.io.parse\_tensor

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/parse_tensor#used_in_the_tutorials)

Transforms a serialized tensorflow.TensorProto proto into a Tensor.

### Aliases:

* tf.compat.v1.io.parse\_tensor
* tf.compat.v1.parse\_tensor
* tf.compat.v2.io.parse\_tensor
* tf.io.parse\_tensor

tf.io.parse\_tensor(  
    serialized,  
    out\_type,  
    name=None  
)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

### Used in the tutorials:

* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)

#### Args:

* **serialized**: A Tensor of type string. A scalar string containing a serialized TensorProto proto.
* **out\_type**: A [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType). The type of the serialized tensor. The provided type must match the type of the serialized tensor and no implicit conversion will take place.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type out\_type.

# tf.io.read\_file

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#aliases)
* [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/read_file#used_in_the_tutorials)

Reads and outputs the entire contents of the input filename.

### Aliases:

* tf.compat.v1.io.read\_file
* tf.compat.v1.read\_file
* tf.compat.v2.io.read\_file
* tf.io.read\_file

tf.io.read\_file(  
    filename,  
    name=None  
)

Defined in generated file: python/ops/gen\_io\_ops.py.

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load images with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/images)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

#### Args:

* **filename**: A Tensor of type string.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.serialize\_many\_sparse

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

Serialize N-minibatch SparseTensor into an [N, 3] Tensor.

### Aliases:

* tf.compat.v2.io.serialize\_many\_sparse
* tf.io.serialize\_many\_sparse

tf.io.serialize\_many\_sparse(  
    sp\_input,  
    out\_type=tf.dtypes.string,  
    name=None  
)

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

The SparseTensor must have rank R greater than 1, and the first dimension is treated as the minibatch dimension. Elements of the SparseTensor must be sorted in increasing order of this first dimension. The serialized SparseTensor objects going into each row of the output Tensor will have rank R-1.

The minibatch size N is extracted from sparse\_shape[0].

#### Args:

* **sp\_input**: The input rank R SparseTensor.
* **out\_type**: The dtype to use for serialization.
* **name**: A name prefix for the returned tensors (optional).

#### Returns:

A matrix (2-D Tensor) with N rows and 3 columns. Each column represents serialized SparseTensor's indices, values, and shape (respectively).

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.

# tf.io.serialize\_sparse

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

Serialize a SparseTensor into a 3-vector (1-D Tensor) object.

### Aliases:

* tf.compat.v2.io.serialize\_sparse
* tf.io.serialize\_sparse

tf.io.serialize\_sparse(  
    sp\_input,  
    out\_type=tf.dtypes.string,  
    name=None  
)

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

#### Args:

* **sp\_input**: The input SparseTensor.
* **out\_type**: The dtype to use for serialization.
* **name**: A name prefix for the returned tensors (optional).

#### Returns:

A 3-vector (1-D Tensor), with each column representing the serialized SparseTensor's indices, values, and shape (respectively).

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.

# tf.io.serialize\_tensor

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

Transforms a Tensor into a serialized TensorProto proto.

### Aliases:

* tf.compat.v1.io.serialize\_tensor
* tf.compat.v1.serialize\_tensor
* tf.compat.v2.io.serialize\_tensor
* tf.io.serialize\_tensor

tf.io.serialize\_tensor(  
    tensor,  
    name=None  
)

Defined in generated file: python/ops/gen\_parsing\_ops.py.

#### Args:

* **tensor**: A Tensor. A Tensor of type T.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.io.SparseFeature

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#top_of_page)
* [Class SparseFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#class_sparsefeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#properties)
  + [index\_key](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/SparseFeature#index_key)

## Class SparseFeature

Configuration for parsing a sparse input feature from an Example.

### Aliases:

* Class tf.compat.v1.SparseFeature
* Class tf.compat.v1.io.SparseFeature
* Class tf.compat.v2.io.SparseFeature
* Class tf.io.SparseFeature

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

Note, preferably use VarLenFeature (possibly in combination with a SequenceExample) in order to parse out SparseTensors instead of SparseFeature due to its simplicity.

Closely mimicking the SparseTensor that will be obtained by parsing an Example with a SparseFeature config, a SparseFeature contains a

* value\_key: The name of key for a Feature in the Example whose parsed Tensor will be the resulting SparseTensor.values.
* index\_key: A list of names - one for each dimension in the resulting SparseTensor whose indices[i][dim] indicating the position of the i-th value in the dim dimension will be equal to the i-th value in the Feature with key named index\_key[dim] in the Example.
* size: A list of ints for the resulting SparseTensor.dense\_shape.

For example, we can represent the following 2D SparseTensor

SparseTensor(indices=[[3, 1], [20, 0]],  
             values=[0.5, -1.0]  
             dense\_shape=[100, 3])

with an Example input proto

features {  
  feature { key: "val" value { float\_list { value: [ 0.5, -1.0 ] } } }  
  feature { key: "ix0" value { int64\_list { value: [ 3, 20 ] } } }  
  feature { key: "ix1" value { int64\_list { value: [ 1, 0 ] } } }  
}

and SparseFeature config with 2 index\_keys

SparseFeature(index\_key=["ix0", "ix1"],  
              value\_key="val",  
              dtype=tf.float32,  
              size=[100, 3])

#### Fields:

* **index\_key**: A single string name or a list of string names of index features. For each key the underlying feature's type must be int64 and its length must always match that of the value\_key feature. To represent SparseTensors with a dense\_shape of rank higher than 1 a list of length rank should be used.
* **value\_key**: Name of value feature. The underlying feature's type must be dtype and its length must always match that of all the index\_keys' features.
* **dtype**: Data type of the value\_key feature.
* **size**: A Python int or list thereof specifying the dense shape. Should be a list if and only if index\_key is a list. In that case the list must be equal to the length of index\_key. Each for each entry i all values in the index\_key[i] feature must be in [0, size[i]).
* **already\_sorted**: A Python boolean to specify whether the values in value\_key are already sorted by their index position. If so skip sorting. False by default (optional).

## Properties

### index\_key

### value\_key

### dtype

### size

### already\_sorted

# tf.io.TFRecordOptions

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#top_of_page)
* [Class TFRecordOptions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#class_tfrecordoptions)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#methods)
  + [get\_compression\_type\_string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#get_compression_type_string)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordOptions#class_members)

## Class TFRecordOptions

Options used for manipulating TFRecord files.

### Aliases:

* Class tf.compat.v1.io.TFRecordOptions
* Class tf.compat.v1.python\_io.TFRecordOptions
* Class tf.compat.v2.io.TFRecordOptions
* Class tf.io.TFRecordOptions

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

## \_\_init\_\_

\_\_init\_\_(  
    compression\_type=None,  
    flush\_mode=None,  
    input\_buffer\_size=None,  
    output\_buffer\_size=None,  
    window\_bits=None,  
    compression\_level=None,  
    compression\_method=None,  
    mem\_level=None,  
    compression\_strategy=None  
)

Creates a TFRecordOptions instance.

Options only effect TFRecordWriter when compression\_type is not None. Documentation, details, and defaults can be found in [zlib\_compression\_options.h](https://www.tensorflow.org/code/tensorflow/core/lib/io/zlib_compression_options.h) and in the [zlib manual](http://www.zlib.net/manual.html). Leaving an option as None allows C++ to set a reasonable default.

#### Args:

* **compression\_type**: "GZIP", "ZLIB", or "" (no compression).
* **flush\_mode**: flush mode or None, Default: Z\_NO\_FLUSH.
* **input\_buffer\_size**: int or None.
* **output\_buffer\_size**: int or None.
* **window\_bits**: int or None.
* **compression\_level**: 0 to 9, or None.
* **compression\_method**: compression method or None.
* **mem\_level**: 1 to 9, or None.
* **compression\_strategy**: strategy or None. Default: Z\_DEFAULT\_STRATEGY.

#### Returns:

A TFRecordOptions object.

#### Raises:

* **ValueError**: If compression\_type is invalid.

## Methods

### get\_compression\_type\_string

@classmethod  
get\_compression\_type\_string(  
    cls,  
    options  
)

Convert various option types to a unified string.

#### Args:

* **options**: TFRecordOption, TFRecordCompressionType, or string.

#### Returns:

Compression type as string (e.g. 'ZLIB', 'GZIP', or '').

#### Raises:

* **ValueError**: If compression\_type is invalid.

## Class Members

* compression\_type\_map

# tf.io.TFRecordWriter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#top_of_page)
* [Class TFRecordWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#class_tfrecordwriter)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#aliases)
  + [Used in the tutorials:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#used_in_the_tutorials)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/TFRecordWriter#__init__)

## Class TFRecordWriter

A class to write records to a TFRecords file.

### Aliases:

* Class tf.compat.v1.io.TFRecordWriter
* Class tf.compat.v1.python\_io.TFRecordWriter
* Class tf.compat.v2.io.TFRecordWriter
* Class tf.io.TFRecordWriter

Defined in [python/lib/io/tf\_record.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/lib/io/tf_record.py).

### Used in the tutorials:

* [Using TFRecords and tf.Example](https://www.tensorflow.org/beta/tutorials/load_data/tf_records)

This class implements \_\_enter\_\_ and \_\_exit\_\_, and can be used in with blocks like a normal file.

## \_\_init\_\_

\_\_init\_\_(  
    path,  
    options=None  
)

Opens file path and creates a TFRecordWriter writing to it.

#### Args:

* **path**: The path to the TFRecords file.
* **options**: (optional) String specifying compression type, TFRecordCompressionType, or TFRecordOptions object.

#### Raises:

* **IOError**: If path cannot be opened for writing.
* **ValueError**: If valid compression\_type can't be determined from options.

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

Enter a with block.

### \_\_exit\_\_

\_\_exit\_\_(  
    unused\_type,  
    unused\_value,  
    unused\_traceback  
)

Exit a with block, closing the file.

### close

close()

Close the file.

### flush

flush()

Flush the file.

### write

write(record)

Write a string record to the file.

#### Args:

* **record**: str

# tf.io.VarLenFeature

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#top_of_page)
* [Class VarLenFeature](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#class_varlenfeature)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#aliases)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#properties)
  + [dtype](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/VarLenFeature#dtype)

## Class VarLenFeature

Configuration for parsing a variable-length input feature.

### Aliases:

* Class tf.compat.v1.VarLenFeature
* Class tf.compat.v1.io.VarLenFeature
* Class tf.compat.v2.io.VarLenFeature
* Class tf.io.VarLenFeature

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

#### Fields:

* **dtype**: Data type of input.

## Properties

### dtype

# tf.io.write\_file

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

Writes contents to the file at input filename. Creates file and recursively

### Aliases:

* tf.compat.v1.io.write\_file
* tf.compat.v1.write\_file
* tf.compat.v2.io.write\_file
* tf.io.write\_file

tf.io.write\_file(  
    filename,  
    contents,  
    name=None  
)

Defined in generated file: python/ops/gen\_io\_ops.py.

creates directory if not existing.

#### Args:

* **filename**: A Tensor of type string. scalar. The name of the file to which we write the contents.
* **contents**: A Tensor of type string. scalar. The content to be written to the output file.
* **name**: A name for the operation (optional).

#### Returns:

The created Operation.

# tf.io.write\_graph

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

Writes a graph proto to a file.

### Aliases:

* tf.compat.v1.io.write\_graph
* tf.compat.v1.train.write\_graph
* tf.compat.v2.io.write\_graph
* tf.io.write\_graph

tf.io.write\_graph(  
    graph\_or\_graph\_def,  
    logdir,  
    name,  
    as\_text=True  
)

Defined in [python/framework/graph\_io.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/framework/graph_io.py).

The graph is written as a text proto unless as\_text is False.

v = tf.Variable(0, name='my\_variable')  
sess = tf.compat.v1.Session()  
tf.io.write\_graph(sess.graph\_def, '/tmp/my-model', 'train.pbtxt')

or

v = tf.Variable(0, name='my\_variable')  
sess = tf.compat.v1.Session()  
tf.io.write\_graph(sess.graph, '/tmp/my-model', 'train.pbtxt')

#### Args:

* **graph\_or\_graph\_def**: A Graph or a GraphDef protocol buffer.
* **logdir**: Directory where to write the graph. This can refer to remote filesystems, such as Google Cloud Storage (GCS).
* **name**: Filename for the graph.
* **as\_text**: If True, writes the graph as an ASCII proto.

#### Returns:

The path of the output proto file.

Module: tf.io.gfile

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

Public API for tf.io.gfile namespace.

Classes

[class GFile](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/GFile): File I/O wrappers without thread locking.

Functions

[copy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/copy): Copies data from src to dst.

[exists(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/exists): Determines whether a path exists or not.

[glob(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/glob): Returns a list of files that match the given pattern(s).

[isdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/isdir): Returns whether the path is a directory or not.

[listdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/listdir): Returns a list of entries contained within a directory.

[makedirs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/makedirs): Creates a directory and all parent/intermediate directories.

[mkdir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/mkdir): Creates a directory with the name given by 'path'.

[remove(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/remove): Deletes the path located at 'path'.

[rename(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/rename): Rename or move a file / directory.

[rmtree(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/rmtree): Deletes everything under path recursively.

[stat(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/stat): Returns file statistics for a given path.

[walk(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/gfile/walk): Recursive directory tree generator for directories.