Module: tf.nn / tf.compat.v1.nn

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

Wrappers for primitive Neural Net (NN) Operations.

Modules

[rnn\_cell](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/rnn_cell) module: Module for constructing RNN Cells.

Functions

[all\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/all_candidate_sampler): Generate the set of all classes.

[atrous\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d): Atrous convolution (a.k.a. convolution with holes or dilated convolution).

[atrous\_conv2d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d_transpose): The transpose of atrous\_conv2d.

[avg\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/avg_pool): Performs the average pooling on the input.

[avg\_pool1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool1d): Performs the average pooling on the input.

[avg\_pool2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/avg_pool): Performs the average pooling on the input.

[avg\_pool3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool3d): Performs the average pooling on the input.

[avg\_pool\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool): Performs the avg pooling on the input.

[batch\_norm\_with\_global\_normalization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/batch_norm_with_global_normalization): Batch normalization.

[batch\_normalization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/batch_normalization): Batch normalization.

[bias\_add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/bias_add): Adds bias to value.

[bidirectional\_dynamic\_rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/bidirectional_dynamic_rnn): Creates a dynamic version of bidirectional recurrent neural network. (deprecated)

[collapse\_repeated(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/collapse_repeated): Merge repeated labels into single labels.

[compute\_accidental\_hits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/compute_accidental_hits): Compute the position ids in sampled\_candidates matching true\_classes.

[compute\_average\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/compute_average_loss): Scales per-example losses with sample\_weights and computes their average.

[conv1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv1d): Computes a 1-D convolution given 3-D input and filter tensors. (deprecated argument values) (deprecated argument values)

[conv1d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv1d_transpose): The transpose of conv1d.

[conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv2d): Computes a 2-D convolution given 4-D input and filter tensors.

[conv2d\_backprop\_filter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv2d_backprop_filter): Computes the gradients of convolution with respect to the filter.

[conv2d\_backprop\_input(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv2d_backprop_input): Computes the gradients of convolution with respect to the input.

[conv2d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv2d_transpose): The transpose of conv2d.

[conv3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv3d): Computes a 3-D convolution given 5-D input and filter tensors.

[conv3d\_backprop\_filter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv3d_backprop_filter): Computes the gradients of 3-D convolution with respect to the filter.

[conv3d\_backprop\_filter\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv3d_backprop_filter): Computes the gradients of 3-D convolution with respect to the filter.

[conv3d\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/conv3d_transpose): The transpose of conv3d.

[conv\_transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv_transpose): The transpose of convolution.

[convolution(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/convolution): Computes sums of N-D convolutions (actually cross-correlation).

[crelu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/crelu): Computes Concatenated ReLU.

[ctc\_beam\_search\_decoder(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/ctc_beam_search_decoder): Performs beam search decoding on the logits given in input.

[ctc\_beam\_search\_decoder\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_beam_search_decoder): Performs beam search decoding on the logits given in input.

[ctc\_greedy\_decoder(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_greedy_decoder): Performs greedy decoding on the logits given in input (best path).

[ctc\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/ctc_loss): Computes the CTC (Connectionist Temporal Classification) Loss.

[ctc\_loss\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_loss): Computes CTC (Connectionist Temporal Classification) loss.

[ctc\_unique\_labels(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_unique_labels): Get unique labels and indices for batched labels for [tf.nn.ctc\_loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_loss).

[depth\_to\_space(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/depth_to_space): DepthToSpace for tensors of type T.

[depthwise\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/depthwise_conv2d): Depthwise 2-D convolution.

[depthwise\_conv2d\_backprop\_filter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_filter): Computes the gradients of depthwise convolution with respect to the filter.

[depthwise\_conv2d\_backprop\_input(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_input): Computes the gradients of depthwise convolution with respect to the input.

[depthwise\_conv2d\_native(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/depthwise_conv2d_native): Computes a 2-D depthwise convolution given 4-D input and filter tensors.

[depthwise\_conv2d\_native\_backprop\_filter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_filter): Computes the gradients of depthwise convolution with respect to the filter.

[depthwise\_conv2d\_native\_backprop\_input(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_input): Computes the gradients of depthwise convolution with respect to the input.

[dilation2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/dilation2d): Computes the grayscale dilation of 4-D input and 3-D filter tensors.

[dropout(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/dropout): Computes dropout. (deprecated arguments)

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

[elu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/elu): Computes exponential linear: exp(features) - 1 if < 0, features otherwise.

[embedding\_lookup(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/embedding_lookup): Looks up ids in a list of embedding tensors.

[embedding\_lookup\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/embedding_lookup_sparse): Computes embeddings for the given ids and weights.

[erosion2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/erosion2d): Computes the grayscale erosion of 4-D value and 3-D kernel tensors.

[fixed\_unigram\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/fixed_unigram_candidate_sampler): Samples a set of classes using the provided (fixed) base distribution.

[fractional\_avg\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/fractional_avg_pool): Performs fractional average pooling on the input. (deprecated)

[fractional\_max\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/fractional_max_pool): Performs fractional max pooling on the input. (deprecated)

[fused\_batch\_norm(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/fused_batch_norm): Batch normalization.

[in\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/in_top_k): Says whether the targets are in the top K predictions.

[l2\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/l2_loss): L2 Loss.

[l2\_normalize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/linalg/l2_normalize): Normalizes along dimension axis using an L2 norm. (deprecated arguments)

[leaky\_relu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/leaky_relu): Compute the Leaky ReLU activation function.

[learned\_unigram\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/learned_unigram_candidate_sampler): Samples a set of classes from a distribution learned during training.

[local\_response\_normalization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/local_response_normalization): Local Response Normalization.

[log\_poisson\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/log_poisson_loss): Computes log Poisson loss given log\_input.

[log\_softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/log_softmax): Computes log softmax activations. (deprecated arguments)

[log\_uniform\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/log_uniform_candidate_sampler): Samples a set of classes using a log-uniform (Zipfian) base distribution.

[lrn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/local_response_normalization): Local Response Normalization.

[max\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/max_pool): Performs the max pooling on the input.

[max\_pool1d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool1d): Performs the max pooling on the input.

[max\_pool2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool2d): Performs the max pooling on the input.

[max\_pool3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool3d): Performs the max pooling on the input.

[max\_pool\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool): Performs the max pooling on the input.

[max\_pool\_with\_argmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/max_pool_with_argmax): Performs max pooling on the input and outputs both max values and indices.

[moments(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/moments): Calculate the mean and variance of x.

[nce\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/nce_loss): Computes and returns the noise-contrastive estimation training loss.

[normalize\_moments(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/normalize_moments): Calculate the mean and variance of based on the sufficient statistics.

[pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/pool): Performs an N-D pooling operation.

[quantized\_avg\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/quantized_avg_pool): Produces the average pool of the input tensor for quantized types.

[quantized\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/quantized_conv2d): Computes a 2D convolution given quantized 4D input and filter tensors.

[quantized\_max\_pool(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/quantized_max_pool): Produces the max pool of the input tensor for quantized types.

[quantized\_relu\_x(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/quantized_relu_x): Computes Quantized Rectified Linear X: min(max(features, 0), max\_value)

[raw\_rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/raw_rnn): Creates an RNN specified by RNNCell cell and loop function loop\_fn.

[relu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/relu): Computes rectified linear: max(features, 0).

[relu6(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/relu6): Computes Rectified Linear 6: min(max(features, 0), 6).

[relu\_layer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/relu_layer): Computes Relu(x \* weight + biases).

[safe\_embedding\_lookup\_sparse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/safe_embedding_lookup_sparse): Lookup embedding results, accounting for invalid IDs and empty features.

[sampled\_softmax\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/sampled_softmax_loss): Computes and returns the sampled softmax training loss.

[scale\_regularization\_loss(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/scale_regularization_loss): Scales the sum of the given regularization losses by number of replicas.

[selu(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/selu): Computes scaled exponential linear: scale \* alpha \* (exp(features) - 1)

[separable\_conv2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/separable_conv2d): 2-D convolution with separable filters.

[sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid): Computes sigmoid of x element-wise.

[sigmoid\_cross\_entropy\_with\_logits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/sigmoid_cross_entropy_with_logits): Computes sigmoid cross entropy given logits.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/math/softmax): Computes softmax activations. (deprecated arguments)

[softmax\_cross\_entropy\_with\_logits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/softmax_cross_entropy_with_logits): Computes softmax cross entropy between logitsand labels. (deprecated)

[softmax\_cross\_entropy\_with\_logits\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/softmax_cross_entropy_with_logits_v2): Computes softmax cross entropy between logits and labels. (deprecated arguments)

[softplus(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/softplus): Computes softplus: log(exp(features) + 1).

[softsign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softsign): Computes softsign: features / (abs(features) + 1).

[space\_to\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/space_to_batch): SpaceToBatch for 4-D tensors of type T.

[space\_to\_depth(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/space_to_depth): SpaceToDepth for tensors of type T.

[sparse\_softmax\_cross\_entropy\_with\_logits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/sparse_softmax_cross_entropy_with_logits): Computes sparse softmax cross entropy between logits and labels.

[static\_bidirectional\_rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/static_bidirectional_rnn): Creates a bidirectional recurrent neural network. (deprecated)

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

[static\_state\_saving\_rnn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/static_state_saving_rnn): RNN that accepts a state saver for time-truncated RNN calculation. (deprecated)

[sufficient\_statistics(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/sufficient_statistics): Calculate the sufficient statistics for the mean and variance of x.

[tanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh): Computes hyperbolic tangent of x element-wise.

[top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/top_k): Finds values and indices of the k largest entries for the last dimension.

[uniform\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/uniform_candidate_sampler): Samples a set of classes using a uniform base distribution.

[weighted\_cross\_entropy\_with\_logits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/weighted_cross_entropy_with_logits): Computes a weighted cross entropy. (deprecated arguments)

[weighted\_moments(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/weighted_moments): Returns the frequency-weighted mean and variance of x.

[with\_space\_to\_batch(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/with_space_to_batch): Performs op on the space-to-batch representation of input.

[xw\_plus\_b(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/nn/xw_plus_b): Computes matmul(x, weights) + biases.

[zero\_fraction(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zero_fraction): Returns the fraction of zeros in value.

Other Members

* swish

# tf.nn.atrous\_conv2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d#aliases)

Atrous convolution (a.k.a. convolution with holes or dilated convolution).

### Aliases:

* tf.compat.v1.nn.atrous\_conv2d
* tf.compat.v2.nn.atrous\_conv2d
* tf.nn.atrous\_conv2d

tf.nn.atrous\_conv2d(  
    value,  
    filters,  
    rate,  
    padding,  
    name=None  
)

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

This function is a simpler wrapper around the more general [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution), and exists only for backwards compatibility. You can use [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) to perform 1-D, 2-D, or 3-D atrous convolution.

Computes a 2-D atrous convolution, also known as convolution with holes or dilated convolution, given 4-D value and filters tensors. If the rate parameter is equal to one, it performs regular 2-D convolution. If the rate parameter is greater than one, it performs convolution with holes, sampling the input values every rate pixels in the height and width dimensions. This is equivalent to convolving the input with a set of upsampled filters, produced by inserting rate - 1 zeros between two consecutive values of the filters along the height and width dimensions, hence the name atrous convolution or convolution with holes (the French word trous means holes in English).

#### More specifically:

output[batch, height, width, out\_channel] =  
    sum\_{dheight, dwidth, in\_channel} (  
        filters[dheight, dwidth, in\_channel, out\_channel] \*  
        value[batch, height + rate\*dheight, width + rate\*dwidth, in\_channel]  
    )

Atrous convolution allows us to explicitly control how densely to compute feature responses in fully convolutional networks. Used in conjunction with bilinear interpolation, it offers an alternative to conv2d\_transpose in dense prediction tasks such as semantic image segmentation, optical flow computation, or depth estimation. It also allows us to effectively enlarge the field of view of filters without increasing the number of parameters or the amount of computation.

For a description of atrous convolution and how it can be used for dense feature extraction, please see: [Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs](http://arxiv.org/abs/1412.7062). The same operation is investigated further in [Multi-Scale Context Aggregation by Dilated Convolutions](http://arxiv.org/abs/1511.07122). Previous works that effectively use atrous convolution in different ways are, among others, [OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks](http://arxiv.org/abs/1312.6229) and [Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks](http://arxiv.org/abs/1302.1700). Atrous convolution is also closely related to the so-called noble identities in multi-rate signal processing.

There are many different ways to implement atrous convolution (see the refs above). The implementation here reduces

    atrous\_conv2d(value, filters, rate, padding=padding)

to the following three operations:

    paddings = ...  
    net = space\_to\_batch(value, paddings, block\_size=rate)  
    net = conv2d(net, filters, strides=[1, 1, 1, 1], padding="VALID")  
    crops = ...  
    net = batch\_to\_space(net, crops, block\_size=rate)

Advanced usage. Note the following optimization: A sequence of atrous\_conv2d operations with identical rate parameters, 'SAME' padding, and filters with odd heights/ widths:

    net = atrous\_conv2d(net, filters1, rate, padding="SAME")  
    net = atrous\_conv2d(net, filters2, rate, padding="SAME")  
    ...  
    net = atrous\_conv2d(net, filtersK, rate, padding="SAME")

can be equivalently performed cheaper in terms of computation and memory as:

    pad = ...  # padding so that the input dims are multiples of rate  
    net = space\_to\_batch(net, paddings=pad, block\_size=rate)  
    net = conv2d(net, filters1, strides=[1, 1, 1, 1], padding="SAME")  
    net = conv2d(net, filters2, strides=[1, 1, 1, 1], padding="SAME")  
    ...  
    net = conv2d(net, filtersK, strides=[1, 1, 1, 1], padding="SAME")  
    net = batch\_to\_space(net, crops=pad, block\_size=rate)

because a pair of consecutive space\_to\_batch and batch\_to\_space ops with the same block\_size cancel out when their respective paddings and crops inputs are identical.

#### Args:

* **value**: A 4-D Tensor of type float. It needs to be in the default "NHWC" format. Its shape is [batch, in\_height, in\_width, in\_channels].
* **filters**: A 4-D Tensor with the same type as value and shape [filter\_height, filter\_width, in\_channels, out\_channels]. filters' in\_channels dimension must match that of value. Atrous convolution is equivalent to standard convolution with upsampled filters with effective height filter\_height + (filter\_height - 1) \* (rate - 1) and effective width filter\_width + (filter\_width - 1) \* (rate - 1), produced by inserting rate - 1 zeros along consecutive elements across the filters' spatial dimensions.
* **rate**: A positive int32. The stride with which we sample input values across the height and width dimensions. Equivalently, the rate by which we upsample the filter values by inserting zeros across the height and width dimensions. In the literature, the same parameter is sometimes called input stride or dilation.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm.
* **name**: Optional name for the returned tensor.

#### Returns:

A Tensor with the same type as value. Output shape with 'VALID' padding is:

[batch, height - 2 \* (filter\_width - 1),  
 width - 2 \* (filter\_height - 1), out\_channels].

Output shape with 'SAME' padding is:

[batch, height, width, out\_channels].

#### Raises:

# ValueError: If input/output depth does not match filters' shape, or if padding is other than 'VALID' or 'SAME'. tf.nn.atrous\_conv2d\_transpose

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d_transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/atrous_conv2d_transpose#aliases)

The transpose of atrous\_conv2d.

### Aliases:

* tf.compat.v1.nn.atrous\_conv2d\_transpose
* tf.compat.v2.nn.atrous\_conv2d\_transpose
* tf.nn.atrous\_conv2d\_transpose

tf.nn.atrous\_conv2d\_transpose(  
    value,  
    filters,  
    output\_shape,  
    rate,  
    padding,  
    name=None  
)

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

This operation is sometimes called "deconvolution" after [Deconvolutional Networks](https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf), but is really the transpose (gradient) of atrous\_conv2d rather than an actual deconvolution.

#### Args:

* **value**: A 4-D Tensor of type float. It needs to be in the default NHWC format. Its shape is [batch, in\_height, in\_width, in\_channels].
* **filters**: A 4-D Tensor with the same type as value and shape [filter\_height, filter\_width, out\_channels, in\_channels]. filters' in\_channels dimension must match that of value. Atrous convolution is equivalent to standard convolution with upsampled filters with effective height filter\_height + (filter\_height - 1) \* (rate - 1) and effective width filter\_width + (filter\_width - 1) \* (rate - 1), produced by inserting rate - 1 zeros along consecutive elements across the filters' spatial dimensions.
* **output\_shape**: A 1-D Tensor of shape representing the output shape of the deconvolution op.
* **rate**: A positive int32. The stride with which we sample input values across the height and width dimensions. Equivalently, the rate by which we upsample the filter values by inserting zeros across the height and width dimensions. In the literature, the same parameter is sometimes called input stride or dilation.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm.
* **name**: Optional name for the returned tensor.

#### Returns:

A Tensor with the same type as value.

#### Raises:

* **ValueError**: If input/output depth does not match filters' shape, or if padding is other than 'VALID' or 'SAME', or if the rate is less than one, or if the output\_shape is not a tensor with 4 elements.

# tf.nn.avg\_pool

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

Performs the avg pooling on the input.

### Aliases:

* tf.compat.v1.nn.avg\_pool\_v2
* tf.compat.v2.nn.avg\_pool
* tf.nn.avg\_pool

tf.nn.avg\_pool(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format=None,  
    name=None  
)

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

Each entry in output is the mean of the corresponding size ksize window in value.

#### Args:

* **input**: Tensor of rank N+2, of shape [batch\_size] + input\_spatial\_shape + [num\_channels] if data\_format does not start with "NC" (default), or [batch\_size, num\_channels] + input\_spatial\_shape if data\_format starts with "NC". Pooling happens over the spatial dimensions only.
* **ksize**: An int or list of ints that has length 1, N or N+2. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, N or N+2. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. Specifies the channel dimension. For N=1 it can be either "NWC" (default) or "NCW", for N=2 it can be either "NHWC" (default) or "NCHW" and for N=3 either "NDHWC" (default) or "NCDHW".
* **name**: Optional name for the operation.

#### Returns:

A Tensor of format specified by data\_format. The average pooled output tensor.

# tf.nn.avg\_pool1d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool1d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool1d#aliases)

Performs the average pooling on the input.

### Aliases:

* tf.compat.v1.nn.avg\_pool1d
* tf.compat.v2.nn.avg\_pool1d
* tf.nn.avg\_pool1d

tf.nn.avg\_pool1d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NWC',  
    name=None  
)

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

Each entry in output is the mean of the corresponding size ksize window in value.

Note internally this op reshapes and uses the underlying 2d operation.

#### Args:

* **input**: A 3-D Tensor of the format specified by data\_format.
* **ksize**: An int or list of ints that has length 1 or 3. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1 or 3. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: An optional string from: "NWC", "NCW". Defaults to "NWC".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.avg\_pool1d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool1d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool1d#aliases)

Performs the average pooling on the input.

### Aliases:

* tf.compat.v1.nn.avg\_pool1d
* tf.compat.v2.nn.avg\_pool1d
* tf.nn.avg\_pool1d

tf.nn.avg\_pool1d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NWC',  
    name=None  
)

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

Each entry in output is the mean of the corresponding size ksize window in value.

Note internally this op reshapes and uses the underlying 2d operation.

#### Args:

* **input**: A 3-D Tensor of the format specified by data\_format.
* **ksize**: An int or list of ints that has length 1 or 3. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1 or 3. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: An optional string from: "NWC", "NCW". Defaults to "NWC".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.avg\_pool3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/avg_pool3d#aliases)

Performs the average pooling on the input.

### Aliases:

* tf.compat.v1.nn.avg\_pool3d
* tf.compat.v2.nn.avg\_pool3d
* tf.nn.avg\_pool3d

tf.nn.avg\_pool3d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NDHWC',  
    name=None  
)

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

Each entry in output is the mean of the corresponding size ksize window in value.

#### Args:

* **input**: A 5-D Tensor of shape [batch, height, width, channels] and type float32, float64, qint8, quint8, or qint32.
* **ksize**: An int or list of ints that has length 1, 3 or 5. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, 3 or 5. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. 'NDHWC' and 'NCDHW' are supported.
* **name**: Optional name for the operation.

#### Returns:

A Tensor with the same type as value. The average pooled output tensor.

# tf.nn.batch\_normalization

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

Batch normalization.

### Aliases:

* tf.compat.v1.nn.batch\_normalization
* tf.compat.v2.nn.batch\_normalization
* tf.nn.batch\_normalization

tf.nn.batch\_normalization(  
    x,  
    mean,  
    variance,  
    offset,  
    scale,  
    variance\_epsilon,  
    name=None  
)

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

Normalizes a tensor by mean and variance, and applies (optionally) a scale γ to it, as well as an offset β:

γ(x−μ)σ+β

mean, variance, offset and scale are all expected to be of one of two shapes:

* In all generality, they can have the same number of dimensions as the input x, with identical sizes as x for the dimensions that are not normalized over (the 'depth' dimension(s)), and dimension 1 for the others which are being normalized over. mean and variance in this case would typically be the outputs of tf.nn.moments(..., keep\_dims=True) during training, or running averages thereof during inference.
* In the common case where the 'depth' dimension is the last dimension in the input tensor x, they may be one dimensional tensors of the same size as the 'depth' dimension. This is the case for example for the common [batch, depth] layout of fully-connected layers, and [batch, height, width, depth] for convolutions. mean and variance in this case would typically be the outputs of tf.nn.moments(..., keep\_dims=False) during training, or running averages thereof during inference.

See Source: [Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift; S. Ioffe, C. Szegedy](http://arxiv.org/abs/1502.03167).

#### Args:

* **x**: Input Tensor of arbitrary dimensionality.
* **mean**: A mean Tensor.
* **variance**: A variance Tensor.
* **offset**: An offset Tensor, often denoted β in equations, or None. If present, will be added to the normalized tensor.
* **scale**: A scale Tensor, often denoted γ in equations, or None. If present, the scale is applied to the normalized tensor.
* **variance\_epsilon**: A small float number to avoid dividing by 0.
* **name**: A name for this operation (optional).

#### Returns:

the normalized, scaled, offset tensor.

# tf.nn.batch\_norm\_with\_global\_normalization

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

Batch normalization.

### Aliases:

* tf.compat.v2.nn.batch\_norm\_with\_global\_normalization
* tf.nn.batch\_norm\_with\_global\_normalization

tf.nn.batch\_norm\_with\_global\_normalization(  
    input,  
    mean,  
    variance,  
    beta,  
    gamma,  
    variance\_epsilon,  
    scale\_after\_normalization,  
    name=None  
)

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

This op is deprecated. See [tf.nn.batch\_normalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/batch_normalization).

#### Args:

* **input**: A 4D input Tensor.
* **mean**: A 1D mean Tensor with size matching the last dimension of t. This is the first output from tf.nn.moments, or a saved moving average thereof.
* **variance**: A 1D variance Tensor with size matching the last dimension of t. This is the second output from tf.nn.moments, or a saved moving average thereof.
* **beta**: A 1D beta Tensor with size matching the last dimension of t. An offset to be added to the normalized tensor.
* **gamma**: A 1D gamma Tensor with size matching the last dimension of t. If "scale\_after\_normalization" is true, this tensor will be multiplied with the normalized tensor.
* **variance\_epsilon**: A small float number to avoid dividing by 0.
* **scale\_after\_normalization**: A bool indicating whether the resulted tensor needs to be multiplied with gamma.
* **name**: A name for this operation (optional).

#### Returns:

A batch-normalized t.

# tf.nn.bias\_add

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

Adds bias to value.

### Aliases:

* tf.compat.v1.nn.bias\_add
* tf.compat.v2.nn.bias\_add
* tf.nn.bias\_add

tf.nn.bias\_add(  
    value,  
    bias,  
    data\_format=None,  
    name=None  
)

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

This is (mostly) a special case of [tf.add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add) where bias is restricted to 1-D. Broadcasting is supported, so value may have any number of dimensions. Unlike [tf.add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add), the type of bias is allowed to differ from value in the case where both types are quantized.

#### Args:

* **value**: A Tensor with type float, double, int64, int32, uint8, int16, int8, complex64, or complex128.
* **bias**: A 1-D Tensor with size matching the last dimension of value. Must be the same type as value unless value is a quantized type, in which case a different quantized type may be used.
* **data\_format**: A string. 'N...C' and 'NC...' are supported.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as value.

# tf.nn.collapse\_repeated

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

Merge repeated labels into single labels.

### Aliases:

* tf.compat.v1.nn.collapse\_repeated
* tf.compat.v2.nn.collapse\_repeated
* tf.nn.collapse\_repeated

tf.nn.collapse\_repeated(  
    labels,  
    seq\_length,  
    name=None  
)

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

#### Args:

* **labels**: Tensor of shape [batch, max value in seq\_length]
* **seq\_length**: Tensor of shape [batch], sequence length of each batch element.
* **name**: A name for this Op. Defaults to "collapse\_repeated\_labels".

#### Returns:

A tuple (collapsed\_labels, new\_seq\_length) where

* **collapsed\_labels**: Tensor of shape [batch, max\_seq\_length] with repeated labels collapsed and padded to max\_seq\_length, eg: [[A, A, B, B, A], [A, B, C, D, E]] => [[A, B, A, 0, 0], [A, B, C, D, E]]
* **new\_seq\_length**: int tensor of shape [batch] with new sequence lengths.

# tf.nn.compute\_accidental\_hits

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

Compute the position ids in sampled\_candidates matching true\_classes.

### Aliases:

* tf.compat.v1.nn.compute\_accidental\_hits
* tf.compat.v2.nn.compute\_accidental\_hits
* tf.nn.compute\_accidental\_hits

tf.nn.compute\_accidental\_hits(  
    true\_classes,  
    sampled\_candidates,  
    num\_true,  
    seed=None,  
    name=None  
)

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

In Candidate Sampling, this operation facilitates virtually removing sampled classes which happen to match target classes. This is done in Sampled Softmax and Sampled Logistic.

See our [Candidate Sampling Algorithms Reference](http://www.tensorflow.org/extras/candidate_sampling.pdf).

We presuppose that the sampled\_candidates are unique.

We call it an 'accidental hit' when one of the target classes matches one of the sampled classes. This operation reports accidental hits as triples (index, id, weight), where index represents the row number in true\_classes, id represents the position in sampled\_candidates, and weight is -FLOAT\_MAX.

The result of this op should be passed through a sparse\_to\_dense operation, then added to the logits of the sampled classes. This removes the contradictory effect of accidentally sampling the true target classes as noise classes for the same example.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. The sampled\_candidates output of CandidateSampler.
* **num\_true**: An int. The number of target classes per training example.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **indices**: A Tensor of type int32 and shape [num\_accidental\_hits]. Values indicate rows in true\_classes.
* **ids**: A Tensor of type int64 and shape [num\_accidental\_hits]. Values indicate positions in sampled\_candidates.
* **weights**: A Tensor of type float and shape [num\_accidental\_hits]. Each value is -FLOAT\_MAX.

# tf.nn.compute\_average\_loss

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

Scales per-example losses with sample\_weights and computes their average.

### Aliases:

* tf.compat.v1.nn.compute\_average\_loss
* tf.compat.v2.nn.compute\_average\_loss
* tf.nn.compute\_average\_loss

tf.nn.compute\_average\_loss(  
    per\_example\_loss,  
    sample\_weight=None,  
    global\_batch\_size=None  
)

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

### Used in the tutorials:

* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Usage with distribution strategy and custom training loop:

with strategy.scope():  
  def compute\_loss(labels, predictions, sample\_weight=None):  
  
    # If you are using a `Loss` class instead, set reduction to `NONE` so that  
    # we can do the reduction afterwards and divide by global batch size.  
    per\_example\_loss = tf.keras.losses.sparse\_categorical\_crossentropy(  
        labels, predictions)  
  
    # Compute loss that is scaled by sample\_weight and by global batch size.  
    return tf.compute\_average\_loss(  
        per\_example\_loss,  
        sample\_weight=sample\_weight,  
        global\_batch\_size=GLOBAL\_BATCH\_SIZE)

#### Args:

* **per\_example\_loss**: Per-example loss.
* **sample\_weight**: Optional weighting for each example.
* **global\_batch\_size**: Optional global batch size value. Defaults to (size of first dimension of losses) \* (number of replicas).

#### Returns:

Scalar loss value.

# tf.nn.compute\_average\_loss

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

Scales per-example losses with sample\_weights and computes their average.

### Aliases:

* tf.compat.v1.nn.compute\_average\_loss
* tf.compat.v2.nn.compute\_average\_loss
* tf.nn.compute\_average\_loss

tf.nn.compute\_average\_loss(  
    per\_example\_loss,  
    sample\_weight=None,  
    global\_batch\_size=None  
)

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

### Used in the tutorials:

* [tf.distribute.Strategy with training loops](https://www.tensorflow.org/beta/tutorials/distribute/training_loops)

Usage with distribution strategy and custom training loop:

with strategy.scope():  
  def compute\_loss(labels, predictions, sample\_weight=None):  
  
    # If you are using a `Loss` class instead, set reduction to `NONE` so that  
    # we can do the reduction afterwards and divide by global batch size.  
    per\_example\_loss = tf.keras.losses.sparse\_categorical\_crossentropy(  
        labels, predictions)  
  
    # Compute loss that is scaled by sample\_weight and by global batch size.  
    return tf.compute\_average\_loss(  
        per\_example\_loss,  
        sample\_weight=sample\_weight,  
        global\_batch\_size=GLOBAL\_BATCH\_SIZE)

#### Args:

* **per\_example\_loss**: Per-example loss.
* **sample\_weight**: Optional weighting for each example.
* **global\_batch\_size**: Optional global batch size value. Defaults to (size of first dimension of losses) \* (number of replicas).

#### Returns:

Scalar loss value.

# tf.nn.conv1d\_transpose

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv1d_transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv1d_transpose#aliases)

The transpose of conv1d.

### Aliases:

* tf.compat.v1.nn.conv1d\_transpose
* tf.compat.v2.nn.conv1d\_transpose
* tf.nn.conv1d\_transpose

tf.nn.conv1d\_transpose(  
    input,  
    filters,  
    output\_shape,  
    strides,  
    padding='SAME',  
    data\_format='NWC',  
    dilations=None,  
    name=None  
)

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

This operation is sometimes called "deconvolution" after [Deconvolutional Networks](https://www.matthewzeiler.com/mattzeiler/deconvolutionalnetworks.pdf), but is really the transpose (gradient) of conv1d rather than an actual deconvolution.

#### Args:

* **input**: A 3-D Tensor of type float and shape [batch, in\_width, in\_channels] for NWCdata format or [batch, in\_channels, in\_width] for NCW data format.
* **filters**: A 3-D Tensor with the same type as value and shape [filter\_width, output\_channels, in\_channels]. filter's in\_channels dimension must match that of value.
* **output\_shape**: A 1-D Tensor, containing three elements, representing the output shape of the deconvolution op.
* **strides**: An int or list of ints that has length 1 or 3. The number of entries by which the filter is moved right at each step.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. 'NWC' and 'NCW' are supported.
* **dilations**: An int or list of ints that has length 1 or 3 which defaults to 1. The dilation factor for each dimension of input. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. Dilations in the batch and depth dimensions must be 1.
* **name**: Optional name for the returned tensor.

#### Returns:

A Tensor with the same type as value.

#### Raises:

* **ValueError**: If input/output depth does not match filter's shape, if output\_shape is not at 3-element vector, if padding is other than 'VALID' or 'SAME', or if data\_format is invalid.

# tf.nn.conv2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d#aliases)

Computes a 2-D convolution given 4-D input and filters tensors.

### Aliases:

* tf.compat.v2.nn.conv2d
* tf.nn.conv2d

tf.nn.conv2d(  
    input,  
    filters,  
    strides,  
    padding,  
    data\_format='NHWC',  
    dilations=None,  
    name=None  
)

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

Given an input tensor of shape [batch, in\_height, in\_width, in\_channels] and a filter / kernel tensor of shape [filter\_height, filter\_width, in\_channels, out\_channels], this op performs the following:

1. Flattens the filter to a 2-D matrix with shape [filter\_height \* filter\_width \* in\_channels, output\_channels].
2. Extracts image patches from the input tensor to form a virtual tensor of shape [batch, out\_height, out\_width, filter\_height \* filter\_width \* in\_channels].
3. For each patch, right-multiplies the filter matrix and the image patch vector.

In detail, with the default NHWC format,

output[b, i, j, k] =  
    sum\_{di, dj, q} input[b, strides[1] \* i + di, strides[2] \* j + dj, q] \*  
                    filter[di, dj, q, k]

Must have strides[0] = strides[3] = 1. For the most common case of the same horizontal and vertices strides, strides = [1, stride, stride, 1].

#### Args:

* **input**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. A 4-D tensor. The dimension order is interpreted according to the value of data\_format, see below for details.
* **filters**: A Tensor. Must have the same type as input. A 4-D tensor of shape[filter\_height, filter\_width, in\_channels, out\_channels]
* **strides**: An int or list of ints that has length 1, 2 or 4. The stride of the sliding window for each dimension of input. If a single value is given it is replicated in the H and W dimension. By default the N and C dimensions are set to 1. The dimension order is determined by the value of data\_format, see below for details.
* **padding**: Either the string "SAME" or "VALID" indicating the type of padding algorithm to use, or a list indicating the explicit paddings at the start and end of each dimension. When explicit padding is used and data\_format is "NHWC", this should be in the form [[0, 0], [pad\_top, pad\_bottom], [pad\_left, pad\_right], [0, 0]]. When explicit padding used and data\_format is "NCHW", this should be in the form [[0, 0], [0, 0], [pad\_top, pad\_bottom], [pad\_left, pad\_right]].
* **data\_format**: An optional string from: "NHWC", "NCHW". Defaults to "NHWC". Specify the data format of the input and output data. With the default format "NHWC", the data is stored in the order of: [batch, height, width, channels]. Alternatively, the format could be "NCHW", the data storage order of: [batch, channels, height, width].
* **dilations**: An int or list of ints that has length 1, 2 or 4, defaults to 1. The dilation factor for each dimension ofinput. If a single value is given it is replicated in the H and W dimension. By default the N and C dimensions are set to 1. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions if a 4-d tensor must be 1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.conv2d\_transpose

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d_transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv2d_transpose#aliases)

The transpose of conv2d.

### Aliases:

* tf.compat.v2.nn.conv2d\_transpose
* tf.nn.conv2d\_transpose

tf.nn.conv2d\_transpose(  
    input,  
    filters,  
    output\_shape,  
    strides,  
    padding='SAME',  
    data\_format='NHWC',  
    dilations=None,  
    name=None  
)

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

This operation is sometimes called "deconvolution" after [Deconvolutional Networks](http://www.matthewzeiler.com/pubs/cvpr2010/cvpr2010.pdf), but is actually the transpose (gradient) of conv2d rather than an actual deconvolution.

#### Args:

* **input**: A 4-D Tensor of type float and shape [batch, height, width, in\_channels] for NHWC data format or [batch, in\_channels, height, width] for NCHW data format.
* **filters**: A 4-D Tensor with the same type as input and shape [height, width, output\_channels, in\_channels]. filter's in\_channels dimension must match that of input.
* **output\_shape**: A 1-D Tensor representing the output shape of the deconvolution op.
* **strides**: An int or list of ints that has length 1, 2 or 4. The stride of the sliding window for each dimension of input. If a single value is given it is replicated in the H and W dimension. By default the N and C dimensions are set to 0. The dimension order is determined by the value of data\_format, see below for details.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. 'NHWC' and 'NCHW' are supported.
* **dilations**: An int or list of ints that has length 1, 2 or 4, defaults to 1. The dilation factor for each dimension ofinput. If a single value is given it is replicated in the H and W dimension. By default the N and C dimensions are set to 1. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions if a 4-d tensor must be 1.
* **name**: Optional name for the returned tensor.

#### Returns:

A Tensor with the same type as input.

#### Raises:

* **ValueError**: If input/output depth does not match filter's shape, or if padding is other than 'VALID' or 'SAME'.

# tf.nn.conv3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv3d#aliases)

Computes a 3-D convolution given 5-D input and filters tensors.

### Aliases:

* tf.compat.v2.nn.conv3d
* tf.nn.conv3d

tf.nn.conv3d(  
    input,  
    filters,  
    strides,  
    padding,  
    data\_format='NDHWC',  
    dilations=None,  
    name=None  
)

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

In signal processing, cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. This is also known as a sliding dot product or sliding inner-product.

Our Conv3D implements a form of cross-correlation.

#### Args:

* **input**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. Shape [batch, in\_depth, in\_height, in\_width, in\_channels].
* **filters**: A Tensor. Must have the same type as input. Shape [filter\_depth, filter\_height, filter\_width, in\_channels, out\_channels]. in\_channels must match between input and filters.
* **strides**: A list of ints that has length >= 5. 1-D tensor of length 5. The stride of the sliding window for each dimension of input. Must have strides[0] = strides[4] = 1.
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: An optional string from: "NDHWC", "NCDHW". Defaults to "NDHWC". The data format of the input and output data. With the default format "NDHWC", the data is stored in the order of: [batch, in\_depth, in\_height, in\_width, in\_channels]. Alternatively, the format could be "NCDHW", the data storage order is: [batch, in\_channels, in\_depth, in\_height, in\_width].
* **dilations**: An optional list of ints. Defaults to [1, 1, 1, 1, 1]. 1-D tensor of length 5. The dilation factor for each dimension of input. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions must be 1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.conv3d\_transpose

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv3d_transpose#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/conv3d_transpose#aliases)

The transpose of conv3d.

### Aliases:

* tf.compat.v2.nn.conv3d\_transpose
* tf.nn.conv3d\_transpose

tf.nn.conv3d\_transpose(  
    input,  
    filters,  
    output\_shape,  
    strides,  
    padding='SAME',  
    data\_format='NDHWC',  
    dilations=None,  
    name=None  
)

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

This operation is sometimes called "deconvolution" after [Deconvolutional Networks](http://www.matthewzeiler.com/pubs/cvpr2010/cvpr2010.pdf), but is actually the transpose (gradient) of conv2d rather than an actual deconvolution.

#### Args:

* **input**: A 5-D Tensor of type float and shape [batch, height, width, in\_channels] for NHWC data format or [batch, in\_channels, height, width] for NCHW data format.
* **filters**: A 5-D Tensor with the same type as value and shape [height, width, output\_channels, in\_channels]. filter's in\_channels dimension must match that of value.
* **output\_shape**: A 1-D Tensor representing the output shape of the deconvolution op.
* **strides**: An int or list of ints that has length 1, 3 or 5. The stride of the sliding window for each dimension of input. If a single value is given it is replicated in the D, H and W dimension. By default the N and C dimensions are set to 0. The dimension order is determined by the value of data\_format, see below for details.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. 'NDHWC' and 'NCDHW' are supported.
* **dilations**: An int or list of ints that has length 1, 3 or 5, defaults to 1. The dilation factor for each dimension ofinput. If a single value is given it is replicated in the D, H and Wdimension. By default the N and C dimensions are set to 1. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions if a 5-d tensor must be 1.
* **name**: Optional name for the returned tensor.

#### Returns:

A Tensor with the same type as value.

# tf.nn.convolution

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

Computes sums of N-D convolutions (actually cross-correlation).

### Aliases:

* tf.compat.v2.nn.convolution
* tf.nn.convolution

tf.nn.convolution(  
    input,  
    filters,  
    strides=None,  
    padding='VALID',  
    data\_format=None,  
    dilations=None,  
    name=None  
)

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

This also supports either output striding via the optional strides parameter or atrous convolution (also known as convolution with holes or dilated convolution, based on the French word "trous" meaning holes in English) via the optional dilations parameter. Currently, however, output striding is not supported for atrous convolutions.

Specifically, in the case that data\_format does not start with "NC", given a rank (N+2) input Tensor of shape

[num\_batches, input\_spatial\_shape[0], ..., input\_spatial\_shape[N-1], num\_input\_channels],

a rank (N+2) filters Tensor of shape

[spatial\_filter\_shape[0], ..., spatial\_filter\_shape[N-1], num\_input\_channels, num\_output\_channels],

an optional dilations tensor of shape [N](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/defaulting%20to%20%5B1%5D*N) specifying the filter upsampling/input downsampling rate, and an optional list of N strides (defaulting [1]\*N), this computes for each N-D spatial output position (x[0], ..., x[N-1]):

  output[b, x[0], ..., x[N-1], k] =  
      sum\_{z[0], ..., z[N-1], q}  
          filter[z[0], ..., z[N-1], q, k] \*  
          padded\_input[b,  
                       x[0]\*strides[0] + dilation\_rate[0]\*z[0],  
                       ...,  
                       x[N-1]\*strides[N-1] + dilation\_rate[N-1]\*z[N-1],  
                       q]

where b is the index into the batch, k is the output channel number, q is the input channel number, and z is the N-D spatial offset within the filter. Here, padded\_input is obtained by zero padding the input using an effective spatial filter shape of (spatial\_filter\_shape-1) \* dilation\_rate + 1 and output striding strides as described in the [comment here](https://tensorflow.org/api_guides/python/nn#Convolution).

In the case that data\_format does start with "NC", the input and output (but not the filters) are simply transposed as follows:

convolution(input, data\_format, \*\*kwargs) = tf.transpose(convolution(tf.transpose(input, [0] + range(2,N+2) + [1]), \*\*kwargs), [0, N+1] + range(1, N+1))

It is required that 1 <= N <= 3.

#### Args:

* **input**: An (N+2)-D Tensor of type T, of shape [batch\_size] + input\_spatial\_shape + [in\_channels] if data\_format does not start with "NC" (default), or [batch\_size, in\_channels] + input\_spatial\_shape if data\_format starts with "NC".
* **filters**: An (N+2)-D Tensor with the same type as input and shapespatial\_filter\_shape + [in\_channels, out\_channels].
* **padding**: A string, either "VALID" or "SAME". The padding algorithm.
* **strides**: Optional. Sequence of N ints >= 1. Specifies the output stride. Defaults to [1]\*N. If any value of strides is > 1, then all values of dilation\_rate must be 1.
* **dilations**: Optional. Sequence of N ints >= 1. Specifies the filter upsampling/input downsampling rate. In the literature, the same parameter is sometimes called input stride or dilation. The effective filter size used for the convolution will be spatial\_filter\_shape + (spatial\_filter\_shape - 1) \* (rate - 1), obtained by inserting (dilation\_rate[i]-1) zeros between consecutive elements of the original filter in each spatial dimension i. If any value of dilation\_rate is > 1, then all values of strides must be 1.
* **name**: Optional name for the returned tensor.
* **data\_format**: A string or None. Specifies whether the channel dimension of the input and output is the last dimension (default, or if data\_format does not start with "NC"), or the second dimension (if data\_format starts with "NC"). For N=1, the valid values are "NWC" (default) and "NCW". For N=2, the valid values are "NHWC" (default) and "NCHW". For N=3, the valid values are "NDHWC" (default) and "NCDHW".
* **filters**: Alias of filter.
* **dilations**: Alias of dilation\_rate.

#### Returns:

A Tensor with the same type as input of shape

`[batch\_size] + output\_spatial\_shape + [out\_channels]`

if data\_format is None or does not start with "NC", or

`[batch\_size, out\_channels] + output\_spatial\_shape`

if data\_format starts with "NC", where output\_spatial\_shape depends on the value of padding.

If padding == "SAME": output\_spatial\_shape[i] = ceil(input\_spatial\_shape[i] / strides[i])

If padding == "VALID": output\_spatial\_shape[i] = ceil((input\_spatial\_shape[i] - (spatial\_filter\_shape[i]-1) \* dilation\_rate[i]) / strides[i]).

#### Raises:

* **ValueError**: If input/output depth does not match filters shape, if padding is other than "VALID" or "SAME", or if data\_format is invalid.

# tf.nn.conv\_transpose

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

The transpose of convolution.

### Aliases:

* tf.compat.v1.nn.conv\_transpose
* tf.compat.v2.nn.conv\_transpose
* tf.nn.conv\_transpose

tf.nn.conv\_transpose(  
    input,  
    filters,  
    output\_shape,  
    strides,  
    padding='SAME',  
    data\_format=None,  
    dilations=None,  
    name=None  
)

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

This operation is sometimes called "deconvolution" after [Deconvolutional Networks](http://www.matthewzeiler.com/pubs/cvpr2010/cvpr2010.pdf), but is actually the transpose (gradient) of convolution rather than an actual deconvolution.

#### Args:

* **input**: An N+2 dimensional Tensor of shape [batch\_size] + input\_spatial\_shape + [in\_channels] if data\_format does not start with "NC" (default), or [batch\_size, in\_channels] + input\_spatial\_shape if data\_format starts with "NC". It must be one of the following types: half, bfloat16, float32, float64.
* **filters**: An N+2 dimensional Tensor with the same type as input and shape spatial\_filter\_shape + [in\_channels, out\_channels].
* **output\_shape**: A 1-D Tensor representing the output shape of the deconvolution op.
* **strides**: An int or list of ints that has length 1, N or N+2. The stride of the sliding window for each dimension of input. If a single value is given it is replicated in the spatial dimensions. By default the N and C dimensions are set to 0. The dimension order is determined by the value of data\_format, see below for details.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string or None. Specifies whether the channel dimension of the input and output is the last dimension (default, or if data\_format does not start with "NC"), or the second dimension (if data\_format starts with "NC"). For N=1, the valid values are "NWC" (default) and "NCW". For N=2, the valid values are "NHWC" (default) and "NCHW". For N=3, the valid values are "NDHWC" (default) and "NCDHW".
* **dilations**: An int or list of ints that has length 1, N or N+2, defaults to 1. The dilation factor for each dimension ofinput. If a single value is given it is replicated in the spatial dimensions. By default the N and C dimensions are set to 1. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details.
* **name**: A name for the operation (optional). If not specified "conv\_transpose" is used.

#### Returns:

A Tensor with the same type as value.

# tf.nn.crelu

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

Computes Concatenated ReLU.

### Aliases:

* tf.compat.v2.nn.crelu
* tf.nn.crelu

tf.nn.crelu(  
    features,  
    axis=-1,  
    name=None  
)

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

Concatenates a ReLU which selects only the positive part of the activation with a ReLU which selects only the negative part of the activation. Note that as a result this non-linearity doubles the depth of the activations. Source: [Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units. W. Shang, et al.](https://arxiv.org/abs/1603.05201)

#### Args:

* **features**: A Tensor with type float, double, int32, int64, uint8, int16, or int8.
* **name**: A name for the operation (optional).
* **axis**: The axis that the output values are concatenated along. Default is -1.

#### Returns:

A Tensor with the same type as features.

# tf.nn.ctc\_beam\_search\_decoder

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

Performs beam search decoding on the logits given in input.

### Aliases:

* tf.compat.v1.nn.ctc\_beam\_search\_decoder\_v2
* tf.compat.v2.nn.ctc\_beam\_search\_decoder
* tf.nn.ctc\_beam\_search\_decoder

tf.nn.ctc\_beam\_search\_decoder(  
    inputs,  
    sequence\_length,  
    beam\_width=100,  
    top\_paths=1  
)

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

**Note** The ctc\_greedy\_decoder is a special case of the ctc\_beam\_search\_decoder with top\_paths=1 and beam\_width=1 (but that decoder is faster for this special case).

#### Args:

* **inputs**: 3-D float Tensor, size [max\_time, batch\_size, num\_classes]. The logits.
* **sequence\_length**: 1-D int32 vector containing sequence lengths, having size [batch\_size].
* **beam\_width**: An int scalar >= 0 (beam search beam width).
* **top\_paths**: An int scalar >= 0, <= beam\_width (controls output size).

#### Returns:

A tuple (decoded, log\_probabilities) where

* **decoded**: A list of length top\_paths, where decoded[j] is a SparseTensor containing the decoded outputs:

decoded[j].indices: Indices matrix [total\_decoded\_outputs[j], 2]; The rows store: [batch, time].

decoded[j].values: Values vector, size [total\_decoded\_outputs[j]]. The vector stores the decoded classes for beam j.

decoded[j].dense\_shape: Shape vector, size (2). The shape values are: [batch\_size, max\_decoded\_length[j]].

* **log\_probability**: A float matrix [batch\_size, top\_paths] containing sequence log-probabilities.

# tf.nn.ctc\_greedy\_decoder

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

Performs greedy decoding on the logits given in input (best path).

### Aliases:

* tf.compat.v1.nn.ctc\_greedy\_decoder
* tf.compat.v2.nn.ctc\_greedy\_decoder
* tf.nn.ctc\_greedy\_decoder

tf.nn.ctc\_greedy\_decoder(  
    inputs,  
    sequence\_length,  
    merge\_repeated=True  
)

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

**Note:** Regardless of the value of merge\_repeated, if the maximum index of a given time and batch corresponds to the blank index **(num\_classes - 1)**, no new element is emitted.

If merge\_repeated is True, merge repeated classes in output. This means that if consecutive logits' maximum indices are the same, only the first of these is emitted. The sequence A B B \* B \* B(where '\*' is the blank label) becomes

* A B B B if merge\_repeated=True.
* A B B B B if merge\_repeated=False.

#### Args:

* **inputs**: 3-D float Tensor sized [max\_time, batch\_size, num\_classes]. The logits.
* **sequence\_length**: 1-D int32 vector containing sequence lengths, having size [batch\_size].
* **merge\_repeated**: Boolean. Default: True.

#### Returns:

A tuple (decoded, neg\_sum\_logits) where

* **decoded**: A single-element list. decoded[0] is an SparseTensor containing the decoded outputs s.t.:

decoded.indices: Indices matrix (total\_decoded\_outputs, 2). The rows store: [batch, time].

decoded.values: Values vector, size (total\_decoded\_outputs). The vector stores the decoded classes.

decoded.dense\_shape: Shape vector, size (2). The shape values are: [batch\_size, max\_decoded\_length]

* **neg\_sum\_logits**: A float matrix (batch\_size x 1) containing, for the sequence found, the negative of the sum of the greatest logit at each timeframe.

# tf.nn.ctc\_loss

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

Computes CTC (Connectionist Temporal Classification) loss.

### Aliases:

* tf.compat.v1.nn.ctc\_loss\_v2
* tf.compat.v2.nn.ctc\_loss
* tf.nn.ctc\_loss

tf.nn.ctc\_loss(  
    labels,  
    logits,  
    label\_length,  
    logit\_length,  
    logits\_time\_major=True,  
    unique=None,  
    blank\_index=None,  
    name=None  
)

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

This op implements the CTC loss as presented in the article:

[A. Graves, S. Fernandez, F. Gomez, J. Schmidhuber. Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks. ICML 2006, Pittsburgh, USA, pp. 369-376.](http://www.cs.toronto.edu/~graves/icml_2006.pdf)

#### Notes:

* Same as the "Classic CTC" in TensorFlow 1.x's tf.compat.v1.nn.ctc\_loss setting of preprocess\_collapse\_repeated=False, ctc\_merge\_repeated=True
* Labels may be supplied as either a dense, zero-padded tensor with a vector of label sequence lengths OR as a SparseTensor.
* On TPU and GPU:
  + Only dense padded labels are supported.
* On CPU:
  + Caller may use SparseTensor or dense padded labels but calling with a SparseTensor will be significantly faster.
* Default blank label is 0 rather num\_classes - 1, unless overridden by blank\_index.

#### Args:

* **labels**: tensor of shape [batch\_size, max\_label\_seq\_length] or SparseTensor
* **logits**: tensor of shape [frames, batch\_size, num\_labels], if logits\_time\_major == False, shape is [batch\_size, frames, num\_labels].
* **label\_length**: tensor of shape [batch\_size], None if labels is SparseTensor Length of reference label sequence in labels.
* **logit\_length**: tensor of shape [batch\_size] Length of input sequence in logits.
* **logits\_time\_major**: (optional) If True (default), logits is shaped [time, batch, logits]. If False, shape is [batch, time, logits]
* **unique**: (optional) Unique label indices as computed by ctc\_unique\_labels(labels). If supplied, enable a faster, memory efficient implementation on TPU.
* **blank\_index**: (optional) Set the class index to use for the blank label. Negative values will start from num\_classes, ie, -1 will reproduce the ctc\_loss behavior of using num\_classes - 1 for the blank symbol. There is some memory/performance overhead to switching from the default of 0 as an additional shifted copy of the logits may be created.
* **name**: A name for this Op. Defaults to "ctc\_loss\_dense".

#### Returns:

* **loss**: tensor of shape [batch\_size], negative log probabilities.

# tf.nn.ctc\_unique\_labels

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

Get unique labels and indices for batched labels for [tf.nn.ctc\_loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_loss).

### Aliases:

* tf.compat.v1.nn.ctc\_unique\_labels
* tf.compat.v2.nn.ctc\_unique\_labels
* tf.nn.ctc\_unique\_labels

tf.nn.ctc\_unique\_labels(  
    labels,  
    name=None  
)

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

For use with [tf.nn.ctc\_loss](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/ctc_loss) optional argument unique: This op can be used to preprocess labels in input pipeline to for better speed/memory use computing the ctc loss on TPU.

#### Example:

ctc\_unique\_labels([[3, 4, 4, 3]]) -> unique labels padded with 0: [[3, 4, 0, 0]] indices of original labels in unique: [0, 1, 1, 0]

#### Args:

* **labels**: tensor of shape [batch\_size, max\_label\_length] padded with 0.
* **name**: A name for this Op. Defaults to "ctc\_unique\_labels".

#### Returns:

tuple of - unique labels, tensor of shape [batch\_size, max\_label\_length] - indices into unique labels, shape [batch\_size, max\_label\_length]

# tf.nn.depthwise\_conv2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d#aliases)

Depthwise 2-D convolution.

### Aliases:

* tf.compat.v2.nn.depthwise\_conv2d
* tf.nn.depthwise\_conv2d

tf.nn.depthwise\_conv2d(  
    input,  
    filter,  
    strides,  
    padding,  
    data\_format=None,  
    dilations=None,  
    name=None  
)

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

Given a 4D input tensor ('NHWC' or 'NCHW' data formats) and a filter tensor of shape[filter\_height, filter\_width, in\_channels, channel\_multiplier] containing in\_channels convolutional filters of depth 1, depthwise\_conv2d applies a different filter to each input channel (expanding from 1 channel to channel\_multiplier channels for each), then concatenates the results together. The output has in\_channels \* channel\_multiplier channels.

In detail, with the default NHWC format,

output[b, i, j, k \* channel\_multiplier + q] = sum\_{di, dj}  
     filter[di, dj, k, q] \* input[b, strides[1] \* i + rate[0] \* di,  
                                     strides[2] \* j + rate[1] \* dj, k]

Must have strides[0] = strides[3] = 1. For the most common case of the same horizontal and vertical strides, strides = [1, stride, stride, 1]. If any value in rate is greater than 1, we perform atrous depthwise convolution, in which case all values in the strides tensor must be equal to 1.

#### Args:

* **input**: 4-D with shape according to data\_format.
* **filter**: 4-D with shape [filter\_height, filter\_width, in\_channels, channel\_multiplier].
* **strides**: 1-D of size 4. The stride of the sliding window for each dimension of input.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: The data format for input. Either "NHWC" (default) or "NCHW".
* **dilations**: 1-D of size 2. The dilation rate in which we sample input values across the heightand width dimensions in atrous convolution. If it is greater than 1, then all values of strides must be 1.
* **name**: A name for this operation (optional).

#### Returns:

A 4-D Tensor with shape according to data\_format. E.g., for "NHWC" format, shape is [batch, out\_height, out\_width, in\_channels \* channel\_multiplier].

# tf.nn.depthwise\_conv2d\_backprop\_filter

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_filter#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_filter#aliases)

Computes the gradients of depthwise convolution with respect to the filter.

### Aliases:

* tf.compat.v1.nn.depthwise\_conv2d\_backprop\_filter
* tf.compat.v1.nn.depthwise\_conv2d\_native\_backprop\_filter
* tf.compat.v2.nn.depthwise\_conv2d\_backprop\_filter
* tf.nn.depthwise\_conv2d\_backprop\_filter

tf.nn.depthwise\_conv2d\_backprop\_filter(  
    input,  
    filter\_sizes,  
    out\_backprop,  
    strides,  
    padding,  
    data\_format='NHWC',  
    dilations=[1, 1, 1, 1],  
    name=None  
)

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

#### Args:

* **input**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 4-D with shape based on data\_format. For example, if data\_format is 'NHWC' then input is a 4-D [batch, in\_height, in\_width, in\_channels] tensor.
* **filter\_sizes**: A Tensor of type int32. An integer vector representing the tensor shape of filter, where filter is a 4-D [filter\_height, filter\_width, in\_channels, depthwise\_multiplier] tensor.
* **out\_backprop**: A Tensor. Must have the same type as input. 4-D with shape based on data\_format. For example, if data\_format is 'NHWC' then out\_backprop shape is [batch, out\_height, out\_width, out\_channels]. Gradients w.r.t. the output of the convolution.
* **strides**: A list of ints. The stride of the sliding window for each dimension of the input of the convolution.
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: An optional string from: "NHWC", "NCHW". Defaults to "NHWC". Specify the data format of the input and output data. With the default format "NHWC", the data is stored in the order of: [batch, height, width, channels]. Alternatively, the format could be "NCHW", the data storage order of: [batch, channels, height, width].
* **dilations**: An optional list of ints. Defaults to [1, 1, 1, 1]. 1-D tensor of length 4. The dilation factor for each dimension of input. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions must be 1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.depthwise\_conv2d\_backprop\_input

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_input#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/depthwise_conv2d_backprop_input#aliases)

Computes the gradients of depthwise convolution with respect to the input.

### Aliases:

* tf.compat.v1.nn.depthwise\_conv2d\_backprop\_input
* tf.compat.v1.nn.depthwise\_conv2d\_native\_backprop\_input
* tf.compat.v2.nn.depthwise\_conv2d\_backprop\_input
* tf.nn.depthwise\_conv2d\_backprop\_input

tf.nn.depthwise\_conv2d\_backprop\_input(  
    input\_sizes,  
    filter,  
    out\_backprop,  
    strides,  
    padding,  
    data\_format='NHWC',  
    dilations=[1, 1, 1, 1],  
    name=None  
)

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

#### Args:

* **input\_sizes**: A Tensor of type int32. An integer vector representing the shape of input, based on data\_format. For example, if data\_format is 'NHWC' then input is a 4-D [batch, height, width, channels] tensor.
* **filter**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 4-D with shape [filter\_height, filter\_width, in\_channels, depthwise\_multiplier].
* **out\_backprop**: A Tensor. Must have the same type as filter. 4-D with shape based on data\_format. For example, if data\_format is 'NHWC' then out\_backprop shape is [batch, out\_height, out\_width, out\_channels]. Gradients w.r.t. the output of the convolution.
* **strides**: A list of ints. The stride of the sliding window for each dimension of the input of the convolution.
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: An optional string from: "NHWC", "NCHW". Defaults to "NHWC". Specify the data format of the input and output data. With the default format "NHWC", the data is stored in the order of: [batch, height, width, channels]. Alternatively, the format could be "NCHW", the data storage order of: [batch, channels, height, width].
* **dilations**: An optional list of ints. Defaults to [1, 1, 1, 1]. 1-D tensor of length 4. The dilation factor for each dimension of input. If set to k > 1, there will be k-1 skipped cells between each filter element on that dimension. The dimension order is determined by the value of data\_format, see above for details. Dilations in the batch and depth dimensions must be 1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as filter.

# tf.nn.depth\_to\_space

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

DepthToSpace for tensors of type T.

### Aliases:

* tf.compat.v2.nn.depth\_to\_space
* tf.nn.depth\_to\_space

tf.nn.depth\_to\_space(  
    input,  
    block\_size,  
    data\_format='NHWC',  
    name=None  
)

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

Rearranges data from depth into blocks of spatial data. This is the reverse transformation of SpaceToDepth. More specifically, this op outputs a copy of the input tensor where values from the depth dimension are moved in spatial blocks to the height and width dimensions. The attr block\_size indicates the input block size and how the data is moved.

* Chunks of data of size block\_size \* block\_size from depth are rearranged into non-overlapping blocks of size block\_size x block\_size
* The width the output tensor is input\_depth \* block\_size, whereas the height is input\_height \* block\_size.
* The Y, X coordinates within each block of the output image are determined by the high order component of the input channel index.
* The depth of the input tensor must be divisible by block\_size \* block\_size.

The data\_format attr specifies the layout of the input and output tensors with the following options: "NHWC": [ batch, height, width, channels ] "NCHW": [ batch, channels, height, width ] "NCHW\_VECT\_C": qint8 [ batch, channels / 4, height, width, 4 ]

It is useful to consider the operation as transforming a 6-D Tensor. e.g. for data\_format = NHWC, Each element in the input tensor can be specified via 6 coordinates, ordered by decreasing memory layout significance as: n,iY,iX,bY,bX,oC (where n=batch index, iX, iY means X or Y coordinates within the input image, bX, bY means coordinates within the output block, oC means output channels). The output would be the input transposed to the following layout: n,iY,bY,iX,bX,oC

This operation is useful for resizing the activations between convolutions (but keeping all data), e.g. instead of pooling. It is also useful for training purely convolutional models.

For example, given an input of shape [1, 1, 1, 4], data\_format = "NHWC" and block\_size = 2:

x = [[[[1, 2, 3, 4]]]]

This operation will output a tensor of shape [1, 2, 2, 1]:

   [[[[1], [2]],  
     [[3], [4]]]]

Here, the input has a batch of 1 and each batch element has shape [1, 1, 4], the corresponding output will have 2x2 elements and will have a depth of 1 channel (1 = 4 / (block\_size \* block\_size)). The output element shape is [2, 2, 1].

For an input tensor with larger depth, here of shape [1, 1, 1, 12], e.g.

x = [[[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]]]]

This operation, for block size of 2, will return the following tensor of shape [1, 2, 2, 3]

   [[[[1, 2, 3], [4, 5, 6]],  
     [[7, 8, 9], [10, 11, 12]]]]

Similarly, for the following input of shape [1 2 2 4], and a block size of 2:

x =  [[[[1, 2, 3, 4],  
       [5, 6, 7, 8]],  
      [[9, 10, 11, 12],  
       [13, 14, 15, 16]]]]

the operator will return the following tensor of shape [1 4 4 1]:

x = [[[ [1],   [2],  [5],  [6]],  
      [ [3],   [4],  [7],  [8]],  
      [ [9],  [10], [13],  [14]],  
      [ [11], [12], [15],  [16]]]]

#### Args:

* **input**: A Tensor.
* **block\_size**: An int that is >= 2. The size of the spatial block, same as in Space2Depth.
* **data\_format**: An optional string from: "NHWC", "NCHW", "NCHW\_VECT\_C". Defaults to "NHWC".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.dilation2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/dilation2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/dilation2d#aliases)

Computes the grayscale dilation of 4-D input and 3-D filters tensors.

### Aliases:

* tf.compat.v2.nn.dilation2d
* tf.nn.dilation2d

tf.nn.dilation2d(  
    input,  
    filters,  
    strides,  
    padding,  
    data\_format,  
    dilations,  
    name=None  
)

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

The input tensor has shape [batch, in\_height, in\_width, depth] and the filters tensor has shape [filter\_height, filter\_width, depth], i.e., each input channel is processed independently of the others with its own structuring function. The output tensor has shape [batch, out\_height, out\_width, depth]. The spatial dimensions of the output tensor depend on the padding algorithm. We currently only support the default "NHWC" data\_format.

In detail, the grayscale morphological 2-D dilation is the max-sum correlation (for consistency with conv2d, we use unmirrored filters):

output[b, y, x, c] =  
   max\_{dy, dx} input[b,  
                      strides[1] \* y + rates[1] \* dy,  
                      strides[2] \* x + rates[2] \* dx,  
                      c] +  
                filters[dy, dx, c]

Max-pooling is a special case when the filter has size equal to the pooling kernel size and contains all zeros.

Note on duality: The dilation of input by the filters is equal to the negation of the erosion of -input by the reflected filters.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64. 4-D with shape [batch, in\_height, in\_width, depth].
* **filters**: A Tensor. Must have the same type as input. 3-D with shape [filter\_height, filter\_width, depth].
* **strides**: A list of ints that has length >= 4. The stride of the sliding window for each dimension of the input tensor. Must be: [1, stride\_height, stride\_width, 1].
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: A string, only "NCHW" is currently supported.
* **dilations**: A list of ints that has length >= 4. The input stride for atrous morphological dilation. Must be: [1, rate\_height, rate\_width, 1].
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.dropout

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

Computes dropout.

### Aliases:

* tf.compat.v2.nn.dropout
* tf.nn.dropout

tf.nn.dropout(  
    x,  
    rate,  
    noise\_shape=None,  
    seed=None,  
    name=None  
)

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

### Used in the guide:

* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

With probability rate, drops elements of x. Input that are kept are scaled up by 1 / (1 - rate), otherwise outputs 0. The scaling is so that the expected sum is unchanged.

**Note:** The behavior of dropout has changed between TensorFlow 1.x and 2.x. When converting 1.x code, please use named arguments to ensure behavior stays consistent.

By default, each element is kept or dropped independently. If noise\_shape is specified, it must be[broadcastable](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html) to the shape of x, and only dimensions with noise\_shape[i] == shape(x)[i] will make independent decisions. For example, if shape(x) = [k, l, m, n] and noise\_shape = [k, 1, 1, n], each batch and channel component will be kept independently and each row and column will be kept or not kept together.

#### Args:

* **x**: A floating point tensor.
* **rate**: A scalar Tensor with the same type as x. The probability that each element is dropped. For example, setting rate=0.1 would drop 10% of input elements.
* **noise\_shape**: A 1-D Tensor of type int32, representing the shape for randomly generated keep/drop flags.
* **seed**: A Python integer. Used to create random seeds. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.
* **name**: A name for this operation (optional).

#### Returns:

A Tensor of the same shape of x.

#### Raises:

* **ValueError**: If rate is not in (0, 1] or if x is not a floating point tensor.

# tf.nn.elu

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

Computes exponential linear: exp(features) - 1 if < 0, features otherwise.

### Aliases:

* tf.compat.v1.nn.elu
* tf.compat.v2.nn.elu
* tf.nn.elu

tf.nn.elu(  
    features,  
    name=None  
)

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

See [Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)](http://arxiv.org/abs/1511.07289)

#### Args:

* **features**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.nn.embedding\_lookup

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

Looks up ids in a list of embedding tensors.

### Aliases:

* tf.compat.v2.nn.embedding\_lookup
* tf.nn.embedding\_lookup

tf.nn.embedding\_lookup(  
    params,  
    ids,  
    max\_norm=None,  
    name=None  
)

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

This function is used to perform parallel lookups on the list of tensors in params. It is a generalization of [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather), where params is interpreted as a partitioning of a large embedding tensor. paramsmay be a PartitionedVariable as returned by using tf.compat.v1.get\_variable() with a partitioner.

If len(params) > 1, each element id of ids is partitioned between the elements of paramsaccording to the partition\_strategy. In all strategies, if the id space does not evenly divide the number of partitions, each of the first (max\_id + 1) % len(params) partitions will be assigned one more id.

The partition\_strategy is always "div" currently. This means that we assign ids to partitions in a contiguous manner. For instance, 13 ids are split across 5 partitions as: [[0, 1, 2], [3, 4, 5], [6, 7, 8], [9, 10], [11, 12]]

The results of the lookup are concatenated into a dense tensor. The returned tensor has shape shape(ids) + shape(params)[1:].

#### Args:

* **params**: A single tensor representing the complete embedding tensor, or a list of P tensors all of same shape except for the first dimension, representing sharded embedding tensors. Alternatively, a PartitionedVariable, created by partitioning along dimension 0. Each element must be appropriately sized for the 'div' partition\_strategy.
* **ids**: A Tensor with type int32 or int64 containing the ids to be looked up in params.
* **max\_norm**: If not None, each embedding is clipped if its l2-norm is larger than this value.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as the tensors in params.

#### Raises:

* **ValueError**: If params is empty.

# tf.nn.embedding\_lookup\_sparse

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

Computes embeddings for the given ids and weights.

### Aliases:

* tf.compat.v2.nn.embedding\_lookup\_sparse
* tf.nn.embedding\_lookup\_sparse

tf.nn.embedding\_lookup\_sparse(  
    params,  
    sp\_ids,  
    sp\_weights,  
    combiner=None,  
    max\_norm=None,  
    name=None  
)

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

This op assumes that there is at least one id for each row in the dense tensor represented by sp\_ids (i.e. there are no rows with empty features), and that all the indices of sp\_ids are in canonical row-major order.

It also assumes that all id values lie in the range [0, p0), where p0 is the sum of the size of params along dimension 0.

#### Args:

* **params**: A single tensor representing the complete embedding tensor, or a list of P tensors all of same shape except for the first dimension, representing sharded embedding tensors. Alternatively, a PartitionedVariable, created by partitioning along dimension 0. Each element must be appropriately sized for "div" partition\_strategy.
* **sp\_ids**: N x M SparseTensor of int64 ids where N is typically batch size and M is arbitrary.
* **sp\_weights**: either a SparseTensor of float / double weights, or None to indicate all weights should be taken to be 1. If specified, sp\_weights must have exactly the same shape and indices as sp\_ids.
* **combiner**: A string specifying the reduction op. Currently "mean", "sqrtn" and "sum" are supported. "sum" computes the weighted sum of the embedding results for each row. "mean" is the weighted sum divided by the total weight. "sqrtn" is the weighted sum divided by the square root of the sum of the squares of the weights.
* **max\_norm**: If not None, each embedding is clipped if its l2-norm is larger than this value, before combining.
* **name**: Optional name for the op.

#### Returns:

A dense tensor representing the combined embeddings for the sparse ids. For each row in the dense tensor represented by sp\_ids, the op looks up the embeddings for all ids in that row, multiplies them by the corresponding weight, and combines these embeddings as specified.

In other words, if

shape(combined params) = [p0, p1, ..., pm]

and

shape(sp\_ids) = shape(sp\_weights) = [d0, d1, ..., dn]

then

shape(output) = [d0, d1, ..., dn-1, p1, ..., pm].

For instance, if params is a 10x20 matrix, and sp\_ids / sp\_weights are

[0, 0]: id 1, weight 2.0  
[0, 1]: id 3, weight 0.5  
[1, 0]: id 0, weight 1.0  
[2, 3]: id 1, weight 3.0

with combiner="mean", then the output will be a 3x20 matrix where

output[0, :] = (params[1, :] \* 2.0 + params[3, :] \* 0.5) / (2.0 + 0.5)  
output[1, :] = (params[0, :] \* 1.0) / 1.0  
output[2, :] = (params[1, :] \* 3.0) / 3.0

#### Raises:

* **TypeError**: If sp\_ids is not a SparseTensor, or if sp\_weights is neither None nor SparseTensor.
* **ValueError**: If combiner is not one of {"mean", "sqrtn", "sum"}.

# tf.nn.erosion2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/erosion2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/erosion2d#aliases)

Computes the grayscale erosion of 4-D value and 3-D filters tensors.

### Aliases:

* tf.compat.v2.nn.erosion2d
* tf.nn.erosion2d

tf.nn.erosion2d(  
    value,  
    filters,  
    strides,  
    padding,  
    data\_format,  
    dilations,  
    name=None  
)

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

The value tensor has shape [batch, in\_height, in\_width, depth] and the filters tensor has shape [filters\_height, filters\_width, depth], i.e., each input channel is processed independently of the others with its own structuring function. The output tensor has shape [batch, out\_height, out\_width, depth]. The spatial dimensions of the output tensor depend on the padding algorithm. We currently only support the default "NHWC" data\_format.

In detail, the grayscale morphological 2-D erosion is given by:

output[b, y, x, c] =  
   min\_{dy, dx} value[b,  
                      strides[1] \* y - dilations[1] \* dy,  
                      strides[2] \* x - dilations[2] \* dx,  
                      c] -  
                filters[dy, dx, c]

Duality: The erosion of value by the filters is equal to the negation of the dilation of -value by the reflected filters.

#### Args:

* **value**: A Tensor. 4-D with shape [batch, in\_height, in\_width, depth].
* **filters**: A Tensor. Must have the same type as value. 3-D with shape [filters\_height, filters\_width, depth].
* **strides**: A list of ints that has length >= 4. 1-D of length 4. The stride of the sliding window for each dimension of the input tensor. Must be: [1, stride\_height, stride\_width, 1].
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: A string, only "NHWC" is currently supported.
* **dilations**: A list of ints that has length >= 4. 1-D of length 4. The input stride for atrous morphological dilation. Must be: [1, rate\_height, rate\_width, 1].
* **name**: A name for the operation (optional). If not specified "erosion2d" is used.

#### Returns:

A Tensor. Has the same type as value. 4-D with shape [batch, out\_height, out\_width, depth].

#### Raises:

* **ValueError**: If the value depth does not match filters' shape, or if padding is other than 'VALID' or 'SAME'.

# tf.nn.fractional\_avg\_pool

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

Performs fractional average pooling on the input.

### Aliases:

* tf.compat.v2.nn.fractional\_avg\_pool
* tf.nn.fractional\_avg\_pool

tf.nn.fractional\_avg\_pool(  
    value,  
    pooling\_ratio,  
    pseudo\_random=False,  
    overlapping=False,  
    seed=0,  
    name=None  
)

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

Fractional average pooling is similar to Fractional max pooling in the pooling region generation step. The only difference is that after pooling regions are generated, a mean operation is performed instead of a max operation in each pooling region.

#### Args:

* **value**: A Tensor. 4-D with shape [batch, height, width, channels].
* **pooling\_ratio**: A list of floats that has length >= 4. Pooling ratio for each dimension of value, currently only supports row and col dimension and should be >= 1.0. For example, a valid pooling ratio looks like [1.0, 1.44, 1.73, 1.0]. The first and last elements must be 1.0 because we don't allow pooling on batch and channels dimensions. 1.44 and 1.73 are pooling ratio on height and width dimensions respectively.
* **pseudo\_random**: An optional bool. Defaults to False. When set to True, generates the pooling sequence in a pseudorandom fashion, otherwise, in a random fashion. Check paper [Benjamin Graham, Fractional Max-Pooling](http://arxiv.org/abs/1412.6071) for difference between pseudorandom and random.
* **overlapping**: An optional bool. Defaults to False. When set to True, it means when pooling, the values at the boundary of adjacent pooling cells are used by both cells. For example:index 0 1 2 3 4 value 20 5 16 3 7 If the pooling sequence is [0, 2, 4], then 16, at index 2 will be used twice. The result would be [20, 16] for fractional avg pooling.
* **seed**: An optional int. Defaults to 0. If set to be non-zero, the random number generator is seeded by the given seed. Otherwise it is seeded by a random seed.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (output, row\_pooling\_sequence, col\_pooling\_sequence). output: Output Tensor after fractional avg pooling. Has the same type as value. row\_pooling\_sequence: A Tensor of type int64. col\_pooling\_sequence: A Tensor of type int64.

# tf.nn.fractional\_max\_pool

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

Performs fractional max pooling on the input.

### Aliases:

* tf.compat.v2.nn.fractional\_max\_pool
* tf.nn.fractional\_max\_pool

tf.nn.fractional\_max\_pool(  
    value,  
    pooling\_ratio,  
    pseudo\_random=False,  
    overlapping=False,  
    seed=0,  
    name=None  
)

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

Fractional max pooling is slightly different than regular max pooling. In regular max pooling, you downsize an input set by taking the maximum value of smaller N x N subsections of the set (often 2x2), and try to reduce the set by a factor of N, where N is an integer. Fractional max pooling, as you might expect from the word "fractional", means that the overall reduction ratio N does not have to be an integer.

The sizes of the pooling regions are generated randomly but are fairly uniform. For example, let's look at the height dimension, and the constraints on the list of rows that will be pool boundaries.

First we define the following:

1. input\_row\_length : the number of rows from the input set
2. output\_row\_length : which will be smaller than the input
3. alpha = input\_row\_length / output\_row\_length : our reduction ratio
4. K = floor(alpha)
5. row\_pooling\_sequence : this is the result list of pool boundary rows

Then, row\_pooling\_sequence should satisfy:

1. a[0] = 0 : the first value of the sequence is 0
2. a[end] = input\_row\_length : the last value of the sequence is the size
3. K <= (a[i+1] - a[i]) <= K+1 : all intervals are K or K+1 size
4. length(row\_pooling\_sequence) = output\_row\_length+1

For more details on fractional max pooling, see this paper: [Benjamin Graham, Fractional Max-Pooling](http://arxiv.org/abs/1412.6071)

#### Args:

* **value**: A Tensor. 4-D with shape [batch, height, width, channels].
* **pooling\_ratio**: An int or list of ints that has length 1, 2 or 4. Pooling ratio for each dimension of value, currently only supports row and col dimension and should be >= 1.0. For example, a valid pooling ratio looks like [1.0, 1.44, 1.73, 1.0]. The first and last elements must be 1.0 because we don't allow pooling on batch and channels dimensions. 1.44 and 1.73 are pooling ratio on height and width dimensions respectively.
* **pseudo\_random**: An optional bool. Defaults to False. When set to True, generates the pooling sequence in a pseudorandom fashion, otherwise, in a random fashion. Check paper [Benjamin Graham, Fractional Max-Pooling](http://arxiv.org/abs/1412.6071) for difference between pseudorandom and random.
* **overlapping**: An optional bool. Defaults to False. When set to True, it means when pooling, the values at the boundary of adjacent pooling cells are used by both cells. For example:index 0 1 2 3 4 value 20 5 16 3 7 If the pooling sequence is [0, 2, 4], then 16, at index 2 will be used twice. The result would be [20, 16] for fractional max pooling.
* **seed**: An optional int. Defaults to 0. If set to be non-zero, the random number generator is seeded by the given seed. Otherwise it is seeded by a random seed.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (output, row\_pooling\_sequence, col\_pooling\_sequence). output: Output Tensor after fractional max pooling. Has the same type as value. row\_pooling\_sequence: A Tensor of type int64. col\_pooling\_sequence: A Tensor of type int64.

# tf.nn.l2\_loss

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/l2_loss#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/l2_loss#aliases)

L2 Loss.

### Aliases:

* tf.compat.v1.nn.l2\_loss
* tf.compat.v2.nn.l2\_loss
* tf.nn.l2\_loss

tf.nn.l2\_loss(  
    t,  
    name=None  
)

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

Computes half the L2 norm of a tensor without the sqrt:

output = sum(t \*\* 2) / 2

#### Args:

* **t**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. Typically 2-D, but may have any dimensions.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as t.

# tf.nn.leaky\_relu

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

Compute the Leaky ReLU activation function.

### Aliases:

* tf.compat.v1.nn.leaky\_relu
* tf.compat.v2.nn.leaky\_relu
* tf.nn.leaky\_relu

tf.nn.leaky\_relu(  
    features,  
    alpha=0.2,  
    name=None  
)

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

Source: [Rectifier Nonlinearities Improve Neural Network Acoustic Models. AL Maas, AY Hannun, AY Ng - Proc. ICML, 2013](https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf).

#### Args:

* **features**: A Tensor representing preactivation values. Must be one of the following types: float16, float32, float64, int32, int64.
* **alpha**: Slope of the activation function at x < 0.
* **name**: A name for the operation (optional).

#### Returns:

The activation value.

# tf.nn.local\_response\_normalization

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

Local Response Normalization.

### Aliases:

* tf.compat.v1.nn.local\_response\_normalization
* tf.compat.v1.nn.lrn
* tf.compat.v2.nn.local\_response\_normalization
* tf.compat.v2.nn.lrn
* tf.nn.local\_response\_normalization
* tf.nn.lrn

tf.nn.local\_response\_normalization(  
    input,  
    depth\_radius=5,  
    bias=1,  
    alpha=1,  
    beta=0.5,  
    name=None  
)

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

The 4-D input tensor is treated as a 3-D array of 1-D vectors (along the last dimension), and each vector is normalized independently. Within a given vector, each component is divided by the weighted, squared sum of inputs within depth\_radius. In detail,

sqr\_sum[a, b, c, d] =  
    sum(input[a, b, c, d - depth\_radius : d + depth\_radius + 1] \*\* 2)  
output = input / (bias + alpha \* sqr\_sum) \*\* beta

For details, see [Krizhevsky et al., ImageNet classification with deep convolutional neural networks (NIPS 2012)](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks).

#### Args:

* **input**: A Tensor. Must be one of the following types: half, bfloat16, float32. 4-D.
* **depth\_radius**: An optional int. Defaults to 5. 0-D. Half-width of the 1-D normalization window.
* **bias**: An optional float. Defaults to 1. An offset (usually positive to avoid dividing by 0).
* **alpha**: An optional float. Defaults to 1. A scale factor, usually positive.
* **beta**: An optional float. Defaults to 0.5. An exponent.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.log\_poisson\_loss

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

Computes log Poisson loss given log\_input.

### Aliases:

* tf.compat.v1.nn.log\_poisson\_loss
* tf.compat.v2.nn.log\_poisson\_loss
* tf.nn.log\_poisson\_loss

tf.nn.log\_poisson\_loss(  
    targets,  
    log\_input,  
    compute\_full\_loss=False,  
    name=None  
)

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

Gives the log-likelihood loss between the prediction and the target under the assumption that the target has a Poisson distribution. Caveat: By default, this is not the exact loss, but the loss minus a constant term [log(z!)]. That has no effect for optimization, but does not play well with relative loss comparisons. To compute an approximation of the log factorial term, specify compute\_full\_loss=True to enable Stirling's Approximation.

For brevity, let c = log(x) = log\_input, z = targets. The log Poisson loss is

  -log(exp(-x) \* (x^z) / z!)  
= -log(exp(-x) \* (x^z)) + log(z!)  
~ -log(exp(-x)) - log(x^z) [+ z \* log(z) - z + 0.5 \* log(2 \* pi \* z)]  
    [ Note the second term is the Stirling's Approximation for log(z!).  
      It is invariant to x and does not affect optimization, though  
      important for correct relative loss comparisons. It is only  
      computed when compute\_full\_loss == True. ]  
= x - z \* log(x) [+ z \* log(z) - z + 0.5 \* log(2 \* pi \* z)]  
= exp(c) - z \* c [+ z \* log(z) - z + 0.5 \* log(2 \* pi \* z)]

#### Args:

* **targets**: A Tensor of the same type and shape as log\_input.
* **log\_input**: A Tensor of type float32 or float64.
* **compute\_full\_loss**: whether to compute the full loss. If false, a constant term is dropped in favor of more efficient optimization.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of the same shape as log\_input with the componentwise logistic losses.

#### Raises:

* **ValueError**: If log\_input and targets do not have the same shape.

# tf.nn.log\_softmax

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

Computes log softmax activations.

### Aliases:

* tf.compat.v2.math.log\_softmax
* tf.compat.v2.nn.log\_softmax
* tf.math.log\_softmax
* tf.nn.log\_softmax

tf.nn.log\_softmax(  
    logits,  
    axis=None,  
    name=None  
)

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

For each batch i and class j we have

logsoftmax = logits - log(reduce\_sum(exp(logits), axis))

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as logits. Same shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.nn.max\_pool

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

Performs the max pooling on the input.

### Aliases:

* tf.compat.v1.nn.max\_pool\_v2
* tf.compat.v2.nn.max\_pool
* tf.nn.max\_pool

tf.nn.max\_pool(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format=None,  
    name=None  
)

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

#### Args:

* **input**: Tensor of rank N+2, of shape [batch\_size] + input\_spatial\_shape + [num\_channels] if data\_format does not start with "NC" (default), or [batch\_size, num\_channels] + input\_spatial\_shape if data\_format starts with "NC". Pooling happens over the spatial dimensions only.
* **ksize**: An int or list of ints that has length 1, N or N+2. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, N or N+2. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. Specifies the channel dimension. For N=1 it can be either "NWC" (default) or "NCW", for N=2 it can be either "NHWC" (default) or "NCHW" and for N=3 either "NDHWC" (default) or "NCDHW".
* **name**: Optional name for the operation.

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.max\_pool1d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool1d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool1d#aliases)

Performs the max pooling on the input.

### Aliases:

* tf.compat.v1.nn.max\_pool1d
* tf.compat.v2.nn.max\_pool1d
* tf.nn.max\_pool1d

tf.nn.max\_pool1d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NWC',  
    name=None  
)

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

Note internally this op reshapes and uses the underlying 2d operation.

#### Args:

* **input**: A 3-D Tensor of the format specified by data\_format.
* **ksize**: An int or list of ints that has length 1 or 3. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1 or 3. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: An optional string from: "NWC", "NCW". Defaults to "NWC".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.max\_pool2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool2d#aliases)

Performs the max pooling on the input.

### Aliases:

* tf.compat.v1.nn.max\_pool2d
* tf.compat.v2.nn.max\_pool2d
* tf.nn.max\_pool2d

tf.nn.max\_pool2d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NHWC',  
    name=None  
)

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

#### Args:

* **input**: A 4-D Tensor of the format specified by data\_format.
* **ksize**: An int or list of ints that has length 1, 2 or 4. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, 2 or 4. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string. 'NHWC', 'NCHW' and 'NCHW\_VECT\_C' are supported.
* **name**: Optional name for the operation.

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.max\_pool3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/max_pool3d#aliases)

Performs the max pooling on the input.

### Aliases:

* tf.compat.v1.nn.max\_pool3d
* tf.compat.v2.nn.max\_pool3d
* tf.nn.max\_pool3d

tf.nn.max\_pool3d(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NDHWC',  
    name=None  
)

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

#### Args:

* **input**: A 5-D Tensor of the format specified by data\_format.
* **ksize**: An int or list of ints that has length 1, 3 or 5. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, 3 or 5. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: An optional string from: "NDHWC", "NCDHW". Defaults to "NDHWC". The data format of the input and output data. With the default format "NDHWC", the data is stored in the order of: [batch, in\_depth, in\_height, in\_width, in\_channels]. Alternatively, the format could be "NCDHW", the data storage order is: [batch, in\_channels, in\_depth, in\_height, in\_width].
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of format specified by data\_format. The max pooled output tensor.

# tf.nn.max\_pool\_with\_argmax

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

Performs max pooling on the input and outputs both max values and indices.

### Aliases:

* tf.compat.v2.nn.max\_pool\_with\_argmax
* tf.nn.max\_pool\_with\_argmax

tf.nn.max\_pool\_with\_argmax(  
    input,  
    ksize,  
    strides,  
    padding,  
    data\_format='NHWC',  
    output\_dtype=tf.dtypes.int64,  
    include\_batch\_in\_index=False,  
    name=None  
)

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

The indices in argmax are flattened, so that a maximum value at position [b, y, x, c] becomes flattened index: (y \* width + x) \* channels + c if include\_batch\_in\_index is False; ((b \* height + y) \* width + x) \* channels + c if include\_batch\_in\_index is True.

The indices returned are always in [0, height) x [0, width) before flattening, even if padding is involved and the mathematically correct answer is outside (either negative or too large). This is a bug, but fixing it is difficult to do in a safe backwards compatible way, especially due to flattening.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64. 4-D with shape [batch, height, width, channels]. Input to pool over.
* **ksize**: An int or list of ints that has length 1, 2 or 4. The size of the window for each dimension of the input tensor.
* **strides**: An int or list of ints that has length 1, 2 or 4. The stride of the sliding window for each dimension of the input tensor.
* **padding**: A string from: "SAME", "VALID". The type of padding algorithm to use.
* **data\_format**: An optional string, must be set to "NHWC". Defaults to "NHWC". Specify the data format of the input and output data.
* **output\_dtype**: 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.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64). The dtype of the returned argmax tensor.
* **include\_batch\_in\_index**: An optional boolean. Defaults to False. Whether to include batch dimension in flattened index of argmax.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (output, argmax).

* **output**: A Tensor. Has the same type as input.
* **argmax**: A Tensor of type output\_dtype.

# tf.nn.moments

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

Calculates the mean and variance of x.

### Aliases:

* tf.compat.v2.nn.moments
* tf.nn.moments

tf.nn.moments(  
    x,  
    axes,  
    shift=None,  
    keepdims=False,  
    name=None  
)

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

The mean and variance are calculated by aggregating the contents of x across axes. If x is 1-D and axes = [0] this is just the mean and variance of a vector.

**Note:** shift is currently not used; the true mean is computed and used.

When using these moments for batch normalization (see [tf.nn.batch\_normalization](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/batch_normalization)):

* for so-called "global normalization", used with convolutional filters with shape [batch, height, width, depth], pass axes=[0, 1, 2].
* for simple batch normalization pass axes=[0] (batch only).

#### Args:

* **x**: A Tensor.
* **axes**: Array of ints. Axes along which to compute mean and variance.
* **shift**: Not used in the current implementation.
* **keepdims**: produce moments with the same dimensionality as the input.
* **name**: Name used to scope the operations that compute the moments.

#### Returns:

Two Tensor objects: mean and variance.

# tf.nn.nce\_loss

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

Computes and returns the noise-contrastive estimation training loss.

### Aliases:

* tf.compat.v2.nn.nce\_loss
* tf.nn.nce\_loss

tf.nn.nce\_loss(  
    weights,  
    biases,  
    labels,  
    inputs,  
    num\_sampled,  
    num\_classes,  
    num\_true=1,  
    sampled\_values=None,  
    remove\_accidental\_hits=False,  
    name='nce\_loss'  
)

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

See [Noise-contrastive estimation: A new estimation principle for unnormalized statistical models](http://www.jmlr.org/proceedings/papers/v9/gutmann10a/gutmann10a.pdf). Also see our [Candidate Sampling Algorithms Reference](https://www.tensorflow.org/extras/candidate_sampling.pdf)

A common use case is to use this method for training, and calculate the full sigmoid loss for evaluation or inference as in the following example:

if mode == "train":  
  loss = tf.nn.nce\_loss(  
      weights=weights,  
      biases=biases,  
      labels=labels,  
      inputs=inputs,  
      ...)  
elif mode == "eval":  
  logits = tf.matmul(inputs, tf.transpose(weights))  
  logits = tf.nn.bias\_add(logits, biases)  
  labels\_one\_hot = tf.one\_hot(labels, n\_classes)  
  loss = tf.nn.sigmoid\_cross\_entropy\_with\_logits(  
      labels=labels\_one\_hot,  
      logits=logits)  
  loss = tf.reduce\_sum(loss, axis=1)

**Note:** when doing embedding lookup on **weights** and **bias**, "div" partition strategy will be used. Support for other partition strategy will be added later.**Note:** By default this uses a log-uniform (Zipfian) distribution for sampling, so your labels must be sorted in order of decreasing frequency to achieve good results. For more details, see[**tf.random.log\_uniform\_candidate\_sampler**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/log_uniform_candidate_sampler).**Note:** In the case where **num\_true** > 1, we assign to each target class the target probability 1 / **num\_true** so that the target probabilities sum to 1 per-example.**Note:** It would be useful to allow a variable number of target classes per example. We hope to provide this functionality in a future release. For now, if you have a variable number of target classes, you can pad them out to a constant number by either repeating them or by padding with an otherwise unused class.

#### Args:

* **weights**: A Tensor of shape [num\_classes, dim], or a list of Tensor objects whose concatenation along dimension 0 has shape [num\_classes, dim]. The (possibly-partitioned) class embeddings.
* **biases**: A Tensor of shape [num\_classes]. The class biases.
* **labels**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **inputs**: A Tensor of shape [batch\_size, dim]. The forward activations of the input network.
* **num\_sampled**: An int. The number of negative classes to randomly sample per batch. This single sample of negative classes is evaluated for each element in the batch.
* **num\_classes**: An int. The number of possible classes.
* **num\_true**: An int. The number of target classes per training example.
* **sampled\_values**: a tuple of (sampled\_candidates, true\_expected\_count,sampled\_expected\_count) returned by a \*\_candidate\_sampler function. (if None, we default to log\_uniform\_candidate\_sampler)
* **remove\_accidental\_hits**: A bool. Whether to remove "accidental hits" where a sampled class equals one of the target classes. If set to True, this is a "Sampled Logistic" loss instead of NCE, and we are learning to generate log-odds instead of log probabilities. See our [Candidate Sampling Algorithms Reference](https://www.tensorflow.org/extras/candidate_sampling.pdf). Default is False.
* **name**: A name for the operation (optional).

#### Returns:

A batch\_size 1-D tensor of per-example NCE losses.

# tf.nn.normalize\_moments

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

Calculate the mean and variance of based on the sufficient statistics.

### Aliases:

* tf.compat.v1.nn.normalize\_moments
* tf.compat.v2.nn.normalize\_moments
* tf.nn.normalize\_moments

tf.nn.normalize\_moments(  
    counts,  
    mean\_ss,  
    variance\_ss,  
    shift,  
    name=None  
)

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

#### Args:

* **counts**: A Tensor containing the total count of the data (one value).
* **mean\_ss**: A Tensor containing the mean sufficient statistics: the (possibly shifted) sum of the elements to average over.
* **variance\_ss**: A Tensor containing the variance sufficient statistics: the (possibly shifted) squared sum of the data to compute the variance over.
* **shift**: A Tensor containing the value by which the data is shifted for numerical stability, or None if no shift was performed.
* **name**: Name used to scope the operations that compute the moments.

#### Returns:

Two Tensor objects: mean and variance.

# tf.nn.pool

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

Performs an N-D pooling operation.

### Aliases:

* tf.compat.v2.nn.pool
* tf.nn.pool

tf.nn.pool(  
    input,  
    window\_shape,  
    pooling\_type,  
    strides=None,  
    padding='VALID',  
    data\_format=None,  
    dilations=None,  
    name=None  
)

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

In the case that data\_format does not start with "NC", computes for 0 <= b < batch\_size, 0 <= x[i] < output\_spatial\_shape[i], 0 <= c < num\_channels:

  output[b, x[0], ..., x[N-1], c] =  
    REDUCE\_{z[0], ..., z[N-1]}  
      input[b,  
            x[0] \* strides[0] - pad\_before[0] + dilation\_rate[0]\*z[0],  
            ...  
            x[N-1]\*strides[N-1] - pad\_before[N-1] + dilation\_rate[N-1]\*z[N-1],  
            c],

where the reduction function REDUCE depends on the value of pooling\_type, and pad\_before is defined based on the value of padding as described in the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution)for details. The reduction never includes out-of-bounds positions.

In the case that data\_format starts with "NC", the input and output are simply transposed as follows:

  pool(input, data\_format, \*\*kwargs) =  
    tf.transpose(pool(tf.transpose(input, [0] + range(2,N+2) + [1]),  
                      \*\*kwargs),  
                 [0, N+1] + range(1, N+1))

#### Args:

* **input**: Tensor of rank N+2, of shape [batch\_size] + input\_spatial\_shape + [num\_channels] if data\_format does not start with "NC" (default), or [batch\_size, num\_channels] + input\_spatial\_shape if data\_format starts with "NC". Pooling happens over the spatial dimensions only.
* **window\_shape**: Sequence of N ints >= 1.
* **pooling\_type**: Specifies pooling operation, must be "AVG" or "MAX".
* **strides**: Optional. Sequence of N ints >= 1. Defaults to [1]\*N. If any value of strides is > 1, then all values of dilation\_rate must be 1.
* **padding**: The padding algorithm, must be "SAME" or "VALID". Defaults to "SAME". See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: A string or None. Specifies whether the channel dimension of the input and output is the last dimension (default, or if data\_format does not start with "NC"), or the second dimension (if data\_format starts with "NC"). For N=1, the valid values are "NWC" (default) and "NCW". For N=2, the valid values are "NHWC" (default) and "NCHW". For N=3, the valid values are "NDHWC" (default) and "NCDHW".
* **dilations**: Optional. Dilation rate. List of N ints >= 1. Defaults to [1]\*N. If any value of dilation\_rate is > 1, then all values of strides must be 1.
* **name**: Optional. Name of the op.

#### Returns:

Tensor of rank N+2, of shape [batch\_size] + output\_spatial\_shape + [num\_channels]

if data\_format is None or does not start with "NC", or

[batch\_size, num\_channels] + output\_spatial\_shape

if data\_format starts with "NC", where output\_spatial\_shape depends on the value of padding:

If padding = "SAME": output\_spatial\_shape[i] = ceil(input\_spatial\_shape[i] / strides[i])

If padding = "VALID": output\_spatial\_shape[i] = ceil((input\_spatial\_shape[i] - (window\_shape[i] - 1) \* dilation\_rate[i]) / strides[i]).

#### Raises:

* **ValueError**: if arguments are invalid.

# tf.nn.relu

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

Computes rectified linear: max(features, 0).

### Aliases:

* tf.compat.v1.nn.relu
* tf.compat.v2.nn.relu
* tf.nn.relu

tf.nn.relu(  
    features,  
    name=None  
)

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

### Used in the guide:

* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Custom layers](https://www.tensorflow.org/beta/tutorials/eager/custom_layers)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)

#### Args:

* **features**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64, qint8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.nn.relu6

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/relu6#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/relu6#aliases)

Computes Rectified Linear 6: min(max(features, 0), 6).

### Aliases:

* tf.compat.v1.nn.relu6
* tf.compat.v2.nn.relu6
* tf.nn.relu6

tf.nn.relu6(  
    features,  
    name=None  
)

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

Source: [Convolutional Deep Belief Networks on CIFAR-10. A. Krizhevsky](http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf)

#### Args:

* **features**: A Tensor with type float, double, int32, int64, uint8, int16, or int8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as features.

# tf.nn.RNNCellDeviceWrapper

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDeviceWrapper#top_of_page)
* [Class RNNCellDeviceWrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDeviceWrapper#class_rnncelldevicewrapper)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDeviceWrapper#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDeviceWrapper#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDeviceWrapper#properties)

## Class RNNCellDeviceWrapper

Operator that ensures an RNNCell runs on a particular device.

### Aliases:

* Class tf.compat.v2.nn.RNNCellDeviceWrapper
* Class tf.nn.RNNCellDeviceWrapper

Defined in [python/keras/layers/rnn\_cell\_wrapper\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/rnn_cell_wrapper_v2.py).

## \_\_init\_\_

\_\_init\_\_(  
    \*args,  
    \*\*kwargs  
)

Construct a DeviceWrapper for cell with device device.

Ensures the wrapped cell is called with tf.device(device).

#### Args:

* **cell**: An instance of RNNCell.
* **device**: A device string or function, for passing to [tf.device](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/device).
* **\*\*kwargs**: dict of keyword arguments for base layer.

## Properties

### output\_size

### state\_size

## Methods

### get\_initial\_state

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

### zero\_state

zero\_state(  
    batch\_size,  
    dtype  
)

# tf.nn.RNNCellDropoutWrapper

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDropoutWrapper#top_of_page)
* [Class RNNCellDropoutWrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDropoutWrapper#class_rnncelldropoutwrapper)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDropoutWrapper#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDropoutWrapper#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellDropoutWrapper#properties)

## Class RNNCellDropoutWrapper

Operator adding dropout to inputs and outputs of the given cell.

### Aliases:

* Class tf.compat.v2.nn.RNNCellDropoutWrapper
* Class tf.nn.RNNCellDropoutWrapper

Defined in [python/keras/layers/rnn\_cell\_wrapper\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/rnn_cell_wrapper_v2.py).

## \_\_init\_\_

\_\_init\_\_(  
    \*args,  
    \*\*kwargs  
)

Create a cell with added input, state, and/or output dropout.

If variational\_recurrent is set to True (**NOT** the default behavior), then the same dropout mask is applied at every step, as described in: [A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. Y. Gal, Z. Ghahramani](https://arxiv.org/abs/1512.05287).

Otherwise a different dropout mask is applied at every time step.

Note, by default (unless a custom dropout\_state\_filter is provided), the memory state (ccomponent of any LSTMStateTuple) passing through a DropoutWrapper is never modified. This behavior is described in the above article.

#### Args:

* **cell**: an RNNCell, a projection to output\_size is added to it.
* **input\_keep\_prob**: unit Tensor or float between 0 and 1, input keep probability; if it is constant and 1, no input dropout will be added.
* **output\_keep\_prob**: unit Tensor or float between 0 and 1, output keep probability; if it is constant and 1, no output dropout will be added.
* **state\_keep\_prob**: unit Tensor or float between 0 and 1, output keep probability; if it is constant and 1, no output dropout will be added. State dropout is performed on the outgoing states of the cell. **Note** the state components to which dropout is applied when state\_keep\_prob is in (0, 1) are also determined by the argument dropout\_state\_filter\_visitor (e.g. by default dropout is never applied to the c component of an LSTMStateTuple).
* **variational\_recurrent**: Python bool. If True, then the same dropout pattern is applied across all time steps per run call. If this parameter is set, input\_size **must** be provided.
* **input\_size**: (optional) (possibly nested tuple of) TensorShape objects containing the depth(s) of the input tensors expected to be passed in to the DropoutWrapper. Required and used **iff**variational\_recurrent = True and input\_keep\_prob < 1.
* **dtype**: (optional) The dtype of the input, state, and output tensors. Required and used **iff**variational\_recurrent = True.
* **seed**: (optional) integer, the randomness seed.
* **dropout\_state\_filter\_visitor**: (optional), default: (see below). Function that takes any hierarchical level of the state and returns a scalar or depth=1 structure of Python booleans describing which terms in the state should be dropped out. In addition, if the function returns True, dropout is applied across this sublevel. If the function returns False, dropout is not applied across this entire sublevel. Default behavior: perform dropout on all terms except the memory (c) state of LSTMCellState objects, and don't try to apply dropout to TensorArrayobjects: def dropout\_state\_filter\_visitor(s): if isinstance(s, LSTMCellState): # Never perform dropout on the c state. return LSTMCellState(c=False, h=True) elif isinstance(s, TensorArray): return False return True
* **\*\*kwargs**: dict of keyword arguments for base layer.

#### Raises:

* **TypeError**: if cell is not an RNNCell, or keep\_state\_fn is provided but not callable.
* **ValueError**: if any of the keep\_probs are not between 0 and 1.

## Properties

### output\_size

### state\_size

### wrapped\_cell

## Methods

### get\_initial\_state

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

### zero\_state

zero\_state(  
    batch\_size,  
    dtype  
)

# tf.nn.RNNCellResidualWrapper

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellResidualWrapper#top_of_page)
* [Class RNNCellResidualWrapper](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellResidualWrapper#class_rnncellresidualwrapper)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellResidualWrapper#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellResidualWrapper#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/RNNCellResidualWrapper#properties)

## Class RNNCellResidualWrapper

RNNCell wrapper that ensures cell inputs are added to the outputs.

### Aliases:

* Class tf.compat.v2.nn.RNNCellResidualWrapper
* Class tf.nn.RNNCellResidualWrapper

Defined in [python/keras/layers/rnn\_cell\_wrapper\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/keras/layers/rnn_cell_wrapper_v2.py).

## \_\_init\_\_

\_\_init\_\_(  
    \*args,  
    \*\*kwargs  
)

Constructs a ResidualWrapper for cell.

#### Args:

* **cell**: An instance of RNNCell.
* **residual\_fn**: (Optional) The function to map raw cell inputs and raw cell outputs to the actual cell outputs of the residual network. Defaults to calling nest.map\_structure on (lambda i, o: i + o), inputs and outputs.
* **\*\*kwargs**: dict of keyword arguments for base layer.

## Properties

### output\_size

### state\_size

## Methods

### get\_initial\_state

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

### zero\_state

zero\_state(  
    batch\_size,  
    dtype  
)

# tf.nn.safe\_embedding\_lookup\_sparse

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

Lookup embedding results, accounting for invalid IDs and empty features.

### Aliases:

* tf.compat.v2.nn.safe\_embedding\_lookup\_sparse
* tf.nn.safe\_embedding\_lookup\_sparse

tf.nn.safe\_embedding\_lookup\_sparse(  
    embedding\_weights,  
    sparse\_ids,  
    sparse\_weights=None,  
    combiner='mean',  
    default\_id=None,  
    max\_norm=None,  
    name=None  
)

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

The partitioned embedding in embedding\_weights must all be the same shape except for the first dimension. The first dimension is allowed to vary as the vocabulary size is not necessarily a multiple of P. embedding\_weights may be a PartitionedVariable as returned by usingtf.compat.v1.get\_variable() with a partitioner.

Invalid IDs (< 0) are pruned from input IDs and weights, as well as any IDs with non-positive weight. For an entry with no features, the embedding vector for default\_id is returned, or the 0-vector if default\_id is not supplied.

The ids and weights may be multi-dimensional. Embeddings are always aggregated along the last dimension.

**Note:** when doing embedding lookup on **embedding\_weights**, "div" partition strategy will be used. Support for other partition strategy will be added later.

#### Args:

* **embedding\_weights**: A list of P float Tensors or values representing partitioned embedding Tensors. Alternatively, a PartitionedVariable created by partitioning along dimension 0. The total unpartitioned shape should be [e\_0, e\_1, ..., e\_m], where e\_0 represents the vocab size and e\_1, ..., e\_m are the embedding dimensions.
* **sparse\_ids**: SparseTensor of shape [d\_0, d\_1, ..., d\_n] containing the ids. d\_0 is typically batch size.
* **sparse\_weights**: SparseTensor of same shape as sparse\_ids, containing float weights corresponding to sparse\_ids, or None if all weights are be assumed to be 1.0.
* **combiner**: A string specifying how to combine embedding results for each entry. Currently "mean", "sqrtn" and "sum" are supported, with "mean" the default.
* **default\_id**: The id to use for an entry with no features.
* **max\_norm**: If not None, all embeddings are l2-normalized to max\_norm before combining.
* **name**: A name for this operation (optional).

#### Returns:

Dense Tensor of shape [d\_0, d\_1, ..., d\_{n-1}, e\_1, ..., e\_m].

#### Raises:

* **ValueError**: if embedding\_weights is empty.

# tf.nn.sampled\_softmax\_loss

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

Computes and returns the sampled softmax training loss.

### Aliases:

* tf.compat.v2.nn.sampled\_softmax\_loss
* tf.nn.sampled\_softmax\_loss

tf.nn.sampled\_softmax\_loss(  
    weights,  
    biases,  
    labels,  
    inputs,  
    num\_sampled,  
    num\_classes,  
    num\_true=1,  
    sampled\_values=None,  
    remove\_accidental\_hits=True,  
    seed=None,  
    name='sampled\_softmax\_loss'  
)

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

This is a faster way to train a softmax classifier over a huge number of classes.

This operation is for training only. It is generally an underestimate of the full softmax loss.

A common use case is to use this method for training, and calculate the full sigmoid loss for evaluation or inference as in the following example:

if mode == "train":  
  loss = tf.nn.sampled\_softmax\_loss(  
      weights=weights,  
      biases=biases,  
      labels=labels,  
      inputs=inputs,  
      ...)  
elif mode == "eval":  
  logits = tf.matmul(inputs, tf.transpose(weights))  
  logits = tf.nn.bias\_add(logits, biases)  
  labels\_one\_hot = tf.one\_hot(labels, n\_classes)  
  loss = tf.nn.softmax\_cross\_entropy\_with\_logits(  
      labels=labels\_one\_hot,  
      logits=logits)

See our [Candidate Sampling Algorithms Reference](https://www.tensorflow.org/extras/candidate_sampling.pdf)

Also see Section 3 of [Jean et al., 2014](http://arxiv.org/abs/1412.2007) ([pdf](http://arxiv.org/pdf/1412.2007.pdf)) for the math.

**Note:** when doing embedding lookup on **weights** and **bias**, "div" partition strategy will be used. Support for other partition strategy will be added later.

#### Args:

* **weights**: A Tensor of shape [num\_classes, dim], or a list of Tensor objects whose concatenation along dimension 0 has shape [num\_classes, dim]. The (possibly-sharded) class embeddings.
* **biases**: A Tensor of shape [num\_classes]. The class biases.
* **labels**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes. Note that this format differs from the labels argument of nn.softmax\_cross\_entropy\_with\_logits.
* **inputs**: A Tensor of shape [batch\_size, dim]. The forward activations of the input network.
* **num\_sampled**: An int. The number of classes to randomly sample per batch.
* **num\_classes**: An int. The number of possible classes.
* **num\_true**: An int. The number of target classes per training example.
* **sampled\_values**: a tuple of (sampled\_candidates, true\_expected\_count,sampled\_expected\_count) returned by a \*\_candidate\_sampler function. (if None, we default to log\_uniform\_candidate\_sampler)
* **remove\_accidental\_hits**: A bool. whether to remove "accidental hits" where a sampled class equals one of the target classes. Default is True.
* **seed**: random seed for candidate sampling. Default to None, which doesn't set the op-level random seed for candidate sampling.
* **name**: A name for the operation (optional).

#### Returns:

A batch\_size 1-D tensor of per-example sampled softmax losses.

# tf.nn.scale\_regularization\_loss

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

Scales the sum of the given regularization losses by number of replicas.

### Aliases:

* tf.compat.v1.nn.scale\_regularization\_loss
* tf.compat.v2.nn.scale\_regularization\_loss
* tf.nn.scale\_regularization\_loss

tf.nn.scale\_regularization\_loss(regularization\_loss)

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

Usage with distribution strategy and custom training loop:

with strategy.scope():  
  def compute\_loss(self, label, predictions):  
    per\_example\_loss = tf.keras.losses.sparse\_categorical\_crossentropy(  
        labels, predictions)  
  
    # Compute loss that is scaled by sample\_weight and by global batch size.  
    loss = tf.compute\_average\_loss(  
        per\_example\_loss,  
        sample\_weight=sample\_weight,  
        global\_batch\_size=GLOBAL\_BATCH\_SIZE)  
  
    # Add scaled regularization losses.  
    loss += tf.scale\_regularization\_loss(tf.nn.l2\_loss(weights))  
    return loss

#### Args:

* **regularization\_loss**: Regularization loss.

#### Returns:

Scalar loss value.

# tf.nn.selu

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

Computes scaled exponential linear: scale \* alpha \* (exp(features) - 1)

### Aliases:

* tf.compat.v1.nn.selu
* tf.compat.v2.nn.selu
* tf.nn.selu

tf.nn.selu(  
    features,  
    name=None  
)

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

if < 0, scale \* features otherwise.

To be used together with initializer = tf.variance\_scaling\_initializer(factor=1.0, mode='FAN\_IN'). For correct dropout, use tf.contrib.nn.alpha\_dropout.

See [Self-Normalizing Neural Networks](https://arxiv.org/abs/1706.02515)

#### Args:

* **features**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.nn.separable\_conv2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/separable_conv2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/separable_conv2d#aliases)

2-D convolution with separable filters.

### Aliases:

* tf.compat.v2.nn.separable\_conv2d
* tf.nn.separable\_conv2d

tf.nn.separable\_conv2d(  
    input,  
    depthwise\_filter,  
    pointwise\_filter,  
    strides,  
    padding,  
    data\_format=None,  
    dilations=None,  
    name=None  
)

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

Performs a depthwise convolution that acts separately on channels followed by a pointwise convolution that mixes channels. Note that this is separability between dimensions [1, 2] and 3, not spatial separability between dimensions 1 and 2.

In detail, with the default NHWC format,

output[b, i, j, k] = sum\_{di, dj, q, r}  
    input[b, strides[1] \* i + di, strides[2] \* j + dj, q] \*  
    depthwise\_filter[di, dj, q, r] \*  
    pointwise\_filter[0, 0, q \* channel\_multiplier + r, k]

strides controls the strides for the depthwise convolution only, since the pointwise convolution has implicit strides of [1, 1, 1, 1]. Must have strides[0] = strides[3] = 1. For the most common case of the same horizontal and vertical strides, strides = [1, stride, stride, 1]. If any value in rate is greater than 1, we perform atrous depthwise convolution, in which case all values in the strides tensor must be equal to 1.

#### Args:

* **input**: 4-D Tensor with shape according to data\_format.
* **depthwise\_filter**: 4-D Tensor with shape [filter\_height, filter\_width, in\_channels, channel\_multiplier]. Contains in\_channels convolutional filters of depth 1.
* **pointwise\_filter**: 4-D Tensor with shape [1, 1, channel\_multiplier \* in\_channels, out\_channels]. Pointwise filter to mix channels after depthwise\_filter has convolved spatially.
* **strides**: 1-D of size 4. The strides for the depthwise convolution for each dimension of input.
* **padding**: A string, either 'VALID' or 'SAME'. The padding algorithm. See the "returns" section of [tf.nn.convolution](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/convolution) for details.
* **data\_format**: The data format for input. Either "NHWC" (default) or "NCHW".
* **dilations**: 1-D of size 2. The dilation rate in which we sample input values across the heightand width dimensions in atrous convolution. If it is greater than 1, then all values of strides must be 1.
* **name**: A name for this operation (optional).

#### Returns:

A 4-D Tensor with shape according to 'data\_format'. For example, with data\_format="NHWC", shape is [batch, out\_height, out\_width, out\_channels].

# tf.nn.sigmoid\_cross\_entropy\_with\_logits

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

Computes sigmoid cross entropy given logits.

### Aliases:

* tf.compat.v2.nn.sigmoid\_cross\_entropy\_with\_logits
* tf.nn.sigmoid\_cross\_entropy\_with\_logits

tf.nn.sigmoid\_cross\_entropy\_with\_logits(  
    labels=None,  
    logits=None,  
    name=None  
)

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

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

Measures the probability error in discrete classification tasks in which each class is independent and not mutually exclusive. For instance, one could perform multilabel classification where a picture can contain both an elephant and a dog at the same time.

For brevity, let x = logits, z = labels. The logistic loss is

  z \* -log(sigmoid(x)) + (1 - z) \* -log(1 - sigmoid(x))  
= z \* -log(1 / (1 + exp(-x))) + (1 - z) \* -log(exp(-x) / (1 + exp(-x)))  
= z \* log(1 + exp(-x)) + (1 - z) \* (-log(exp(-x)) + log(1 + exp(-x)))  
= z \* log(1 + exp(-x)) + (1 - z) \* (x + log(1 + exp(-x))  
= (1 - z) \* x + log(1 + exp(-x))  
= x - x \* z + log(1 + exp(-x))

For x < 0, to avoid overflow in exp(-x), we reformulate the above

  x - x \* z + log(1 + exp(-x))  
= log(exp(x)) - x \* z + log(1 + exp(-x))  
= - x \* z + log(1 + exp(x))

Hence, to ensure stability and avoid overflow, the implementation uses this equivalent formulation

max(x, 0) - x \* z + log(1 + exp(-abs(x)))

logits and labels must have the same type and shape.

#### Args:

* **labels**: A Tensor of the same type and shape as logits.
* **logits**: A Tensor of type float32 or float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of the same shape as logits with the componentwise logistic losses.

#### Raises:

* **ValueError**: If logits and labels do not have the same shape.

# tf.nn.softmax

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

Computes softmax activations.

### Aliases:

* tf.compat.v2.math.softmax
* tf.compat.v2.nn.softmax
* tf.math.softmax
* tf.nn.softmax

tf.nn.softmax(  
    logits,  
    axis=None,  
    name=None  
)

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

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

This function performs the equivalent of

softmax = tf.exp(logits) / tf.reduce\_sum(tf.exp(logits), axis)

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type and shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.nn.softmax\_cross\_entropy\_with\_logits

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softmax_cross_entropy_with_logits#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softmax_cross_entropy_with_logits#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softmax_cross_entropy_with_logits#used_in_the_guide)

Computes softmax cross entropy between logits and labels.

### Aliases:

* tf.compat.v2.nn.softmax\_cross\_entropy\_with\_logits
* tf.nn.softmax\_cross\_entropy\_with\_logits

tf.nn.softmax\_cross\_entropy\_with\_logits(  
    labels,  
    logits,  
    axis=-1,  
    name=None  
)

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

### Used in the guide:

* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)

Measures the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class). For example, each CIFAR-10 image is labeled with one and only one label: an image can be a dog or a truck, but not both.

**NOTE:** While the classes are mutually exclusive, their probabilities need not be. All that is required is that each row of labels is a valid probability distribution. If they are not, the computation of the gradient will be incorrect.

If using exclusive labels (wherein one and only one class is true at a time), see sparse\_softmax\_cross\_entropy\_with\_logits.

**WARNING:** This op expects unscaled logits, since it performs a softmax on logits internally for efficiency. Do not call this op with the output of softmax, as it will produce incorrect results.

A common use case is to have logits and labels of shape [batch\_size, num\_classes], but higher dimensions are supported, with the axis argument specifying the class dimension.

logits and labels must have the same dtype (either float16, float32, or float64).

Backpropagation will happen into both logits and labels. To disallow backpropagation into labels, pass label tensors through [tf.stop\_gradient](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/stop_gradient) before feeding it to this function.

**Note that to avoid confusion, it is required to pass only named arguments to this function.**

#### Args:

* **labels**: Each vector along the class dimension should hold a valid probability distribution e.g. for the case in which labels are of shape [batch\_size, num\_classes], each row of labels[i] must be a valid probability distribution.
* **logits**: Per-label activations, typically a linear output. These activation energies are interpreted as unnormalized log probabilities.
* **axis**: The class dimension. Defaulted to -1 which is the last dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor that contains the softmax cross entropy loss. Its type is the same as logits and its shape is the same as labels except that it does not have the last dimension of labels.

# tf.nn.softsign

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

Computes softsign: features / (abs(features) + 1).

### Aliases:

* tf.compat.v1.math.softsign
* tf.compat.v1.nn.softsign
* tf.compat.v2.math.softsign
* tf.compat.v2.nn.softsign
* tf.math.softsign
* tf.nn.softsign

tf.nn.softsign(  
    features,  
    name=None  
)

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

#### Args:

* **features**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.nn.space\_to\_depth

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

SpaceToDepth for tensors of type T.

### Aliases:

* tf.compat.v2.nn.space\_to\_depth
* tf.nn.space\_to\_depth

tf.nn.space\_to\_depth(  
    input,  
    block\_size,  
    data\_format='NHWC',  
    name=None  
)

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

Rearranges blocks of spatial data, into depth. More specifically, this op outputs a copy of the input tensor where values from the height and width dimensions are moved to the depth dimension. The attr block\_size indicates the input block size.

* Non-overlapping blocks of size block\_size x block size are rearranged into depth at each location.
* The depth of the output tensor is block\_size \* block\_size \* input\_depth.
* The Y, X coordinates within each block of the input become the high order component of the output channel index.
* The input tensor's height and width must be divisible by block\_size.

The data\_format attr specifies the layout of the input and output tensors with the following options: "NHWC": [ batch, height, width, channels ] "NCHW": [ batch, channels, height, width ] "NCHW\_VECT\_C": qint8 [ batch, channels / 4, height, width, 4 ]

It is useful to consider the operation as transforming a 6-D Tensor. e.g. for data\_format = NHWC, Each element in the input tensor can be specified via 6 coordinates, ordered by decreasing memory layout significance as: n,oY,bY,oX,bX,iC (where n=batch index, oX, oY means X or Y coordinates within the output image, bX, bY means coordinates within the input block, iC means input channels). The output would be a transpose to the following layout: n,oY,oX,bY,bX,iC

This operation is useful for resizing the activations between convolutions (but keeping all data), e.g. instead of pooling. It is also useful for training purely convolutional models.

For example, given an input of shape [1, 2, 2, 1], data\_format = "NHWC" and block\_size = 2:

x = [[[[1], [2]],  
      [[3], [4]]]]

This operation will output a tensor of shape [1, 1, 1, 4]:

[[[[1, 2, 3, 4]]]]

Here, the input has a batch of 1 and each batch element has shape [2, 2, 1], the corresponding output will have a single element (i.e. width and height are both 1) and will have a depth of 4 channels (1 \* block\_size \* block\_size). The output element shape is [1, 1, 4].

For an input tensor with larger depth, here of shape [1, 2, 2, 3], e.g.

x = [[[[1, 2, 3], [4, 5, 6]],  
      [[7, 8, 9], [10, 11, 12]]]]

This operation, for block\_size of 2, will return the following tensor of shape [1, 1, 1, 12]

[[[[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]]]]

Similarly, for the following input of shape [1 4 4 1], and a block size of 2:

x = [[[[1],   [2],  [5],  [6]],  
      [[3],   [4],  [7],  [8]],  
      [[9],  [10], [13],  [14]],  
      [[11], [12], [15],  [16]]]]

the operator will return the following tensor of shape [1 2 2 4]:

x = [[[[1, 2, 3, 4],  
       [5, 6, 7, 8]],  
      [[9, 10, 11, 12],  
       [13, 14, 15, 16]]]]

#### Args:

* **input**: A Tensor.
* **block\_size**: An int that is >= 2. The size of the spatial block.
* **data\_format**: An optional string from: "NHWC", "NCHW", "NCHW\_VECT\_C". Defaults to "NHWC".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits

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

Computes sparse softmax cross entropy between logits and labels.

### Aliases:

* tf.compat.v2.nn.sparse\_softmax\_cross\_entropy\_with\_logits
* tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits

tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits(  
    labels,  
    logits,  
    name=None  
)

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

Measures the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class). For example, each CIFAR-10 image is labeled with one and only one label: an image can be a dog or a truck, but not both.

**NOTE:** For this operation, the probability of a given label is considered exclusive. That is, soft classes are not allowed, and the labels vector must provide a single specific index for the true class for each row of logits (each minibatch entry). For soft softmax classification with a probability distribution for each entry, see softmax\_cross\_entropy\_with\_logits\_v2.

**WARNING:** This op expects unscaled logits, since it performs a softmax on logits internally for efficiency. Do not call this op with the output of softmax, as it will produce incorrect results.

A common use case is to have logits of shape [batch\_size, num\_classes] and have labels of shape [batch\_size], but higher dimensions are supported, in which case the dim-th dimension is assumed to be of size num\_classes. logits must have the dtype of float16, float32, or float64, and labels must have the dtype of int32 or int64.

**Note that to avoid confusion, it is required to pass only named arguments to this function.**

#### Args:

* **labels**: Tensor of shape [d\_0, d\_1, ..., d\_{r-1}] (where r is rank of labels and result) and dtype int32 or int64. Each entry in labels must be an index in [0, num\_classes). Other values will raise an exception when this op is run on CPU, and return NaNfor corresponding loss and gradient rows on GPU.
* **logits**: Unscaled log probabilities of shape [d\_0, d\_1, ..., d\_{r-1}, num\_classes] and dtype float16, float32, or float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of the same shape as labels and of the same type as logits with the softmax cross entropy loss.

#### Raises:

* **ValueError**: If logits are scalars (need to have rank >= 1) or if the rank of the labels is not equal to the rank of the logits minus one.

# tf.nn.sufficient\_statistics

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

Calculate the sufficient statistics for the mean and variance of x.

### Aliases:

* tf.compat.v2.nn.sufficient\_statistics
* tf.nn.sufficient\_statistics

tf.nn.sufficient\_statistics(  
    x,  
    axes,  
    shift=None,  
    keepdims=False,  
    name=None  
)

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

These sufficient statistics are computed using the one pass algorithm on an input that's optionally shifted. See: https://en.wikipedia.org/wiki/Algorithms\_for\_calculating\_variance#Computing\_shifted\_data

#### Args:

* **x**: A Tensor.
* **axes**: Array of ints. Axes along which to compute mean and variance.
* **shift**: A Tensor containing the value by which to shift the data for numerical stability, or Noneif no shift is to be performed. A shift close to the true mean provides the most numerically stable results.
* **keepdims**: produce statistics with the same dimensionality as the input.
* **name**: Name used to scope the operations that compute the sufficient stats.

#### Returns:

Four Tensor objects of the same type as x:

* the count (number of elements to average over).
* the (possibly shifted) sum of the elements in the array.
* the (possibly shifted) sum of squares of the elements in the array.
* the shift by which the mean must be corrected or None if shift is None.

# tf.nn.weighted\_cross\_entropy\_with\_logits

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

Computes a weighted cross entropy.

### Aliases:

* tf.compat.v2.nn.weighted\_cross\_entropy\_with\_logits
* tf.nn.weighted\_cross\_entropy\_with\_logits

tf.nn.weighted\_cross\_entropy\_with\_logits(  
    labels,  
    logits,  
    pos\_weight,  
    name=None  
)

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

This is like sigmoid\_cross\_entropy\_with\_logits() except that pos\_weight, allows one to trade off recall and precision by up- or down-weighting the cost of a positive error relative to a negative error.

The usual cross-entropy cost is defined as:

labels \* -log(sigmoid(logits)) +  
    (1 - labels) \* -log(1 - sigmoid(logits))

A value pos\_weights > 1 decreases the false negative count, hence increasing the recall. Conversely setting pos\_weights < 1 decreases the false positive count and increases the precision. This can be seen from the fact that pos\_weight is introduced as a multiplicative coefficient for the positive labels term in the loss expression:

labels \* -log(sigmoid(logits)) \* pos\_weight +  
    (1 - labels) \* -log(1 - sigmoid(logits))

For brevity, let x = logits, z = labels, q = pos\_weight. The loss is:

  qz \* -log(sigmoid(x)) + (1 - z) \* -log(1 - sigmoid(x))  
= qz \* -log(1 / (1 + exp(-x))) + (1 - z) \* -log(exp(-x) / (1 + exp(-x)))  
= qz \* log(1 + exp(-x)) + (1 - z) \* (-log(exp(-x)) + log(1 + exp(-x)))  
= qz \* log(1 + exp(-x)) + (1 - z) \* (x + log(1 + exp(-x))  
= (1 - z) \* x + (qz +  1 - z) \* log(1 + exp(-x))  
= (1 - z) \* x + (1 + (q - 1) \* z) \* log(1 + exp(-x))

Setting l = (1 + (q - 1) \* z), to ensure stability and avoid overflow, the implementation uses

(1 - z) \* x + l \* (log(1 + exp(-abs(x))) + max(-x, 0))

logits and labels must have the same type and shape.

#### Args:

* **labels**: A Tensor of the same type and shape as logits.
* **logits**: A Tensor of type float32 or float64.
* **pos\_weight**: A coefficient to use on the positive examples.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of the same shape as logits with the componentwise weighted logistic losses.

#### Raises:

* **ValueError**: If logits and labels do not have the same shape.

# tf.nn.weighted\_moments

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

Returns the frequency-weighted mean and variance of x.

### Aliases:

* tf.compat.v2.nn.weighted\_moments
* tf.nn.weighted\_moments

tf.nn.weighted\_moments(  
    x,  
    axes,  
    frequency\_weights,  
    keepdims=False,  
    name=None  
)

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

#### Args:

* **x**: A tensor.
* **axes**: 1-d tensor of int32 values; these are the axes along which to compute mean and variance.
* **frequency\_weights**: A tensor of positive weights which can be broadcast with x.
* **keepdims**: Produce moments with the same dimensionality as the input.
* **name**: Name used to scope the operation.

#### Returns:

Two tensors: weighted\_mean and weighted\_variance.

# tf.nn.with\_space\_to\_batch

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

Performs op on the space-to-batch representation of input.

### Aliases:

* tf.compat.v1.nn.with\_space\_to\_batch
* tf.compat.v2.nn.with\_space\_to\_batch
* tf.nn.with\_space\_to\_batch

tf.nn.with\_space\_to\_batch(  
    input,  
    dilation\_rate,  
    padding,  
    op,  
    filter\_shape=None,  
    spatial\_dims=None,  
    data\_format=None  
)

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

This has the effect of transforming sliding window operations into the corresponding "atrous" operation in which the input is sampled at the specified dilation\_rate.

In the special case that dilation\_rate is uniformly 1, this simply returns:

op(input, num\_spatial\_dims, padding)

Otherwise, it returns:

batch\_to\_space\_nd( op(space\_to\_batch\_nd(input, adjusted\_dilation\_rate, adjusted\_paddings), num\_spatial\_dims, "VALID") adjusted\_dilation\_rate, adjusted\_crops),

where:

adjusted\_dilation\_rate is an int64 tensor of shape [max(spatialdims)], adjusted{paddings,crops} are int64 tensors of shape [max(spatial\_dims), 2]

defined as follows:

We first define two int64 tensors paddings and crops of shape [num\_spatial\_dims, 2] based on the value of padding and the spatial dimensions of the input:

If padding = "VALID", then:

paddings, crops = required\_space\_to\_batch\_paddings( input\_shape[spatial\_dims], dilation\_rate)

If padding = "SAME", then:

dilated\_filter\_shape = filter\_shape + (filter\_shape - 1) \* (dilation\_rate - 1)

paddings, crops = required\_space\_to\_batch\_paddings( input\_shape[spatial\_dims], dilation\_rate, [(dilated\_filter\_shape - 1) // 2, dilated\_filter\_shape - 1 - (dilated\_filter\_shape - 1) // 2])

Because space\_to\_batch\_nd and batch\_to\_space\_nd assume that the spatial dimensions are contiguous starting at the second dimension, but the specified spatial\_dims may not be, we must adjust dilation\_rate, paddings and crops in order to be usable with these operations. For a given dimension, if the block size is 1, and both the starting and ending padding and crop amounts are 0, then space\_to\_batch\_nd effectively leaves that dimension alone, which is what is needed for dimensions not part of spatial\_dims. Furthermore, space\_to\_batch\_nd and batch\_to\_space\_ndhandle this case efficiently for any number of leading and trailing dimensions.

For 0 <= i < len(spatial\_dims), we assign:

adjusted\_dilation\_rate[spatial\_dims[i] - 1] = dilation\_rate[i] adjusted\_paddings[spatial\_dims[i] - 1, :] = paddings[i, :] adjusted\_crops[spatial\_dims[i] - 1, :] = crops[i, :]

All unassigned values of adjusted\_dilation\_rate default to 1, while all unassigned values of adjusted\_paddings and adjusted\_crops default to 0.

Note in the case that dilation\_rate is not uniformly 1, specifying "VALID" padding is equivalent to specifying padding = "SAME" with a filter\_shape of [1]\*N.

Advanced usage. Note the following optimization: A sequence of with\_space\_to\_batch operations with identical (not uniformly 1) dilation\_rate parameters and "VALID" padding

net = with\_space\_to\_batch(net, dilation\_rate, "VALID", op\_1) ... net = with\_space\_to\_batch(net, dilation\_rate, "VALID", op\_k)

can be combined into a single with\_space\_to\_batch operation as follows:

def combined\_op(converted\_input, num\_spatial\_dims, \_): result = op\_1(converted\_input, num\_spatial\_dims, "VALID") ... result = op\_k(result, num\_spatial\_dims, "VALID")

net = with\_space\_to\_batch(net, dilation\_rate, "VALID", combined\_op)

This eliminates the overhead of k-1 calls to space\_to\_batch\_nd and batch\_to\_space\_nd.

Similarly, a sequence of with\_space\_to\_batch operations with identical (not uniformly 1) dilation\_rate parameters, "SAME" padding, and odd filter dimensions

net = with\_space\_to\_batch(net, dilation\_rate, "SAME", op\_1, filter\_shape\_1) ... net = with\_space\_to\_batch(net, dilation\_rate, "SAME", op\_k, filter\_shape\_k)

can be combined into a single with\_space\_to\_batch operation as follows:

def combined\_op(converted\_input, num\_spatial\_dims, \_): result = op\_1(converted\_input, num\_spatial\_dims, "SAME") ... result = op\_k(result, num\_spatial\_dims, "SAME")

net = with\_space\_to\_batch(net, dilation\_rate, "VALID", combined\_op)

#### Args:

* **input**: Tensor of rank > max(spatial\_dims).
* **dilation\_rate**: int32 Tensor of known shape [num\_spatial\_dims].
* **padding**: str constant equal to "VALID" or "SAME"
* **op**: Function that maps (input, num\_spatial\_dims, padding) -> output
* **filter\_shape**: If padding = "SAME", specifies the shape of the convolution kernel/pooling window as an integer Tensor of shape [>=num\_spatial\_dims]. If padding = "VALID", filter\_shape is ignored and need not be specified.
* **spatial\_dims**: Monotonically increasing sequence of num\_spatial\_dims integers (which are >= 1) specifying the spatial dimensions of input and output. Defaults to: range(1, num\_spatial\_dims+1).
* **data\_format**: A string or None. Specifies whether the channel dimension of the input and output is the last dimension (default, or if data\_format does not start with "NC"), or the second dimension (if data\_format starts with "NC"). For N=1, the valid values are "NWC" (default) and "NCW". For N=2, the valid values are "NHWC" (default) and "NCHW". For N=3, the valid values are "NDHWC" (default) and "NCDHW".

#### Returns:

The output Tensor as described above, dimensions will vary based on the op provided.

#### Raises:

* **ValueError**: if padding is invalid or the arguments are incompatible.
* **ValueError**: if spatial\_dims are invalid.

Module: tf.compat.v1.ragged / tf.ragged

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged#top_of_page)
* [Additional ops that support RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged#additional_ops_that_support_raggedtensor)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged#classes)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged#functions)

Ragged Tensors.

This package defines ops for manipulating ragged tensors ([tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor)), which are tensors with non-uniform shapes. In particular, each RaggedTensor has one or more *ragged dimensions*, which are dimensions whose slices may have different lengths. For example, the inner (column) dimension of rt=[[3, 1, 4, 1], [], [5, 9, 2], [6], []] is ragged, since the column slices (rt[0, :], ..., rt[4, :]) have different lengths. For a more detailed description of ragged tensors, see the [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) class documentation and the [Ragged Tensor Guide](https://www.tensorflow.org/guide/ragged_tensors).

Additional ops that support RaggedTensor

Arguments that accept RaggedTensors are marked in **bold**.

* tf.batch\_gather(**params**, **indices**, name=None)
* [tf.bitwise.bitwise\_and](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/bitwise_and)(**x**, **y**, name=None)
* [tf.bitwise.bitwise\_or](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/bitwise_or)(**x**, **y**, name=None)
* [tf.bitwise.bitwise\_xor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/bitwise_xor)(**x**, **y**, name=None)
* [tf.bitwise.invert](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/invert)(**x**, name=None)
* [tf.bitwise.left\_shift](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/left_shift)(**x**, **y**, name=None)
* [tf.bitwise.right\_shift](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/bitwise/right_shift)(**x**, **y**, name=None)
* [tf.clip\_by\_value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/clip_by_value)(**t**, clip\_value\_min, clip\_value\_max, name=None)
* [tf.concat](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/concat)(**values**, axis, name='concat')
* [tf.debugging.check\_numerics](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/debugging/check_numerics)(**tensor**, message, name=None)
* [tf.dtypes.cast](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/cast)(**x**, dtype, name=None)
* [tf.dtypes.complex](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/complex)(**real**, **imag**, name=None)
* [tf.dtypes.saturate\_cast](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/saturate_cast)(**value**, dtype, name=None)
* [tf.expand\_dims](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/expand_dims)(**input**, axis=None, name=None, dim=None)
* [tf.gather\_nd](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather_nd)(**params**, **indices**, name=None, batch\_dims=0)
* [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather)(**params**, **indices**, validate\_indices=None, name=None, axis=None, batch\_dims=0)
* [tf.identity](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/identity)(**input**, name=None)
* [tf.io.decode\_base64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_base64)(**input**, name=None)
* [tf.io.decode\_compressed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_compressed)(**bytes**, compression\_type='', name=None)
* [tf.io.encode\_base64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/encode_base64)(**input**, pad=False, name=None)
* [tf.math.abs](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs)(**x**, name=None)
* [tf.math.acos](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acos)(**x**, name=None)
* [tf.math.acosh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acosh)(**x**, name=None)
* [tf.math.add\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n)(**inputs**, name=None)
* [tf.math.add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add)(**x**, **y**, name=None)
* [tf.math.angle](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle)(**input**, name=None)
* [tf.math.asin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin)(**x**, name=None)
* [tf.math.asinh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asinh)(**x**, name=None)
* [tf.math.atan2](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2)(**y**, **x**, name=None)
* [tf.math.atan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan)(**x**, name=None)
* [tf.math.atanh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atanh)(**x**, name=None)
* [tf.math.ceil](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil)(**x**, name=None)
* [tf.math.conj](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/conj)(**x**, name=None)
* [tf.math.cos](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cos)(**x**, name=None)
* [tf.math.cosh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cosh)(**x**, name=None)
* [tf.math.digamma](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/digamma)(**x**, name=None)
* [tf.math.divide\_no\_nan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide_no_nan)(**x**, **y**, name=None)
* [tf.math.divide](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide)(**x**, **y**, name=None)
* [tf.math.equal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal)(**x**, **y**, name=None)
* [tf.math.erf](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erf)(**x**, name=None)
* [tf.math.erfc](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erfc)(**x**, name=None)
* [tf.math.exp](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp)(**x**, name=None)
* [tf.math.expm1](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1)(**x**, name=None)
* [tf.math.floor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floor)(**x**, name=None)
* [tf.math.floordiv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv)(**x**, **y**, name=None)
* [tf.math.floormod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod)(**x**, **y**, name=None)
* [tf.math.greater\_equal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater_equal)(**x**, **y**, name=None)
* [tf.math.greater](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater)(**x**, **y**, name=None)
* [tf.math.imag](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag)(**input**, name=None)
* [tf.math.is\_finite](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_finite)(**x**, name=None)
* [tf.math.is\_inf](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_inf)(**x**, name=None)
* [tf.math.is\_nan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/is_nan)(**x**, name=None)
* [tf.math.less\_equal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less_equal)(**x**, **y**, name=None)
* [tf.math.less](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less)(**x**, **y**, name=None)
* [tf.math.lgamma](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lgamma)(**x**, name=None)
* [tf.math.log1p](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p)(**x**, name=None)
* [tf.math.log\_sigmoid](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log_sigmoid)(**x**, name=None)
* [tf.math.log](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log)(**x**, name=None)
* [tf.math.logical\_and](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and)(**x**, **y**, name=None)
* [tf.math.logical\_not](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not)(**x**, name=None)
* [tf.math.logical\_or](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_or)(**x**, **y**, name=None)
* [tf.math.logical\_xor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_xor)(**x**, **y**, name='LogicalXor')
* [tf.math.maximum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum)(**x**, **y**, name=None)
* [tf.math.minimum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum)(**x**, **y**, name=None)
* [tf.math.multiply](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply)(**x**, **y**, name=None)
* [tf.math.negative](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/negative)(**x**, name=None)
* [tf.math.not\_equal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal)(**x**, **y**, name=None)
* [tf.math.pow](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow)(**x**, **y**, name=None)
* [tf.math.real](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/real)(**input**, name=None)
* [tf.math.reciprocal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reciprocal)(**x**, name=None)
* [tf.math.reduce\_any](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_any)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.reduce\_max](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_max)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.reduce\_min](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_min)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.reduce\_prod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_prod)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.reduce\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.math.rint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rint)(**x**, name=None)
* [tf.math.round](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/round)(**x**, name=None)
* [tf.math.rsqrt](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt)(**x**, name=None)
* [tf.math.sign](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sign)(**x**, name=None)
* [tf.math.sin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin)(**x**, name=None)
* [tf.math.sinh](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sinh)(**x**, name=None)
* [tf.math.sqrt](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt)(**x**, name=None)
* [tf.math.square](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square)(**x**, name=None)
* [tf.math.squared\_difference](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/squared_difference)(**x**, **y**, name=None)
* [tf.math.subtract](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/subtract)(**x**, **y**, name=None)
* [tf.math.tan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan)(**x**, name=None)
* [tf.math.truediv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/truediv)(**x**, **y**, name=None)
* [tf.math.unsorted\_segment\_max](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_max)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.math.unsorted\_segment\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_mean)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.math.unsorted\_segment\_min](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_min)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.math.unsorted\_segment\_prod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.math.unsorted\_segment\_sqrt\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sqrt_n)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.math.unsorted\_segment\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sum)(**data**, **segment\_ids**, num\_segments, name=None)
* [tf.ones\_like](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ones_like)(**tensor**, dtype=None, name=None, optimize=True)
* [tf.rank](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/rank)(**input**, name=None)
* [tf.realdiv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/realdiv)(**x**, **y**, name=None)
* [tf.reduce\_all](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reduce_all)(**input\_tensor**, axis=None, keepdims=False, name=None)
* [tf.size](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/size)(**input**, name=None, out\_type=[tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32))
* [tf.squeeze](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/squeeze)(**input**, axis=None, name=None, squeeze\_dims=None)
* [tf.stack](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/stack)(**values**, axis=0, name='stack')
* [tf.strings.as\_string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/as_string)(**input**, precision=-1, scientific=False, shortest=False, width=-1, fill='', name=None)
* [tf.strings.join](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join)(**inputs**, separator='', name=None)
* [tf.strings.length](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/length)(**input**, name=None, unit='BYTE')
* [tf.strings.regex\_full\_match](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_full_match)(**input**, pattern, name=None)
* [tf.strings.regex\_replace](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_replace)(**input**, pattern, rewrite, replace\_global=True, name=None)
* [tf.strings.strip](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/strip)(**input**, name=None)
* [tf.strings.substr](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/substr)(**input**, pos, len, name=None, unit='BYTE')
* [tf.strings.to\_hash\_bucket\_fast](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast)(**input**, num\_buckets, name=None)
* [tf.strings.to\_hash\_bucket\_strong](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_strong)(**input**, num\_buckets, key, name=None)
* [tf.strings.unicode\_script](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_script)(**input**, name=None)
* [tf.tile](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tile)(**input**, multiples, name=None)
* [tf.truncatediv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/truncatediv)(**x**, **y**, name=None)
* [tf.truncatemod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/truncatemod)(**x**, **y**, name=None)
* [tf.where](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/where)(**condition**, **x**=None, **y**=None, name=None)
* [tf.zeros\_like](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/zeros_like)(**tensor**, dtype=None, name=None, optimize=True)n

Classes

[class RaggedTensorValue](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue): Represents the value of a RaggedTensor.

Functions

[boolean\_mask(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/boolean_mask): Applies a boolean mask to data without flattening the mask dimensions.

[constant(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/constant): Constructs a constant RaggedTensor from a nested Python list.

[constant\_value(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/constant_value): Constructs a RaggedTensorValue from a nested Python list.

[map\_flat\_values(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/map_flat_values): Applies op to the values of one or more RaggedTensors.

[placeholder(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/placeholder): Creates a placeholder for a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) that will always be fed.

[range(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/range): Returns a RaggedTensor containing the specified sequences of numbers.

[row\_splits\_to\_segment\_ids(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/row_splits_to_segment_ids): Generates the segmentation corresponding to a RaggedTensor row\_splits.

[segment\_ids\_to\_row\_splits(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/segment_ids_to_row_splits): Generates the RaggedTensor row\_splits corresponding to a segmentation.

# tf.compat.v1.ragged.constant\_value

Constructs a RaggedTensorValue from a nested Python list.

tf.compat.v1.ragged.constant\_value(  
    pylist,  
    dtype=None,  
    ragged\_rank=None,  
    inner\_shape=None,  
    row\_splits\_dtype='int64'  
)

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

**Warning:** This function returns a **RaggedTensorValue**, not a **RaggedTensor**. If you wish to construct a constant **RaggedTensor**, use [**ragged.constant(...)**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/constant) instead.

#### Example:

>>> ragged.constant\_value([[1, 2], [3], [4, 5, 6]])  
RaggedTensorValue(values=[1, 2, 3, 4, 5, 6], splits=[0, 2, 3, 6])

All scalar values in pylist must have the same nesting depth K, and the returned RaggedTensorValue will have rank K. If pylist contains no scalar values, then K is one greater than the maximum depth of empty lists in pylist. All scalar values in pylist must be compatible with dtype.

#### Args:

* **pylist**: A nested list, tuple or np.ndarray. Any nested element that is not a list or tuple must be a scalar value compatible with dtype.
* **dtype**: numpy.dtype. The type of elements for the returned RaggedTensor. If not specified, then a default is chosen based on the scalar values in pylist.
* **ragged\_rank**: An integer specifying the ragged rank of the returned RaggedTensorValue. Must be nonnegative and less than K. Defaults to max(0, K - 1) if inner\_shape is not specified. Defaults to `max(0, K
  + 1 - len(inner\_shape))ifinner\_shape` is specified.
* **inner\_shape**: A tuple of integers specifying the shape for individual inner values in the returned RaggedTensorValue. Defaults to () if ragged\_rank is not specified. If ragged\_rank is specified, then a default is chosen based on the contents of pylist.
* **row\_splits\_dtype**: data type for the constructed RaggedTensorValue's row\_splits. One of numpy.int32 or numpy.int64.

#### Returns:

A tf.RaggedTensorValue or numpy.array with rank K and the specified ragged\_rank, containing the values from pylist.

#### Raises:

* **ValueError**: If the scalar values in pylist have inconsistent nesting depth; or if ragged\_rank or inner\_shape are incompatible with pylist.

# tf.compat.v1.ragged.placeholder

Creates a placeholder for a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) that will always be fed.

tf.compat.v1.ragged.placeholder(  
    dtype,  
    ragged\_rank,  
    value\_shape=None,  
    name=None  
)

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

**Important**: This ragged tensor will produce an error if evaluated. Its value must be fed using the feed\_dict optional argument to Session.run(), Tensor.eval(), or Operation.run().

@compatibility{eager} Placeholders are not compatible with eager execution.

#### Args:

* **dtype**: The data type for the RaggedTensor.
* **ragged\_rank**: The ragged rank for the RaggedTensor
* **value\_shape**: The shape for individual flat values in the RaggedTensor.
* **name**: A name for the operation (optional).

#### Returns:

A RaggedTensor that may be used as a handle for feeding a value, but not evaluated directly.

#### Raises:

* **RuntimeError**: if eager execution is enabled

# tf.compat.v1.ragged.RaggedTensorValue

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue#top_of_page)
* [Class RaggedTensorValue](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue#class_raggedtensorvalue)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue#properties)
  + [dtype](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/ragged/RaggedTensorValue#dtype)

## Class RaggedTensorValue

Represents the value of a RaggedTensor.

Defined in [python/ops/ragged/ragged\_tensor\_value.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/ragged/ragged_tensor_value.py).

**Warning:** **RaggedTensorValue** should only be used in graph mode; in eager mode, the [**tf.RaggedTensor**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) class contains its value directly.

See [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) for a description of ragged tensors.

## \_\_init\_\_

\_\_init\_\_(  
    values,  
    row\_splits  
)

Creates a RaggedTensorValue.

#### Args:

* **values**: A numpy array of any type and shape; or a RaggedTensorValue.
* **row\_splits**: A 1-D int32 or int64 numpy array.

## Properties

### dtype

The numpy dtype of values in this tensor.

### flat\_values

The innermost values array for this ragged tensor value.

### nested\_row\_splits

The row\_splits for all ragged dimensions in this ragged tensor value.

### ragged\_rank

The number of ragged dimensions in this ragged tensor value.

### row\_splits

The split indices for the ragged tensor value.

### shape

A tuple indicating the shape of this RaggedTensorValue.

### values

The concatenated values for all rows in this tensor.

## Methods

### to\_list

to\_list()

Returns this ragged tensor value as a nested Python list.

# tf.ragged.boolean\_mask

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

Applies a boolean mask to data without flattening the mask dimensions.

### Aliases:

* tf.compat.v1.ragged.boolean\_mask
* tf.compat.v2.ragged.boolean\_mask
* tf.ragged.boolean\_mask

tf.ragged.boolean\_mask(  
    data,  
    mask,  
    name=None  
)

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

Returns a potentially ragged tensor that is formed by retaining the elements in data where the corresponding value in mask is True.

* output[a1...aA, i, b1...bB] = data[a1...aA, j, b1...bB]

Where j is the ith True entry of mask[a1...aA].

Note that output preserves the mask dimensions a1...aA; this differs from [tf.boolean\_mask](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/boolean_mask), which flattens those dimensions.

#### Args:

* **data**: A potentially ragged tensor.
* **mask**: A potentially ragged boolean tensor. mask's shape must be a prefix of data's shape.rank(mask) must be known statically.
* **name**: A name prefix for the returned tensor (optional).

#### Returns:

A potentially ragged tensor that is formed by retaining the elements in data where the corresponding value in mask is True.

* rank(output) = rank(data).
* output.ragged\_rank = max(data.ragged\_rank, rank(mask) - 1).

#### Raises:

* **ValueError**: if rank(mask) is not known statically; or if mask.shape is not a prefix of data.shape.

#### Examples:

>>> # Aliases for True & False so data and mask line up.  
>>> T, F = (True, False)  
  
>>> tf.ragged.boolean\_mask(  # Mask a 2D Tensor.  
...     data=[[1, 2, 3], [4, 5, 6], [7, 8, 9]],  
...     mask=[[T, F, T], [F, F, F], [T, F, F]]).tolist()  
[[1, 3], [], [7]]  
  
>>> tf.ragged.boolean\_mask(  # Mask a 2D RaggedTensor.  
...     tf.ragged.constant([[1, 2, 3], [4], [5, 6]]),  
...     tf.ragged.constant([[F, F, T], [F], [T, T]])).tolist()  
[[3], [], [5, 6]]  
  
>>> tf.ragged.boolean\_mask(  # Mask rows of a 2D RaggedTensor.  
...     tf.ragged.constant([[1, 2, 3], [4], [5, 6]]),  
...     tf.ragged.constant([True, False, True])).tolist()  
[[1, 2, 3], [5, 6]]

# tf.ragged.constant

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/constant#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/constant#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/constant#used_in_the_guide)

Constructs a constant RaggedTensor from a nested Python list.

### Aliases:

* tf.compat.v1.ragged.constant
* tf.compat.v2.ragged.constant
* tf.ragged.constant

tf.ragged.constant(  
    pylist,  
    dtype=None,  
    ragged\_rank=None,  
    inner\_shape=None,  
    name=None,  
    row\_splits\_dtype=tf.dtypes.int64  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

#### Example:

>>> ragged.constant([[1, 2], [3], [4, 5, 6]]).eval()  
RaggedTensorValue(values=[1, 2, 3, 4, 5, 6], splits=[0, 2, 3, 6])

All scalar values in pylist must have the same nesting depth K, and the returned RaggedTensorwill have rank K. If pylist contains no scalar values, then K is one greater than the maximum depth of empty lists in pylist. All scalar values in pylist must be compatible with dtype.

#### Args:

* **pylist**: A nested list, tuple or np.ndarray. Any nested element that is not a list, tuple or np.ndarray must be a scalar value compatible with dtype.
* **dtype**: The type of elements for the returned RaggedTensor. If not specified, then a default is chosen based on the scalar values in pylist.
* **ragged\_rank**: An integer specifying the ragged rank of the returned RaggedTensor. Must be nonnegative and less than K. Defaults to max(0, K - 1) if inner\_shape is not specified. Defaults to `max(0, K
  + 1 - len(inner\_shape))ifinner\_shape` is specified.
* **inner\_shape**: A tuple of integers specifying the shape for individual inner values in the returned RaggedTensor. Defaults to () if ragged\_rank is not specified. If ragged\_rank is specified, then a default is chosen based on the contents of pylist.
* **name**: A name prefix for the returned tensor (optional).
* **row\_splits\_dtype**: data type for the constructed RaggedTensor's row\_splits. One of [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).

#### Returns:

A potentially ragged tensor with rank K and the specified ragged\_rank, containing the values from pylist.

#### Raises:

* **ValueError**: If the scalar values in pylist have inconsistent nesting depth; or if ragged\_rank or inner\_shape are incompatible with pylist.

# tf.ragged.map\_flat\_values

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/map_flat_values#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/map_flat_values#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/ragged/map_flat_values#used_in_the_guide)

Applies op to the values of one or more RaggedTensors.

### Aliases:

* tf.compat.v1.ragged.map\_flat\_values
* tf.compat.v2.ragged.map\_flat\_values
* tf.ragged.map\_flat\_values

tf.ragged.map\_flat\_values(  
    op,  
    \*args,  
    \*\*kwargs  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

Replaces any RaggedTensor in args or kwargs with its flat\_values tensor, and then calls op. Returns a RaggedTensor that is constructed from the input RaggedTensors' nested\_row\_splitsand the value returned by the op.

If the input arguments contain multiple RaggedTensors, then they must have identical nested\_row\_splits.

#### Examples:

>>> rt = ragged.constant([[1, 2, 3], [], [4, 5], [6]])  
>>> ragged.map\_flat\_values(tf.ones\_like, rt).eval().tolist()  
[[1, 1, 1], [], [1, 1], [1]]  
>>> ragged.map\_flat\_values(tf.multiply, rt, rt).eval().tolist()  
[[1, 4, 9], [], [16, 25], [36]]  
>>> ragged.map\_flat\_values(tf.add, rt, 5).eval().tolist()  
[[6, 7, 8], [], [9, 10], [11]]

#### Args:

* **op**: The operation that should be applied to the RaggedTensor flat\_values. op is typically an element-wise operation (such as math\_ops.add), but any operation that preserves the size of the outermost dimension can be used. I.e., shape[0] of the value returned by op must matchshape[0] of the RaggedTensors' flat\_values tensors.
* **\*args**: Arguments for op.
* **\*\*kwargs**: Keyword arguments for op.

#### Returns:

A RaggedTensor whose ragged\_rank matches the ragged\_rank of all input RaggedTensors.

#### Raises:

* **ValueError**: If args contains no RaggedTensors, or if the nested\_splits of the input RaggedTensors are not identical.

# tf.ragged.range

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

Returns a RaggedTensor containing the specified sequences of numbers.

### Aliases:

* tf.compat.v1.ragged.range
* tf.compat.v2.ragged.range
* tf.ragged.range

tf.ragged.range(  
    starts,  
    limits=None,  
    deltas=1,  
    dtype=None,  
    name=None,  
    row\_splits\_dtype=tf.dtypes.int64  
)

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

Each row of the returned RaggedTensor contains a single sequence:

ragged.range(starts, limits, deltas)[i] ==  
    tf.range(starts[i], limits[i], deltas[i])

If start[i] < limits[i] and deltas[i] > 0, then output[i] will be an empty list. Similarly, if start[i] > limits[i] and deltas[i] < 0, then output[i] will be an empty list. This behavior is consistent with the Python range function, but differs from the [tf.range](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/range) op, which returns an error for these cases.

#### Examples:

>>> ragged.range([3, 5, 2]).eval().tolist()  
[[0, 1, 2], [0, 1, 2, 3, 4], [0, 1]]  
>>> ragged.range([0, 5, 8], [3, 3, 12]).eval().tolist()  
[[0, 1, 2], [], [8, 9, 10, 11]]  
>>> ragged.range([0, 5, 8], [3, 3, 12], 2).eval().tolist()  
[[0, 2], [], [8, 10]]

The input tensors starts, limits, and deltas may be scalars or vectors. The vector inputs must all have the same size. Scalar inputs are broadcast to match the size of the vector inputs.

#### Args:

* **starts**: Vector or scalar Tensor. Specifies the first entry for each range if limits is not None; otherwise, specifies the range limits, and the first entries default to 0.
* **limits**: Vector or scalar Tensor. Specifies the exclusive upper limits for each range.
* **deltas**: Vector or scalar Tensor. Specifies the increment for each range. Defaults to 1.
* **dtype**: The type of the elements of the resulting tensor. If not specified, then a value is chosen based on the other args.
* **name**: A name for the operation.
* **row\_splits\_dtype**: dtype for the returned RaggedTensor's row\_splits tensor. One of [tf.int32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int32) or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).

#### Returns:

A RaggedTensor of type dtype with ragged\_rank=1.

# tf.ragged.row\_splits\_to\_segment\_ids

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

Generates the segmentation corresponding to a RaggedTensor row\_splits.

### Aliases:

* tf.compat.v1.ragged.row\_splits\_to\_segment\_ids
* tf.compat.v2.ragged.row\_splits\_to\_segment\_ids
* tf.ragged.row\_splits\_to\_segment\_ids

tf.ragged.row\_splits\_to\_segment\_ids(  
    splits,  
    name=None,  
    out\_type=None  
)

Defined in [python/ops/ragged/segment\_id\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/ragged/segment_id_ops.py).

Returns an integer vector segment\_ids, where segment\_ids[i] == j if splits[j] <= i < splits[j+1]. Example:

>>> ragged.row\_splits\_to\_segment\_ids([0, 3, 3, 5, 6, 9]).eval()  
[ 0 0 0 2 2 3 4 4 4 ]

#### Args:

* **splits**: A sorted 1-D integer Tensor. splits[0] must be zero.
* **name**: A name prefix for the returned tensor (optional).
* **out\_type**: The dtype for the return value. Defaults to splits.dtype, or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) if splitsdoes not have a dtype.

#### Returns:

A sorted 1-D integer Tensor, with shape=[splits[-1]]

#### Raises:

* **ValueError**: If splits is invalid.

# tf.ragged.segment\_ids\_to\_row\_splits

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

Generates the RaggedTensor row\_splits corresponding to a segmentation.

### Aliases:

* tf.compat.v1.ragged.segment\_ids\_to\_row\_splits
* tf.compat.v2.ragged.segment\_ids\_to\_row\_splits
* tf.ragged.segment\_ids\_to\_row\_splits

tf.ragged.segment\_ids\_to\_row\_splits(  
    segment\_ids,  
    num\_segments=None,  
    out\_type=None,  
    name=None  
)

Defined in [python/ops/ragged/segment\_id\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/ragged/segment_id_ops.py).

Returns an integer vector splits, where splits[0] = 0 and splits[i] = splits[i-1] + count(segment\_ids==i). Example:

>>> ragged.segment\_ids\_to\_row\_splits([0, 0, 0, 2, 2, 3, 4, 4, 4]).eval()  
[ 0 3 3 5 6 9 ]

#### Args:

* **segment\_ids**: A 1-D integer Tensor.
* **num\_segments**: A scalar integer indicating the number of segments. Defaults to max(segment\_ids) + 1 (or zero if segment\_ids is empty).
* **out\_type**: The dtype for the return value. Defaults to segment\_ids.dtype, or [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64) if segment\_ids does not have a dtype.
* **name**: A name prefix for the returned tensor (optional).

#### Returns:

A sorted 1-D integer Tensor, with shape=[num\_segments + 1].

Module: tf.compat.v1.random / tf.random

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

Public API for tf.random namespace.

Modules

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

Functions

[all\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/all_candidate_sampler): Generate the set of all classes.

[categorical(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/categorical): Draws samples from a categorical distribution.

[fixed\_unigram\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/fixed_unigram_candidate_sampler): Samples a set of classes using the provided (fixed) base distribution.

[gamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/gamma): Draws shape samples from each of the given Gamma distribution(s).

[get\_seed(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/get_seed): Returns the local seeds an operation should use given an op-specific seed.

[learned\_unigram\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/learned_unigram_candidate_sampler): Samples a set of classes from a distribution learned during training.

[log\_uniform\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/log_uniform_candidate_sampler): Samples a set of classes using a log-uniform (Zipfian) base distribution.

[multinomial(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/multinomial): Draws samples from a multinomial distribution. (deprecated)

[normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/normal): Outputs random values from a normal distribution.

[poisson(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/random_poisson): Draws shape samples from each of the given Poisson distribution(s).

[set\_random\_seed(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed): Sets the graph-level random seed for the default graph.

[shuffle(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/shuffle): Randomly shuffles a tensor along its first dimension.

[stateless\_categorical(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/stateless_categorical): Draws deterministic pseudorandom samples from a categorical distribution.

[stateless\_multinomial(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/random/stateless_multinomial): Draws deterministic pseudorandom samples from a multinomial distribution. (deprecated)

[stateless\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/stateless_normal): Outputs deterministic pseudorandom values from a normal distribution.

[stateless\_truncated\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/stateless_truncated_normal): Outputs deterministic pseudorandom values, truncated normally distributed.

[stateless\_uniform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/stateless_uniform): Outputs deterministic pseudorandom values from a uniform distribution.

[truncated\_normal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal): Outputs random values from a truncated normal distribution.

[uniform(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/uniform): Outputs random values from a uniform distribution.

[uniform\_candidate\_sampler(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/uniform_candidate_sampler): Samples a set of classes using a uniform base distribution.

# tf.compat.v1.random.stateless\_multinomial

Draws deterministic pseudorandom samples from a multinomial distribution. (deprecated)

tf.compat.v1.random.stateless\_multinomial(  
    logits,  
    num\_samples,  
    seed,  
    output\_dtype=tf.dtypes.int64,  
    name=None  
)

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

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use [**tf.random.stateless\_categorical**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/stateless_categorical) instead.

This is a stateless version of [tf.random.categorical](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/categorical): if run twice with the same seeds, it will produce the same pseudorandom numbers. The output is consistent across multiple runs on the same hardware (and between CPU and GPU), but may change between versions of TensorFlow or on non-CPU/GPU hardware.

#### Example:

# samples has shape [1, 5], where each value is either 0 or 1 with equal  
# probability.  
samples = tf.random.stateless\_categorical(  
    tf.math.log([[10., 10.]]), 5, seed=[7, 17])

#### Args:

* **logits**: 2-D Tensor with shape [batch\_size, num\_classes]. Each slice [i, :] represents the unnormalized log-probabilities for all classes.
* **num\_samples**: 0-D. Number of independent samples to draw for each row slice.
* **seed**: A shape [2] integer Tensor of seeds to the random number generator.
* **output\_dtype**: integer type to use for the output. Defaults to int64.
* **name**: Optional name for the operation.

#### Returns:

The drawn samples of shape [batch\_size, num\_samples].

# tf.random.all\_candidate\_sampler

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

Generate the set of all classes.

### Aliases:

* tf.compat.v1.nn.all\_candidate\_sampler
* tf.compat.v1.random.all\_candidate\_sampler
* tf.compat.v2.nn.all\_candidate\_sampler
* tf.compat.v2.random.all\_candidate\_sampler
* tf.nn.all\_candidate\_sampler
* tf.random.all\_candidate\_sampler

tf.random.all\_candidate\_sampler(  
    true\_classes,  
    num\_true,  
    num\_sampled,  
    unique,  
    seed=None,  
    name=None  
)

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

Deterministically generates and returns the set of all possible classes. For testing purposes. There is no need to use this, since you might as well use full softmax or full logistic regression.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **num\_true**: An int. The number of target classes per training example.
* **num\_sampled**: An int. The number of possible classes.
* **unique**: A bool. Ignored. unique.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. This operation deterministically returns the entire range [0, num\_sampled].
* **true\_expected\_count**: A tensor of type float. Same shape as true\_classes. The expected counts under the sampling distribution of each of true\_classes. All returned values are 1.0.
* **sampled\_expected\_count**: A tensor of type float. Same shape as sampled\_candidates. The expected counts under the sampling distribution of each of sampled\_candidates. All returned values are 1.0.

# tf.random.categorical

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

Draws samples from a categorical distribution.

### Aliases:

* tf.compat.v1.random.categorical
* tf.compat.v2.random.categorical
* tf.random.categorical

tf.random.categorical(  
    logits,  
    num\_samples,  
    dtype=None,  
    seed=None,  
    name=None  
)

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

### Used in the tutorials:

* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)

#### Example:

# samples has shape [1, 5], where each value is either 0 or 1 with equal  
# probability.  
samples = tf.random.categorical(tf.math.log([[10., 10.]]), 5)

#### Args:

* **logits**: 2-D Tensor with shape [batch\_size, num\_classes]. Each slice [i, :] represents the unnormalized log-probabilities for all classes.
* **num\_samples**: 0-D. Number of independent samples to draw for each row slice.
* **dtype**: integer type to use for the output. Defaults to int64.
* **seed**: A Python integer. Used to create a random seed for the distribution. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: Optional name for the operation.

#### Returns:

The drawn samples of shape [batch\_size, num\_samples].

# tf.random.fixed\_unigram\_candidate\_sampler

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

Samples a set of classes using the provided (fixed) base distribution.

### Aliases:

* tf.compat.v1.nn.fixed\_unigram\_candidate\_sampler
* tf.compat.v1.random.fixed\_unigram\_candidate\_sampler
* tf.compat.v2.nn.fixed\_unigram\_candidate\_sampler
* tf.compat.v2.random.fixed\_unigram\_candidate\_sampler
* tf.nn.fixed\_unigram\_candidate\_sampler
* tf.random.fixed\_unigram\_candidate\_sampler

tf.random.fixed\_unigram\_candidate\_sampler(  
    true\_classes,  
    num\_true,  
    num\_sampled,  
    unique,  
    range\_max,  
    vocab\_file='',  
    distortion=1.0,  
    num\_reserved\_ids=0,  
    num\_shards=1,  
    shard=0,  
    unigrams=(),  
    seed=None,  
    name=None  
)

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

This operation randomly samples a tensor of sampled classes (sampled\_candidates) from the range of integers [0, range\_max).

The elements of sampled\_candidates are drawn without replacement (if unique=True) or with replacement (if unique=False) from the base distribution.

The base distribution is read from a file or passed in as an in-memory array. There is also an option to skew the distribution by applying a distortion power to the weights.

In addition, this operation returns tensors true\_expected\_count and sampled\_expected\_countrepresenting the number of times each of the target classes (true\_classes) and the sampled classes (sampled\_candidates) is expected to occur in an average tensor of sampled classes. These values correspond to Q(y|x) defined in [this document](http://www.tensorflow.org/extras/candidate_sampling.pdf). If unique=True, then these are post-rejection probabilities and we compute them approximately.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **num\_true**: An int. The number of target classes per training example.
* **num\_sampled**: An int. The number of classes to randomly sample.
* **unique**: A bool. Determines whether all sampled classes in a batch are unique.
* **range\_max**: An int. The number of possible classes.
* **vocab\_file**: Each valid line in this file (which should have a CSV-like format) corresponds to a valid word ID. IDs are in sequential order, starting from num\_reserved\_ids. The last entry in each line is expected to be a value corresponding to the count or relative probability. Exactly one of vocab\_file and unigrams needs to be passed to this operation.
* **distortion**: The distortion is used to skew the unigram probability distribution. Each weight is first raised to the distortion's power before adding to the internal unigram distribution. As a result,distortion = 1.0 gives regular unigram sampling (as defined by the vocab file), and distortion = 0.0 gives a uniform distribution.
* **num\_reserved\_ids**: Optionally some reserved IDs can be added in the range [0, num\_reserved\_ids) by the users. One use case is that a special unknown word token is used as ID 0. These IDs will have a sampling probability of 0.
* **num\_shards**: A sampler can be used to sample from a subset of the original range in order to speed up the whole computation through parallelism. This parameter (together with shard) indicates the number of partitions that are being used in the overall computation.
* **shard**: A sampler can be used to sample from a subset of the original range in order to speed up the whole computation through parallelism. This parameter (together with num\_shards) indicates the particular partition number of the operation, when partitioning is being used.
* **unigrams**: A list of unigram counts or probabilities, one per ID in sequential order. Exactly one of vocab\_file and unigrams should be passed to this operation.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. The sampled classes.
* **true\_expected\_count**: A tensor of type float. Same shape as true\_classes. The expected counts under the sampling distribution of each of true\_classes.
* **sampled\_expected\_count**: A tensor of type float. Same shape as sampled\_candidates. The expected counts under the sampling distribution of each of sampled\_candidates.

# tf.random.gamma

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

Draws shape samples from each of the given Gamma distribution(s).

### Aliases:

* tf.compat.v1.random.gamma
* tf.compat.v1.random\_gamma
* tf.compat.v2.random.gamma
* tf.random.gamma

tf.random.gamma(  
    shape,  
    alpha,  
    beta=None,  
    dtype=tf.dtypes.float32,  
    seed=None,  
    name=None  
)

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

alpha is the shape parameter describing the distribution(s), and beta is the inverse scale parameter(s).

**Note:** Because internal calculations are done using **float64** and casting has **floor** semantics, we must manually map zero outcomes to the smallest possible positive floating-point value, i.e., **np.finfo(dtype).tiny**. This means that **np.finfo(dtype).tiny** occurs more frequently than it otherwise should. This bias can only happen for small values of **alpha**, i.e., **alpha << 1** or large values of **beta**, i.e., **beta >> 1**.

The samples are differentiable w.r.t. alpha and beta. The derivatives are computed using the approach described in the paper

[Michael Figurnov, Shakir Mohamed, Andriy Mnih. Implicit Reparameterization Gradients, 2018](https://arxiv.org/abs/1805.08498)

#### Example:

samples = tf.random.gamma([10], [0.5, 1.5])  
# samples has shape [10, 2], where each slice [:, 0] and [:, 1] represents  
# the samples drawn from each distribution  
  
samples = tf.random.gamma([7, 5], [0.5, 1.5])  
# samples has shape [7, 5, 2], where each slice [:, :, 0] and [:, :, 1]  
# represents the 7x5 samples drawn from each of the two distributions  
  
alpha = tf.constant([[1.],[3.],[5.]])  
beta = tf.constant([[3., 4.]])  
samples = tf.random.gamma([30], alpha=alpha, beta=beta)  
# samples has shape [30, 3, 2], with 30 samples each of 3x2 distributions.  
  
loss = tf.reduce\_mean(tf.square(samples))  
dloss\_dalpha, dloss\_dbeta = tf.gradients(loss, [alpha, beta])  
# unbiased stochastic derivatives of the loss function  
alpha.shape == dloss\_dalpha.shape  # True  
beta.shape == dloss\_dbeta.shape  # True

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output samples to be drawn per alpha/beta-parameterized distribution.
* **alpha**: A Tensor or Python value or N-D array of type dtype. alpha provides the shape parameter(s) describing the gamma distribution(s) to sample. Must be broadcastable with beta.
* **beta**: A Tensor or Python value or N-D array of type dtype. Defaults to 1. beta provides the inverse scale parameter(s) of the gamma distribution(s) to sample. Must be broadcastable with alpha.
* **dtype**: The type of alpha, beta, and the output: float16, float32, or float64.
* **seed**: A Python integer. Used to create a random seed for the distributions. See[tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: Optional name for the operation.

#### Returns:

* **samples**: a Tensor of shape tf.concat([shape, tf.shape(alpha + beta)], axis=0)with values of type dtype.

# tf.random.learned\_unigram\_candidate\_sampler

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

Samples a set of classes from a distribution learned during training.

### Aliases:

* tf.compat.v1.nn.learned\_unigram\_candidate\_sampler
* tf.compat.v1.random.learned\_unigram\_candidate\_sampler
* tf.compat.v2.nn.learned\_unigram\_candidate\_sampler
* tf.compat.v2.random.learned\_unigram\_candidate\_sampler
* tf.nn.learned\_unigram\_candidate\_sampler
* tf.random.learned\_unigram\_candidate\_sampler

tf.random.learned\_unigram\_candidate\_sampler(  
    true\_classes,  
    num\_true,  
    num\_sampled,  
    unique,  
    range\_max,  
    seed=None,  
    name=None  
)

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

This operation randomly samples a tensor of sampled classes (sampled\_candidates) from the range of integers [0, range\_max).

The elements of sampled\_candidates are drawn without replacement (if unique=True) or with replacement (if unique=False) from the base distribution.

The base distribution for this operation is constructed on the fly during training. It is a unigram distribution over the target classes seen so far during training. Every integer in [0, range\_max)begins with a weight of 1, and is incremented by 1 each time it is seen as a target class. The base distribution is not saved to checkpoints, so it is reset when the model is reloaded.

In addition, this operation returns tensors true\_expected\_count and sampled\_expected\_countrepresenting the number of times each of the target classes (true\_classes) and the sampled classes (sampled\_candidates) is expected to occur in an average tensor of sampled classes. These values correspond to Q(y|x) defined in [this document](http://www.tensorflow.org/extras/candidate_sampling.pdf). If unique=True, then these are post-rejection probabilities and we compute them approximately.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **num\_true**: An int. The number of target classes per training example.
* **num\_sampled**: An int. The number of classes to randomly sample.
* **unique**: A bool. Determines whether all sampled classes in a batch are unique.
* **range\_max**: An int. The number of possible classes.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. The sampled classes.
* **true\_expected\_count**: A tensor of type float. Same shape as true\_classes. The expected counts under the sampling distribution of each of true\_classes.
* **sampled\_expected\_count**: A tensor of type float. Same shape as sampled\_candidates. The expected counts under the sampling distribution of each of sampled\_candidates.

# tf.random.log\_uniform\_candidate\_sampler

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

Samples a set of classes using a log-uniform (Zipfian) base distribution.

### Aliases:

* tf.compat.v1.nn.log\_uniform\_candidate\_sampler
* tf.compat.v1.random.log\_uniform\_candidate\_sampler
* tf.compat.v2.random.log\_uniform\_candidate\_sampler
* tf.random.log\_uniform\_candidate\_sampler

tf.random.log\_uniform\_candidate\_sampler(  
    true\_classes,  
    num\_true,  
    num\_sampled,  
    unique,  
    range\_max,  
    seed=None,  
    name=None  
)

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

This operation randomly samples a tensor of sampled classes (sampled\_candidates) from the range of integers [0, range\_max).

The elements of sampled\_candidates are drawn without replacement (if unique=True) or with replacement (if unique=False) from the base distribution.

The base distribution for this operation is an approximately log-uniform or Zipfian distribution:

P(class) = (log(class + 2) - log(class + 1)) / log(range\_max + 1)

This sampler is useful when the target classes approximately follow such a distribution - for example, if the classes represent words in a lexicon sorted in decreasing order of frequency. If your classes are not ordered by decreasing frequency, do not use this op.

In addition, this operation returns tensors true\_expected\_count and sampled\_expected\_countrepresenting the number of times each of the target classes (true\_classes) and the sampled classes (sampled\_candidates) is expected to occur in an average tensor of sampled classes. These values correspond to Q(y|x) defined in [this document](http://www.tensorflow.org/extras/candidate_sampling.pdf). If unique=True, then these are post-rejection probabilities and we compute them approximately.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **num\_true**: An int. The number of target classes per training example.
* **num\_sampled**: An int. The number of classes to randomly sample.
* **unique**: A bool. Determines whether all sampled classes in a batch are unique.
* **range\_max**: An int. The number of possible classes.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. The sampled classes.
* **true\_expected\_count**: A tensor of type float. Same shape as true\_classes. The expected counts under the sampling distribution of each of true\_classes.
* **sampled\_expected\_count**: A tensor of type float. Same shape as sampled\_candidates. The expected counts under the sampling distribution of each of sampled\_candidates.

# tf.random.normal

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

Outputs random values from a normal distribution.

### Aliases:

* tf.compat.v1.random.normal
* tf.compat.v1.random\_normal
* tf.compat.v2.random.normal
* tf.random.normal

tf.random.normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    seed=None,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom training: basics](https://www.tensorflow.org/beta/tutorials/eager/custom_training)
* [Deep Convolutional Generative Adversarial Network](https://www.tensorflow.org/beta/tutorials/generative/dcgan)

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution.
* **dtype**: The type of the output.
* **seed**: A Python integer. Used to create a random seed for the distribution. See[tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random normal values.

# tf.random.poisson

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

Draws shape samples from each of the given Poisson distribution(s).

### Aliases:

* tf.compat.v2.random.poisson
* tf.random.poisson

tf.random.poisson(  
    shape,  
    lam,  
    dtype=tf.dtypes.float32,  
    seed=None,  
    name=None  
)

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

lam is the rate parameter describing the distribution(s).

#### Example:

samples = tf.random.poisson([10], [0.5, 1.5])  
# samples has shape [10, 2], where each slice [:, 0] and [:, 1] represents  
# the samples drawn from each distribution  
  
samples = tf.random.poisson([7, 5], [12.2, 3.3])  
# samples has shape [7, 5, 2], where each slice [:, :, 0] and [:, :, 1]  
# represents the 7x5 samples drawn from each of the two distributions

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output samples to be drawn per "rate"-parameterized distribution.
* **lam**: A Tensor or Python value or N-D array of type dtype. lam provides the rate parameter(s) describing the poisson distribution(s) to sample.
* **dtype**: The type of the output: float16, float32, float64, int32 or int64.
* **seed**: A Python integer. Used to create a random seed for the distributions. See[tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: Optional name for the operation.

#### Returns:

* **samples**: a Tensor of shape tf.concat([shape, tf.shape(lam)], axis=0) with values of type dtype.

# tf.random.set\_seed

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

Sets the graph-level random seed.

### Aliases:

* tf.compat.v2.random.set\_seed
* tf.random.set\_seed

tf.random.set\_seed(seed)

Defined in [python/framework/random\_seed.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/framework/random_seed.py).

Operations that rely on a random seed actually derive it from two seeds: the graph-level and operation-level seeds. This sets the graph-level seed.

Its interactions with operation-level seeds is as follows:

1. If neither the graph-level nor the operation seed is set: A random seed is used for this op.
2. If the graph-level seed is set, but the operation seed is not: The system deterministically picks an operation seed in conjunction with the graph-level seed so that it gets a unique random sequence.
3. If the graph-level seed is not set, but the operation seed is set: A default graph-level seed and the specified operation seed are used to determine the random sequence.
4. If both the graph-level and the operation seed are set: Both seeds are used in conjunction to determine the random sequence.

To illustrate the user-visible effects, consider these examples:

To generate different sequences across sessions, set neither graph-level nor op-level seeds:

a = tf.random.uniform([1])  
b = tf.random.normal([1])  
  
print("Session 1")  
with tf.compat.v1.Session() as sess1:  
  print(sess1.run(a))  # generates 'A1'  
  print(sess1.run(a))  # generates 'A2'  
  print(sess1.run(b))  # generates 'B1'  
  print(sess1.run(b))  # generates 'B2'  
  
print("Session 2")  
with tf.compat.v1.Session() as sess2:  
  print(sess2.run(a))  # generates 'A3'  
  print(sess2.run(a))  # generates 'A4'  
  print(sess2.run(b))  # generates 'B3'  
  print(sess2.run(b))  # generates 'B4'

To generate the same repeatable sequence for an op across sessions, set the seed for the op:

a = tf.random.uniform([1], seed=1)  
b = tf.random.normal([1])  
  
# Repeatedly running this block with the same graph will generate the same  
# sequence of values for 'a', but different sequences of values for 'b'.  
print("Session 1")  
with tf.compat.v1.Session() as sess1:  
  print(sess1.run(a))  # generates 'A1'  
  print(sess1.run(a))  # generates 'A2'  
  print(sess1.run(b))  # generates 'B1'  
  print(sess1.run(b))  # generates 'B2'  
  
print("Session 2")  
with tf.compat.v1.Session() as sess2:  
  print(sess2.run(a))  # generates 'A1'  
  print(sess2.run(a))  # generates 'A2'  
  print(sess2.run(b))  # generates 'B3'  
  print(sess2.run(b))  # generates 'B4'

To make the random sequences generated by all ops be repeatable across sessions, set a graph-level seed:

tf.random.set\_seed(1234)  
a = tf.random.uniform([1])  
b = tf.random.normal([1])  
  
# Repeatedly running this block with the same graph will generate the same  
# sequences of 'a' and 'b'.  
print("Session 1")  
with tf.compat.v1.Session() as sess1:  
  print(sess1.run(a))  # generates 'A1'  
  print(sess1.run(a))  # generates 'A2'  
  print(sess1.run(b))  # generates 'B1'  
  print(sess1.run(b))  # generates 'B2'  
  
print("Session 2")  
with tf.compat.v1.Session() as sess2:  
  print(sess2.run(a))  # generates 'A1'  
  print(sess2.run(a))  # generates 'A2'  
  print(sess2.run(b))  # generates 'B1'  
  print(sess2.run(b))  # generates 'B2'

#### Args:

* **seed**: integer.

# tf.random.shuffle

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

Randomly shuffles a tensor along its first dimension.

### Aliases:

* tf.compat.v1.random.shuffle
* tf.compat.v1.random\_shuffle
* tf.compat.v2.random.shuffle
* tf.random.shuffle

tf.random.shuffle(  
    value,  
    seed=None,  
    name=None  
)

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

The tensor is shuffled along dimension 0, such that each value[j] is mapped to one and only one output[i]. For example, a mapping that might occur for a 3x2 tensor is:

[[1, 2],       [[5, 6],  
 [3, 4],  ==>   [1, 2],  
 [5, 6]]        [3, 4]]

#### Args:

* **value**: A Tensor to be shuffled.
* **seed**: A Python integer. Used to create a random seed for the distribution. See[tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of same shape and type as value, shuffled along its first dimension.

# tf.random.stateless\_categorical

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

Draws deterministic pseudorandom samples from a categorical distribution.

### Aliases:

* tf.compat.v1.random.stateless\_categorical
* tf.compat.v2.random.stateless\_categorical
* tf.random.stateless\_categorical

tf.random.stateless\_categorical(  
    logits,  
    num\_samples,  
    seed,  
    dtype=tf.dtypes.int64,  
    name=None  
)

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

This is a stateless version of tf.categorical: if run twice with the same seeds, it will produce the same pseudorandom numbers. The output is consistent across multiple runs on the same hardware (and between CPU and GPU), but may change between versions of TensorFlow or on non-CPU/GPU hardware.

#### Example:

# samples has shape [1, 5], where each value is either 0 or 1 with equal  
# probability.  
samples = tf.random.stateless\_categorical(  
    tf.math.log([[10., 10.]]), 5, seed=[7, 17])

#### Args:

* **logits**: 2-D Tensor with shape [batch\_size, num\_classes]. Each slice [i, :] represents the unnormalized log-probabilities for all classes.
* **num\_samples**: 0-D. Number of independent samples to draw for each row slice.
* **seed**: A shape [2] integer Tensor of seeds to the random number generator.
* **dtype**: integer type to use for the output. Defaults to int64.
* **name**: Optional name for the operation.

#### Returns:

The drawn samples of shape [batch\_size, num\_samples].

# tf.random.stateless\_normal

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

Outputs deterministic pseudorandom values from a normal distribution.

### Aliases:

* tf.compat.v1.random.stateless\_normal
* tf.compat.v2.random.stateless\_normal
* tf.random.stateless\_normal

tf.random.stateless\_normal(  
    shape,  
    seed,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

This is a stateless version of [tf.random.normal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/normal): if run twice with the same seeds, it will produce the same pseudorandom numbers. The output is consistent across multiple runs on the same hardware (and between CPU and GPU), but may change between versions of TensorFlow or on non-CPU/GPU hardware.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **seed**: A shape [2] integer Tensor of seeds to the random number generator.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random normal values.

# tf.random.stateless\_truncated\_normal

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

Outputs deterministic pseudorandom values, truncated normally distributed.

### Aliases:

* tf.compat.v1.random.stateless\_truncated\_normal
* tf.compat.v2.random.stateless\_truncated\_normal
* tf.random.stateless\_truncated\_normal

tf.random.stateless\_truncated\_normal(  
    shape,  
    seed,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

This is a stateless version of [tf.random.truncated\_normal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal): if run twice with the same seeds, it will produce the same pseudorandom numbers. The output is consistent across multiple runs on the same hardware (and between CPU and GPU), but may change between versions of TensorFlow or on non-CPU/GPU hardware.

The generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **seed**: A shape [2] integer Tensor of seeds to the random number generator.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the truncated normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution, before truncation.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random truncated normal values.

# tf.random.stateless\_truncated\_normal

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

Outputs deterministic pseudorandom values, truncated normally distributed.

### Aliases:

* tf.compat.v1.random.stateless\_truncated\_normal
* tf.compat.v2.random.stateless\_truncated\_normal
* tf.random.stateless\_truncated\_normal

tf.random.stateless\_truncated\_normal(  
    shape,  
    seed,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

This is a stateless version of [tf.random.truncated\_normal](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal): if run twice with the same seeds, it will produce the same pseudorandom numbers. The output is consistent across multiple runs on the same hardware (and between CPU and GPU), but may change between versions of TensorFlow or on non-CPU/GPU hardware.

The generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **seed**: A shape [2] integer Tensor of seeds to the random number generator.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the truncated normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution, before truncation.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random truncated normal values.

# tf.random.truncated\_normal

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/truncated_normal#used_in_the_guide)

Outputs random values from a truncated normal distribution.

### Aliases:

* tf.compat.v1.random.truncated\_normal
* tf.compat.v1.truncated\_normal
* tf.compat.v2.random.truncated\_normal
* tf.random.truncated\_normal

tf.random.truncated\_normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    seed=None,  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

The generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the truncated normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution, before truncation.
* **dtype**: The type of the output.
* **seed**: A Python integer. Used to create a random seed for the distribution. See[tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random truncated normal values

.

# tf.random.uniform

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

Outputs random values from a uniform distribution.

### Aliases:

* tf.compat.v1.random.uniform
* tf.compat.v1.random\_uniform
* tf.compat.v2.random.uniform
* tf.random.uniform

tf.random.uniform(  
    shape,  
    minval=0,  
    maxval=None,  
    dtype=tf.dtypes.float32,  
    seed=None,  
    name=None  
)

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

### Used in the guide:

* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

The generated values follow a uniform distribution in the range [minval, maxval). The lower bound minval is included in the range, while the upper bound maxval is excluded.

For floats, the default range is [0, 1). For ints, at least maxval must be specified explicitly.

In the integer case, the random integers are slightly biased unless maxval - minval is an exact power of two. The bias is small for values of maxval - minval significantly smaller than the range of the output (either 2\*\*32 or 2\*\*64).

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **minval**: A 0-D Tensor or Python value of type dtype. The lower bound on the range of random values to generate. Defaults to 0.
* **maxval**: A 0-D Tensor or Python value of type dtype. The upper bound on the range of random values to generate. Defaults to 1 if dtype is floating point.
* **dtype**: The type of the output: float16, float32, float64, int32, or int64.
* **seed**: A Python integer. Used to create a random seed for the distribution. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed) for behavior.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random uniform values.

#### Raises:

* **ValueError**: If dtype is integral and maxval is not specified.

# tf.random.uniform\_candidate\_sampler

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

Samples a set of classes using a uniform base distribution.

### Aliases:

* tf.compat.v1.nn.uniform\_candidate\_sampler
* tf.compat.v1.random.uniform\_candidate\_sampler
* tf.compat.v2.random.uniform\_candidate\_sampler
* tf.random.uniform\_candidate\_sampler

tf.random.uniform\_candidate\_sampler(  
    true\_classes,  
    num\_true,  
    num\_sampled,  
    unique,  
    range\_max,  
    seed=None,  
    name=None  
)

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

This operation randomly samples a tensor of sampled classes (sampled\_candidates) from the range of integers [0, range\_max).

The elements of sampled\_candidates are drawn without replacement (if unique=True) or with replacement (if unique=False) from the base distribution.

The base distribution for this operation is the uniform distribution over the range of integers [0, range\_max).

In addition, this operation returns tensors true\_expected\_count and sampled\_expected\_countrepresenting the number of times each of the target classes (true\_classes) and the sampled classes (sampled\_candidates) is expected to occur in an average tensor of sampled classes. These values correspond to Q(y|x) defined in [this document](http://www.tensorflow.org/extras/candidate_sampling.pdf). If unique=True, then these are post-rejection probabilities and we compute them approximately.

#### Args:

* **true\_classes**: A Tensor of type int64 and shape [batch\_size, num\_true]. The target classes.
* **num\_true**: An int. The number of target classes per training example.
* **num\_sampled**: An int. The number of classes to randomly sample. The sampled\_candidatesreturn value will have shape [num\_sampled]. If unique=True, num\_sampled must be less than or equal to range\_max.
* **unique**: A bool. Determines whether all sampled classes in a batch are unique.
* **range\_max**: An int. The number of possible classes.
* **seed**: An int. An operation-specific seed. Default is 0.
* **name**: A name for the operation (optional).

#### Returns:

* **sampled\_candidates**: A tensor of type int64 and shape [num\_sampled]. The sampled classes, either with possible duplicates (unique=False) or all unique (unique=True). In either case, sampled\_candidates is independent of the true classes.
* **true\_expected\_count**: A tensor of type float. Same shape as true\_classes. The expected counts under the sampling distribution of each of true\_classes.
* **sampled\_expected\_count**: A tensor of type float. Same shape as sampled\_candidates. The expected counts under the sampling distribution of each of sampled\_candidates.

Module: tf.random.experimental

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

Public API for tf.random.experimental namespace.

Classes

[class Generator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator): Random-number generator.

Functions

[create\_rng\_state(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/create_rng_state): Creates a RNG state.

[get\_global\_generator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/get_global_generator)

[set\_global\_generator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/set_global_generator): Replaces the global generator with another Generator object.

# tf.random.experimental.create\_rng\_state

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

Creates a RNG state.

### Aliases:

* tf.compat.v1.random.experimental.create\_rng\_state
* tf.compat.v2.random.experimental.create\_rng\_state
* tf.random.experimental.create\_rng\_state

tf.random.experimental.create\_rng\_state(  
    seed,  
    algorithm  
)

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

#### Args:

* **seed**: an integer or 1-D tensor.
* **algorithm**: an integer representing the RNG algorithm.

#### Returns:

a 1-D tensor whose size depends on the algorithm.

# tf.random.experimental.Generator

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator#top_of_page)
* [Class Generator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator#class_generator)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/random/experimental/Generator#properties)

## Class Generator

Random-number generator.

### Aliases:

* Class tf.compat.v1.random.experimental.Generator
* Class tf.compat.v2.random.experimental.Generator
* Class tf.random.experimental.Generator

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

It uses Variable to manage its internal state, and allows choosing an Random-Number-Generation (RNG) algorithm.

CPU, GPU and TPU with the same algorithm and seed will generate the same integer random numbers. Float-point results (such as the output of normal) may have small numerical discrepancies between CPU and GPU.

## \_\_init\_\_

\_\_init\_\_(  
    copy\_from=None,  
    state=None,  
    alg=None  
)

Creates a generator.

The new generator will be initialized by one of the following ways, with decreasing precedence: (1) If copy\_from is not None, the new generator is initialized by copying information from another generator. (3) If state and alg are not None (they must be set together), the new generator is initialized by a state.

#### Args:

* **copy\_from**: a generator to be copied from.
* **state**: a vector of dtype STATE\_TYPE representing the initial state of the RNG, whose length and semantics are algorithm-specific.
* **alg**: the RNG algorithm. Possible values are RNG\_ALG\_PHILOX for the Philox algorithm and RNG\_ALG\_THREEFRY for the ThreeFry algorithm (see paper 'Parallel Random Numbers: As Easy as 1, 2, 3' [https://www.thesalmons.org/john/random123/papers/random123sc11.pdf]).

## Properties

### algorithm

The RNG algorithm.

### key

The 'key' part of the state of a counter-based RNG.

For a counter-base RNG algorithm such as Philox and ThreeFry (as described in paper 'Parallel Random Numbers: As Easy as 1, 2, 3' [https://www.thesalmons.org/john/random123/papers/random123sc11.pdf]), the RNG state consists of two parts: counter and key. The output is generated via the formula: output=hash(key, counter), i.e. a hashing of the counter parametrized by the key. Two RNGs with two different keys can be thought as generating two independent random-number streams (a stream is formed by increasing the counter).

#### Returns:

A scalar which is the 'key' part of the state, if the RNG algorithm is counter-based; otherwise it raises a ValueError.

### state

The internal state of the RNG.

## Methods

### binomial

binomial(  
    shape,  
    counts,  
    probs,  
    dtype=tf.dtypes.int32,  
    name=None  
)

Outputs random values from a binomial distribution.

The generated values follow a binomial distribution with specified count and probability of success parameters.

#### Example:

counts = [10., 20.]  
# Probability of success.  
probs = [0.8, 0.9]  
  
rng = tf.random.experimental.Generator(seed=234)  
binomial\_samples = rng.binomial(shape=[2], counts=counts, probs=probs)

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **counts**: A 0/1-D Tensor or Python value`. The counts of the binomial distribution.
* **probs**: A 0/1-D Tensor or Python value`. The probability of success for the binomial distribution.
* **dtype**: The type of the output. Default: tf.int32
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random binomial values.

### from\_key\_counter

@classmethod  
from\_key\_counter(  
    cls,  
    key,  
    counter,  
    alg  
)

Creates a generator from a key and a counter.

This constructor only applies if the algorithm is a counter-based algorithm. See method key for the meaning of "key" and "counter".

#### Args:

* **key**: the key for the RNG, a scalar of type STATE\_TYPE.
* **counter**: a vector of dtype STATE\_TYPE representing the initial counter for the RNG, whose length is algorithm-specific.,
* **alg**: the RNG algorithm. If None, it will be auto-selected. See \_\_init\_\_ for its possible values.

#### Returns:

The new generator.

### from\_non\_deterministic\_state

@classmethod  
from\_non\_deterministic\_state(  
    cls,  
    alg=None  
)

Creates a generator by non-deterministically initializing its state.

The source of the non-determinism will be platform- and time-dependent.

#### Args:

* **alg**: (optional) the RNG algorithm. If None, it will be auto-selected. See \_\_init\_\_ for its possible values.

#### Returns:

The new generator.

### from\_seed

@classmethod  
from\_seed(  
    cls,  
    seed,  
    alg=None  
)

Creates a generator from a seed.

A seed is a 1024-bit unsigned integer represented either as a Python integer or a vector of integers. Seeds shorter than 1024-bit will be padded. The padding, the internal structure of a seed and the way a seed is converted to a state are all opaque (unspecified). The only semantics specification of seeds is that two different seeds are likely to produce two independent generators (but no guarantee).

#### Args:

* **seed**: the seed for the RNG.
* **alg**: (optional) the RNG algorithm. If None, it will be auto-selected. See \_\_init\_\_ for its possible values.

#### Returns:

The new generator.

### from\_state

@classmethod  
from\_state(  
    cls,  
    state,  
    alg  
)

Creates a generator from a state.

See \_\_init\_\_ for description of state and alg.

#### Args:

* **state**: the new state.
* **alg**: the RNG algorithm.

#### Returns:

The new generator.

### make\_seeds

make\_seeds(count=1)

Generates seeds for stateless random ops.

#### For example:

seeds = get\_global\_generator().make\_seeds(count=10)  
for i in range(10):  
  seed = seeds[:, i]  
  numbers = stateless\_random\_normal(shape=[2, 3], seed=seed)  
  ...

#### Args:

* **count**: the number of seed pairs (note that stateless random ops need a pair of seeds to invoke).

#### Returns:

A tensor of shape [2, count] and dtype int64.

### normal

normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

Outputs random values from a normal distribution.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random normal values.

### reset

reset(state)

Resets the generator by a new state.

See \_\_init\_\_ for the meaning of "state".

#### Args:

* **state**: the new state.

### reset\_from\_key\_counter

reset\_from\_key\_counter(  
    key,  
    counter  
)

Resets the generator by a new key-counter pair.

See from\_key\_counter for the meaning of "key" and "counter".

#### Args:

* **key**: the new key.
* **counter**: the new counter.

### reset\_from\_seed

reset\_from\_seed(seed)

Resets the generator by a new seed.

See from\_seed for the meaning of "seed".

#### Args:

* **seed**: the new seed.

### skip

skip(delta)

Advance the counter of a counter-based RNG.

#### Args:

* **delta**: the amount of advancement. The state of the RNG after skip(n) will be the same as that after normal([n]) (or any other distribution). The actual increment added to the counter is an unspecified implementation detail.

### split

split(count=1)

Returns a list of independent Generator objects.

Two generators are independent of each other in the sense that the random-number streams they generate don't have statistically detectable correlations. The new generators are also independent of the old one. The old generator's state will be changed (like other random-number generating methods), so two calls of split will return different new generators.

#### For example:

gens = get\_global\_generator().split(count=10)  
for gen in gens:  
  numbers = gen.normal(shape=[2, 3])  
  # ...  
gens2 = get\_global\_generator().split(count=10)  
# gens2 will be different from gens

The new generators will be put on the current device (possible different from the old generator's), for example:

with tf.device("/device:CPU:0"):  
  gen = Generator(seed=1234)  # gen is on CPU  
with tf.device("/device:GPU:0"):  
  gens = gen.split(count=10)  # gens are on GPU

#### Args:

* **count**: the number of generators to return.

#### Returns:

A list (length count) of Generator objects independent of each other. The new generators have the same RNG algorithm as the old one.

### truncated\_normal

truncated\_normal(  
    shape,  
    mean=0.0,  
    stddev=1.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

Outputs random values from a truncated normal distribution.

The generated values follow a normal distribution with specified mean and standard deviation, except that values whose magnitude is more than 2 standard deviations from the mean are dropped and re-picked.

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **mean**: A 0-D Tensor or Python value of type dtype. The mean of the truncated normal distribution.
* **stddev**: A 0-D Tensor or Python value of type dtype. The standard deviation of the normal distribution, before truncation.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random truncated normal values.

### uniform

uniform(  
    shape,  
    minval=0,  
    maxval=None,  
    dtype=tf.dtypes.float32,  
    name=None  
)

Outputs random values from a uniform distribution.

The generated values follow a uniform distribution in the range [minval, maxval). The lower bound minval is included in the range, while the upper bound maxval is excluded. (For float numbers especially low-precision types like bfloat16, because of rounding, the result may sometimes include maxval.)

For floats, the default range is [0, 1). For ints, at least maxval must be specified explicitly.

In the integer case, the random integers are slightly biased unless maxval - minval is an exact power of two. The bias is small for values of maxval - minval significantly smaller than the range of the output (either 2\*\*32 or 2\*\*64).

#### Args:

* **shape**: A 1-D integer Tensor or Python array. The shape of the output tensor.
* **minval**: A 0-D Tensor or Python value of type dtype. The lower bound on the range of random values to generate. Defaults to 0.
* **maxval**: A 0-D Tensor or Python value of type dtype. The upper bound on the range of random values to generate. Defaults to 1 if dtype is floating point.
* **dtype**: The type of the output.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the specified shape filled with random uniform values.

#### Raises:

* **ValueError**: If dtype is integral and maxval is not specified.

### uniform\_full\_int

uniform\_full\_int(  
    shape,  
    dtype=tf.dtypes.uint64,  
    name=None  
)

Uniform distribution on an integer type's entire range.

The other method uniform only covers the range [minval, maxval), which cannot be dtype's full range because maxval is of type dtype.

#### Args:

* **shape**: the shape of the output.
* **dtype**: (optional) the integer type, default to uint64.
* **name**: (optional) the name of the node.

#### Returns:

A tensor of random numbers of the required shape.

# tf.random.experimental.get\_global\_generator

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

### Aliases:

* tf.compat.v1.random.experimental.get\_global\_generator
* tf.compat.v2.random.experimental.get\_global\_generator
* tf.random.experimental.get\_global\_generator

tf.random.experimental.get\_global\_generator()

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

# tf.random.experimental.set\_global\_generator

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

Replaces the global generator with another Generator object.

### Aliases:

* tf.compat.v1.random.experimental.set\_global\_generator
* tf.compat.v2.random.experimental.set\_global\_generator
* tf.random.experimental.set\_global\_generator

tf.random.experimental.set\_global\_generator(generator)

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

This function creates a new Generator object (and the Variable object within), which does not work well with tf.function because (1) tf.function puts restrictions on Variable creation thus reset\_global\_generator can't be freely used inside tf.function; (2) redirecting a global variable to a new object is problematic with tf.function because the old object may be captured by a 'tf.function'ed function and still be used by it. A 'tf.function'ed function only keeps weak references to variables, so deleting a variable and then calling that function again may raise an error, as demonstrated by random\_test.py/RandomTest.testResetGlobalGeneratorBadWithDefun .

#### Args:

* **generator**: the new Generator object.

Module: tf.compat.v1.saved\_model / tf.saved\_model

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

Public API for tf.saved\_model namespace.

Modules

[builder](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/builder) module: SavedModel builder.

[constants](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/constants) module: Constants for SavedModel save and restore operations.

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

[loader](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/loader) module: Loader functionality for SavedModel with hermetic, language-neutral exports.

[main\_op](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/main_op) module: SavedModel main op.

[signature\_constants](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/signature_constants) module: Signature constants for SavedModel save and restore operations.

[signature\_def\_utils](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/signature_def_utils) module: SignatureDef utility functions.

[tag\_constants](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/tag_constants) module: Common tags used for graphs in SavedModel.

[utils](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/utils) module: SavedModel utility functions.

Classes

[class Builder](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder): Builds the SavedModel protocol buffer and saves variables and assets.

Functions

[build\_signature\_def(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/build_signature_def): Utility function to build a SignatureDef protocol buffer.

[build\_tensor\_info(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/build_tensor_info): Utility function to build TensorInfo proto from a Tensor. (deprecated)

[classification\_signature\_def(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/classification_signature_def): Creates classification signature from given examples and predictions.

[contains\_saved\_model(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/contains_saved_model): Checks whether the provided export directory could contain a SavedModel.

[get\_tensor\_from\_tensor\_info(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/get_tensor_from_tensor_info): Returns the Tensor or SparseTensor described by a TensorInfo proto. (deprecated)

[is\_valid\_signature(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/is_valid_signature): Determine whether a SignatureDef can be served by TensorFlow Serving.

[load(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/load): Loads the model from a SavedModel as specified by tags. (deprecated)

[load\_v2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load): Load a SavedModel from export\_dir.

[main\_op\_with\_restore(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/main_op_with_restore): Returns a main op to init variables, tables and restore the graph. (deprecated)

[maybe\_saved\_model\_directory(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/contains_saved_model): Checks whether the provided export directory could contain a SavedModel.

[predict\_signature\_def(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/predict_signature_def): Creates prediction signature from given inputs and outputs.

[regression\_signature\_def(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/regression_signature_def): Creates regression signature from given examples and predictions.

[save(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save): Exports the Trackable object obj to [SavedModel format](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md).

[simple\_save(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/simple_save): Convenience function to build a SavedModel suitable for serving. (deprecated)

Other Members

* ASSETS\_DIRECTORY = 'assets'
* ASSETS\_KEY = 'saved\_model\_assets'
* CLASSIFY\_INPUTS = 'inputs'
* CLASSIFY\_METHOD\_NAME = 'tensorflow/serving/classify'
* CLASSIFY\_OUTPUT\_CLASSES = 'classes'
* CLASSIFY\_OUTPUT\_SCORES = 'scores'
* DEFAULT\_SERVING\_SIGNATURE\_DEF\_KEY = 'serving\_default'
* GPU = 'gpu'
* LEGACY\_INIT\_OP\_KEY = 'legacy\_init\_op'
* MAIN\_OP\_KEY = 'saved\_model\_main\_op'
* PREDICT\_INPUTS = 'inputs'
* PREDICT\_METHOD\_NAME = 'tensorflow/serving/predict'
* PREDICT\_OUTPUTS = 'outputs'
* REGRESS\_INPUTS = 'inputs'
* REGRESS\_METHOD\_NAME = 'tensorflow/serving/regress'
* REGRESS\_OUTPUTS = 'outputs'
* SAVED\_MODEL\_FILENAME\_PB = 'saved\_model.pb'
* SAVED\_MODEL\_FILENAME\_PBTXT = 'saved\_model.pbtxt'
* SAVED\_MODEL\_SCHEMA\_VERSION = 1
* SERVING = 'serve'
* TPU = 'tpu'
* TRAINING = 'train'
* VARIABLES\_DIRECTORY = 'variables'
* VARIABLES\_FILENAME = 'variables'

# tf.saved\_model.contains\_saved\_model

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

Checks whether the provided export directory could contain a SavedModel.

### Aliases:

* tf.compat.v2.saved\_model.contains\_saved\_model
* tf.saved\_model.contains\_saved\_model

tf.saved\_model.contains\_saved\_model(export\_dir)

Defined in [python/saved\_model/loader\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/loader_impl.py).

Note that the method does not load any data by itself. If the method returns false, the export directory definitely does not contain a SavedModel. If the method returns true, the export directory may contain a SavedModel but provides no guarantee that it can be loaded.

#### Args:

* **export\_dir**: Absolute string path to possible export location. For example, '/my/foo/model'.

#### Returns:

True if the export directory contains SavedModel files, False otherwise.

# tf.saved\_model.load

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load#used_in_the_guide)

Load a SavedModel from export\_dir.

### Aliases:

* tf.compat.v1.saved\_model.load\_v2
* tf.compat.v2.saved\_model.load
* tf.saved\_model.load

tf.saved\_model.load(  
    export\_dir,  
    tags=None  
)

Defined in [python/saved\_model/load.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/load.py).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

Signatures associated with the SavedModel are available as functions:

imported = tf.saved\_model.load(path)  
f = imported.signatures["serving\_default"]  
print(f(x=tf.constant([[1.]])))

Objects exported with [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save) additionally have trackable objects and functions assigned to attributes:

exported = tf.train.Checkpoint(v=tf.Variable(3.))  
exported.f = tf.function(  
    lambda x: exported.v \* x,  
    input\_signature=[tf.TensorSpec(shape=None, dtype=tf.float32)])  
tf.saved\_model.save(exported, path)  
imported = tf.saved\_model.load(path)  
assert 3. == imported.v.numpy()  
assert 6. == imported.f(x=tf.constant(2.)).numpy()

Loading Keras models

Keras models are trackable, so they can be saved to SavedModel. The object returned by [tf.saved\_model.load](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load) is not a Keras object (i.e. doesn't have .fit, .predict, etc. methods). A few attributes and functions are still available: .variables, .trainable\_variables and .\_\_call\_\_.

model = tf.keras.Model(...)  
tf.saved\_model.save(model, path)  
imported = tf.saved\_model.load(path)  
outputs = imported(inputs)

Use [tf.keras.models.load\_model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/models/load_model) to restore the Keras model.

Importing SavedModels from TensorFlow 1.x

SavedModels from [tf.estimator.Estimator](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/estimator/Estimator) or 1.x SavedModel APIs have a flat graph instead of [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) objects. These SavedModels will have functions corresponding to their signatures in the .signatures attribute, but also have a .prune method which allows you to extract functions for new subgraphs. This is equivalent to importing the SavedModel and naming feeds and fetches in a Session from TensorFlow 1.x.

imported = tf.saved\_model.load(path\_to\_v1\_saved\_model)  
pruned = imported.prune("x:0", "out:0")  
pruned(tf.ones([]))

See [tf.compat.v1.wrap\_function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/wrap_function) for details. These SavedModels also have a .variablesattribute containing imported variables, and a .graph attribute representing the whole imported graph. For SavedModels exported from [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save), variables are instead assigned to whichever attributes they were assigned before export.

#### Args:

* **export\_dir**: The SavedModel directory to load from.
* **tags**: A tag or sequence of tags identifying the MetaGraph to load. Optional if the SavedModel contains a single MetaGraph, as for those exported from [tf.saved\_model.load](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load).

#### Returns:

A trackable object with a signatures attribute mapping from signature keys to functions. If the SavedModel was exported by [tf.saved\_model.load](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load), it also points to trackable objects and functions which were attached to the exported object.

#### Raises:

* **ValueError**: If tags don't match a MetaGraph in the SavedModel.

# tf.saved\_model.save

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save#used_in_the_guide)

Exports the Trackable object obj to [SavedModel format](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md).

### Aliases:

* tf.compat.v1.saved\_model.experimental.save
* tf.compat.v1.saved\_model.save
* tf.compat.v2.saved\_model.save
* tf.saved\_model.save

tf.saved\_model.save(  
    obj,  
    export\_dir,  
    signatures=None  
)

Defined in [python/saved\_model/save.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/save.py).

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

#### Example usage:

class Adder(tf.Module):  
  
  @tf.function(input\_signature=[tf.TensorSpec(shape=None, dtype=tf.float32)])  
  def add(self, x):  
    return x + x + 1.  
  
to\_export = Adder()  
tf.saved\_model.save(to\_export, '/tmp/adder')

The resulting SavedModel is then servable with an input named "x", its value having any shape and dtype float32.

The optional signatures argument controls which methods in obj will be available to programs which consume SavedModels, for example serving APIs. Python functions may be decorated with@tf.function(input\_signature=...) and passed as signatures directly, or lazily with a call to get\_concrete\_function on the method decorated with @tf.function.

If the signatures argument is omitted, obj will be searched for @tf.function-decorated methods. If exactly one @tf.function is found, that method will be used as the default signature for the SavedModel. This behavior is expected to change in the future, when a corresponding[tf.saved\_model.load](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load) symbol is added. At that point signatures will be completely optional, and any @tf.function attached to obj or its dependencies will be exported for use with load.

When invoking a signature in an exported SavedModel, Tensor arguments are identified by name. These names will come from the Python function's argument names by default. They may be overridden by specifying a name=... argument in the corresponding [tf.TensorSpec](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/TensorSpec) object. Explicit naming is required if multiple Tensors are passed through a single argument to the Python function.

The outputs of functions used as signatures must either be flat lists, in which case outputs will be numbered, or a dictionary mapping string keys to Tensor, in which case the keys will be used to name outputs.

Signatures are available in objects returned by [tf.saved\_model.load](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/load) as a .signatures attribute. This is a reserved attribute: [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save) on an object with a custom .signaturesattribute will raise an exception.

Since [tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) objects are also Trackable, this function can be used to export Keras models. For example, exporting with a signature specified:

class Model(tf.keras.Model):  
  
  @tf.function(input\_signature=[tf.TensorSpec(shape=[None], dtype=tf.string)])  
  def serve(self, serialized):  
    ...  
  
m = Model()  
tf.saved\_model.save(m, '/tmp/saved\_model/')

Exporting from a function without a fixed signature:

class Model(tf.keras.Model):  
  
  @tf.function  
  def call(self, x):  
    ...  
  
m = Model()  
tf.saved\_model.save(  
    m, '/tmp/saved\_model/',  
    signatures=m.call.get\_concrete\_function(  
        tf.TensorSpec(shape=[None, 3], dtype=tf.float32, name="inp")))

[tf.keras.Model](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/Model) instances constructed from inputs and outputs already have a signature and so do not require a @tf.function decorator or a signatures argument. If neither are specified, the model's forward pass is exported.

x = input\_layer.Input((4,), name="x")  
y = core.Dense(5, name="out")(x)  
model = training.Model(x, y)  
tf.saved\_model.save(model, '/tmp/saved\_model/')  
# The exported SavedModel takes "x" with shape [None, 4] and returns "out"  
# with shape [None, 5]

Variables must be tracked by assigning them to an attribute of a tracked object or to an attribute of obj directly. TensorFlow objects (e.g. layers from [tf.keras.layers](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/keras/layers), optimizers from [tf.train](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train)) track their variables automatically. This is the same tracking scheme that [tf.train.Checkpoint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint)uses, and an exported Checkpoint object may be restored as a training checkpoint by pointing [tf.train.Checkpoint.restore](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/train/Checkpoint#restore) to the SavedModel's "variables/" subdirectory. Currently variables are the only stateful objects supported by [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save), but others (e.g. tables) will be supported in the future.

[tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) does not hard-code device annotations from outside the function body, instead using the calling context's device. This means for example that exporting a model which runs on a GPU and serving it on a CPU will generally work, with some exceptions. [tf.device](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/device) annotations inside the body of the function will be hard-coded in the exported model; this type of annotation is discouraged. Device-specific operations, e.g. with "cuDNN" in the name or with device-specific layouts, may cause issues. Currently a DistributionStrategy is another exception: active distribution strategies will cause device placements to be hard-coded in a function. Exporting a single-device computation and importing under a DistributionStrategy is not currently supported, but may be in the future.

SavedModels exported with [tf.saved\_model.save](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/saved_model/save) [strip default-valued attributes](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md#stripping-default-valued-attributes) automatically, which removes one source of incompatibilities when the consumer of a SavedModel is running an older TensorFlow version than the producer. There are however other sources of incompatibilities which are not handled automatically, such as when the exported model contains operations which the consumer does not have definitions for.

#### Args:

* **obj**: A trackable object to export.
* **export\_dir**: A directory in which to write the SavedModel.
* **signatures**: Optional, either a [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) with an input signature specified or the result of f.get\_concrete\_function on a @tf.function-decorated function f, in which case f will be used to generate a signature for the SavedModel under the default serving signature key. signatures may also be a dictionary, in which case it maps from signature keys to either [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) instances with input signatures or concrete functions. The keys of such a dictionary may be arbitrary strings, but will typically be from thetf.saved\_model.signature\_constants module.

#### Raises:

* **ValueError**: If obj is not trackable.

#### Eager Compatibility

Not well supported when graph building. From TensorFlow 1.x,tf.compat.v1.enable\_eager\_execution() should run first. Calling tf.saved\_model.save in a loop when graph building from TensorFlow 1.x will add new save operations to the default graph each iteration.

May not be called from within a function body.

# tf.compat.v1.saved\_model.Builder

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder#top_of_page)
* [Class Builder](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder#class_builder)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/saved_model/Builder#methods)

## Class Builder

Builds the SavedModel protocol buffer and saves variables and assets.

### Aliases:

* Class tf.compat.v1.saved\_model.Builder
* Class tf.compat.v1.saved\_model.builder.SavedModelBuilder

Defined in [python/saved\_model/builder\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/builder_impl.py).

The SavedModelBuilder class provides functionality to build a SavedModel protocol buffer. Specifically, this allows multiple meta graphs to be saved as part of a single language-neutral SavedModel, while sharing variables and assets.

To build a SavedModel, the first meta graph must be saved with variables. Subsequent meta graphs will simply be saved with their graph definitions. If assets need to be saved and written or copied to disk, they can be provided when the meta graph def is added. If multiple meta graph defs are associated an asset of the same name, only the first version is retained.

Each meta graph added to the SavedModel must be annotated with tags. The tags provide a means to identify the specific meta graph to load and restore, along with the shared set of variables and assets.

Typical usage for the SavedModelBuilder:

...  
builder = tf.compat.v1.saved\_model.Builder(export\_dir)  
  
with tf.compat.v1.Session(graph=tf.Graph()) as sess:  
  ...  
  builder.add\_meta\_graph\_and\_variables(sess,  
                                  ["foo-tag"],  
                                  signature\_def\_map=foo\_signatures,  
                                  assets\_collection=foo\_assets)  
...  
  
with tf.compat.v1.Session(graph=tf.Graph()) as sess:  
  ...  
  builder.add\_meta\_graph(["bar-tag", "baz-tag"])  
...  
  
builder.save()

**Note:** This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.builder.SavedModelBuilder or tf.compat.v1.saved\_model.Builder. Tensorflow 2.0 will introduce a new object-based method of creating SavedModels.

## \_\_init\_\_

\_\_init\_\_(export\_dir)

## Methods

### add\_meta\_graph

add\_meta\_graph(  
    tags,  
    signature\_def\_map=None,  
    assets\_collection=None,  
    legacy\_init\_op=None,  
    clear\_devices=False,  
    main\_op=None,  
    strip\_default\_attrs=False,  
    saver=None  
)

Adds the current meta graph to the SavedModel.

Creates a Saver in the current scope and uses the Saver to export the meta graph def. Invoking this API requires the add\_meta\_graph\_and\_variables() API to have been invoked before.

#### Args:

* **tags**: The set of tags to annotate the meta graph def with.
* **signature\_def\_map**: The map of signature defs to be added to the meta graph def.
* **assets\_collection**: Assets to be saved with SavedModel. Note that this list should be a subset of the assets saved as part of the first meta graph in the SavedModel.
* **clear\_devices**: Set to true if the device info on the default graph should be cleared.
* **init\_op**: Op or group of ops to execute when the graph is loaded. Note that when the init\_op is specified it is run after the restore op at load-time.
* **train\_op**: Op or group of opts that trains the model when run. This will not be run automatically when the graph is loaded, instead saved in a SignatureDef accessible through the exported MetaGraph.
* **saver**: An instance of tf.compat.v1.train.Saver that will be used to export the metagraph. If None, a sharded Saver that restores all variables will be used.

#### Raises:

* **AssertionError**: If the variables for the SavedModel have not been saved yet, or if the graph already contains one or more legacy init ops.

### add\_meta\_graph\_and\_variables

add\_meta\_graph\_and\_variables(  
    sess,  
    tags,  
    signature\_def\_map=None,  
    assets\_collection=None,  
    legacy\_init\_op=None,  
    clear\_devices=False,  
    main\_op=None,  
    strip\_default\_attrs=False,  
    saver=None  
)

Adds the current meta graph to the SavedModel and saves variables.

Creates a Saver to save the variables from the provided session. Exports the corresponding meta graph def. This function assumes that the variables to be saved have been initialized. For a given SavedModelBuilder, this API must be called exactly once and for the first meta graph to save. For subsequent meta graph defs to be added, the add\_meta\_graph() API must be used.

#### Args:

* **sess**: The TensorFlow session from which to save the meta graph and variables.
* **tags**: The set of tags with which to save the meta graph.
* **signature\_def\_map**: The map of signature def map to add to the meta graph def.
* **assets\_collection**: Assets to be saved with SavedModel.
* **clear\_devices**: Set to true if the device info on the default graph should be cleared.
* **init\_op**: Op or group of ops to execute when the graph is loaded. Note that when the init\_op is specified it is run after the restore op at load-time.
* **train\_op**: Op or group of ops that trains the model when run. This will not be run automatically when the graph is loaded, instead saved in a SignatureDef accessible through the exported MetaGraph.
* **strip\_default\_attrs**: Boolean. If True, default-valued attributes will be removed from the NodeDefs. For a detailed guide, see [Stripping Default-Valued Attributes](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md#stripping-default-valued-attributes).
* **saver**: An instance of tf.compat.v1.train.Saver that will be used to export the metagraph and save variables. If None, a sharded Saver that restores all variables will be used.

### save

save(as\_text=False)

Writes a SavedModel protocol buffer to disk.

The function writes the SavedModel protocol buffer to the export directory in serialized format.

#### Args:

* **as\_text**: Writes the SavedModel protocol buffer in text format to disk. Protocol buffers in text format are useful for debugging, but parsing fails when it encounters an unknown field and so is not forward compatible. This means changes to TensorFlow may prevent deployment of new text format SavedModels to existing serving binaries. Do not deploy as\_text SavedModels to production.
* Returns:

# The path to which the SavedModel protocol buffer was written.

# tf.compat.v1.saved\_model.build\_signature\_def

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

Utility function to build a SignatureDef protocol buffer.

### Aliases:

* tf.compat.v1.saved\_model.build\_signature\_def
* tf.compat.v1.saved\_model.signature\_def\_utils.build\_signature\_def

tf.compat.v1.saved\_model.build\_signature\_def(  
    inputs=None,  
    outputs=None,  
    method\_name=None  
)

Defined in [python/saved\_model/signature\_def\_utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/signature_def_utils_impl.py).

#### Args:

* **inputs**: Inputs of the SignatureDef defined as a proto map of string to tensor info.
* **outputs**: Outputs of the SignatureDef defined as a proto map of string to tensor info.
* **method\_name**: Method name of the SignatureDef as a string.

#### Returns:

A SignatureDef protocol buffer constructed based on the supplied arguments.

# tf.compat.v1.saved\_model.build\_tensor\_info

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

Utility function to build TensorInfo proto from a Tensor. (deprecated)

### Aliases:

* tf.compat.v1.saved\_model.build\_tensor\_info
* tf.compat.v1.saved\_model.utils.build\_tensor\_info

tf.compat.v1.saved\_model.build\_tensor\_info(tensor)

Defined in [python/saved\_model/utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/utils_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.utils.build\_tensor\_info or tf.compat.v1.saved\_model.build\_tensor\_info.

#### Args:

* **tensor**: Tensor or SparseTensor whose name, dtype and shape are used to build the TensorInfo. For SparseTensors, the names of the three constituent Tensors are used.

#### Returns:

A TensorInfo protocol buffer constructed based on the supplied argument.

#### Raises:

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

# tf.compat.v1.saved\_model.classification\_signature\_def

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

Creates classification signature from given examples and predictions.

### Aliases:

* tf.compat.v1.saved\_model.classification\_signature\_def
* tf.compat.v1.saved\_model.signature\_def\_utils.classification\_signature\_def

tf.compat.v1.saved\_model.classification\_signature\_def(  
    examples,  
    classes,  
    scores  
)

Defined in [python/saved\_model/signature\_def\_utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/signature_def_utils_impl.py).

This function produces signatures intended for use with the TensorFlow Serving Classify API (tensorflow\_serving/apis/prediction\_service.proto), and so constrains the input and output types to those allowed by TensorFlow Serving.

#### Args:

* **examples**: A string Tensor, expected to accept serialized tf.Examples.
* **classes**: A string Tensor. Note that the ClassificationResponse message requires that class labels are strings, not integers or anything else.
* **scores**: a float Tensor.

#### Returns:

A classification-flavored signature\_def.

#### Raises:

* **ValueError**: If examples is None.

# tf.compat.v1.saved\_model.contains\_saved\_model

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

Checks whether the provided export directory could contain a SavedModel.

### Aliases:

* tf.compat.v1.saved\_model.contains\_saved\_model
* tf.compat.v1.saved\_model.loader.maybe\_saved\_model\_directory
* tf.compat.v1.saved\_model.maybe\_saved\_model\_directory

tf.compat.v1.saved\_model.contains\_saved\_model(export\_dir)

Defined in [python/saved\_model/loader\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/loader_impl.py).

Note that the method does not load any data by itself. If the method returns false, the export directory definitely does not contain a SavedModel. If the method returns true, the export directory may contain a SavedModel but provides no guarantee that it can be loaded.

#### Args:

* **export\_dir**: Absolute string path to possible export location. For example, '/my/foo/model'.

#### Returns:

True if the export directory contains SavedModel files, False otherwise.

# tf.compat.v1.saved\_model.get\_tensor\_from\_tensor\_info

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

Returns the Tensor or SparseTensor described by a TensorInfo proto. (deprecated)

### Aliases:

* tf.compat.v1.saved\_model.get\_tensor\_from\_tensor\_info
* tf.compat.v1.saved\_model.utils.get\_tensor\_from\_tensor\_info

tf.compat.v1.saved\_model.get\_tensor\_from\_tensor\_info(  
    tensor\_info,  
    graph=None,  
    import\_scope=None  
)

Defined in [python/saved\_model/utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/utils_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.utils.get\_tensor\_from\_tensor\_info or tf.compat.v1.saved\_model.get\_tensor\_from\_tensor\_info.

#### Args:

* **tensor\_info**: A TensorInfo proto describing a Tensor or SparseTensor.
* **graph**: The tf.Graph in which tensors are looked up. If None, the current default graph is used.
* **import\_scope**: If not None, names in tensor\_info are prefixed with this string before lookup.

#### Returns:

The Tensor or SparseTensor in graph described by tensor\_info.

#### Raises:

* **KeyError**: If tensor\_info does not correspond to a tensor in graph.
* **ValueError**: If tensor\_info is malformed.

# tf.compat.v1.saved\_model.is\_valid\_signature

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

Determine whether a SignatureDef can be served by TensorFlow Serving.

### Aliases:

* tf.compat.v1.saved\_model.is\_valid\_signature
* tf.compat.v1.saved\_model.signature\_def\_utils.is\_valid\_signature

tf.compat.v1.saved\_model.is\_valid\_signature(signature\_def)

Defined in [python/saved\_model/signature\_def\_utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/signature_def_utils_impl.py).

# tf.compat.v1.saved\_model.load

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

Loads the model from a SavedModel as specified by tags. (deprecated)

### Aliases:

* tf.compat.v1.saved\_model.load
* tf.compat.v1.saved\_model.loader.load

tf.compat.v1.saved\_model.load(  
    sess,  
    tags,  
    export\_dir,  
    import\_scope=None,  
    \*\*saver\_kwargs  
)

Defined in [python/saved\_model/loader\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/loader_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.loader.load or tf.compat.v1.saved\_model.load. There will be a new function for importing SavedModels in Tensorflow 2.0.

#### Args:

* **sess**: The TensorFlow session to restore the variables.
* **tags**: Set of string tags to identify the required MetaGraphDef. These should correspond to the tags used when saving the variables using the SavedModel save() API.
* **export\_dir**: Directory in which the SavedModel protocol buffer and variables to be loaded are located.
* **import\_scope**: Optional string -- if specified, prepend this string followed by '/' to all loaded tensor names. This scope is applied to tensor instances loaded into the passed session, but it is not written through to the static MetaGraphDef protocol buffer that is returned.
* **\*\*saver\_kwargs**: Optional keyword arguments passed through to Saver.

#### Returns:

The MetaGraphDef protocol buffer loaded in the provided session. This can be used to further extract signature-defs, collection-defs, etc.

#### Raises:

* **RuntimeError**: MetaGraphDef associated with the tags cannot be found.

# tf.compat.v1.saved\_model.main\_op\_with\_restore

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

Returns a main op to init variables, tables and restore the graph. (deprecated)

### Aliases:

* tf.compat.v1.saved\_model.main\_op.main\_op\_with\_restore
* tf.compat.v1.saved\_model.main\_op\_with\_restore

tf.compat.v1.saved\_model.main\_op\_with\_restore(restore\_op\_name)

Defined in [python/saved\_model/main\_op\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/main_op_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.main\_op\_with\_restore or tf.compat.v1.saved\_model.main\_op.main\_op\_with\_restore.

Returns the main op including the group of ops that initializes all variables, initialize local variables, initialize all tables and the restore op name.

#### Args:

* **restore\_op\_name**: Name of the op to use to restore the graph.

#### Returns:

The set of ops to be run as part of the main op upon the load operation.

# tf.compat.v1.saved\_model.predict\_signature\_def

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

Creates prediction signature from given inputs and outputs.

### Aliases:

* tf.compat.v1.saved\_model.predict\_signature\_def
* tf.compat.v1.saved\_model.signature\_def\_utils.predict\_signature\_def

tf.compat.v1.saved\_model.predict\_signature\_def(  
    inputs,  
    outputs  
)

Defined in [python/saved\_model/signature\_def\_utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/signature_def_utils_impl.py).

This function produces signatures intended for use with the TensorFlow Serving Predict API (tensorflow\_serving/apis/prediction\_service.proto). This API imposes no constraints on the input and output types.

#### Args:

* **inputs**: dict of string to Tensor.
* **outputs**: dict of string to Tensor.

#### Returns:

A prediction-flavored signature\_def.

#### Raises:

* **ValueError**: If inputs or outputs is None.

# tf.compat.v1.saved\_model.regression\_signature\_def

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

Creates regression signature from given examples and predictions.

### Aliases:

* tf.compat.v1.saved\_model.regression\_signature\_def
* tf.compat.v1.saved\_model.signature\_def\_utils.regression\_signature\_def

tf.compat.v1.saved\_model.regression\_signature\_def(  
    examples,  
    predictions  
)

Defined in [python/saved\_model/signature\_def\_utils\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/signature_def_utils_impl.py).

This function produces signatures intended for use with the TensorFlow Serving Regress API (tensorflow\_serving/apis/prediction\_service.proto), and so constrains the input and output types to those allowed by TensorFlow Serving.

#### Args:

* **examples**: A string Tensor, expected to accept serialized tf.Examples.
* **predictions**: A float Tensor.

#### Returns:

A regression-flavored signature\_def.

#### Raises:

* **ValueError**: If examples is None.

# tf.compat.v1.saved\_model.simple\_save

Convenience function to build a SavedModel suitable for serving. (deprecated)

tf.compat.v1.saved\_model.simple\_save(  
    session,  
    export\_dir,  
    inputs,  
    outputs,  
    legacy\_init\_op=None  
)

Defined in [python/saved\_model/simple\_save.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/simple_save.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.simple\_save.

In many common cases, saving models for serving will be as simple as:

simple\_save(session,  
            export\_dir,  
            inputs={"x": x, "y": y},  
            outputs={"z": z})

Although in many cases it's not necessary to understand all of the many ways to configure a SavedModel, this method has a few practical implications: - It will be treated as a graph for inference / serving (i.e. uses the tag saved\_model.SERVING) - The SavedModel will load in TensorFlow Serving and supports the [Predict API](https://github.com/tensorflow/serving/blob/master/tensorflow_serving/apis/predict.proto). To use the Classify, Regress, or MultiInference APIs, please use either[tf.Estimator](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator) or the lower level [SavedModel APIs](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/README.md). - Some TensorFlow ops depend on information on disk or other information called "assets". These are generally handled automatically by adding the assets to the GraphKeys.ASSET\_FILEPATHS collection. Only assets in that collection are exported; if you need more custom behavior, you'll need to use the [SavedModelBuilder](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved_model/builder.py).

More information about SavedModel and signatures can be found here: https://github.com/tensorflow/tensorflow/blob/master/tensorflow/python/saved\_model/README.md.

#### Args:

* **session**: The TensorFlow session from which to save the meta graph and variables.
* **export\_dir**: The path to which the SavedModel will be stored.
* **inputs**: dict mapping string input names to tensors. These are added to the SignatureDef as the inputs.
* **outputs**: dict mapping string output names to tensors. These are added to the SignatureDef as the outputs.
* **legacy\_init\_op**: Legacy support for op or group of ops to execute after the restore op upon a load.

# tf.compat.v1.saved\_model.main\_op.main\_op

Returns a main op to init variables and tables. (deprecated)

tf.compat.v1.saved\_model.main\_op.main\_op()

Defined in [python/saved\_model/main\_op\_impl.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/saved_model/main_op_impl.py).

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: This function will only be available through the v1 compatibility library as tf.compat.v1.saved\_model.main\_op.main\_op.

Returns the main op including the group of ops that initializes all variables, initializes local variables and initialize all tables.

#### Returns:

The set of ops to be run as part of the main op upon the load operation.

Module: tf.sets

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

Tensorflow set operations.

Functions

[difference(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sets/difference): Compute set difference of elements in last dimension of a and b.

[intersection(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sets/intersection): Compute set intersection of elements in last dimension of a and b.

[size(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sets/size): Compute number of unique elements along last dimension of a.

[union(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sets/union): Compute set union of elements in last dimension of a and b.

# tf.sets.difference

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

Compute set difference of elements in last dimension of a and b.

### Aliases:

* tf.compat.v1.sets.difference
* tf.compat.v1.sets.set\_difference
* tf.compat.v2.sets.difference
* tf.sets.difference

tf.sets.difference(  
    a,  
    b,  
    aminusb=True,  
    validate\_indices=True  
)

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

All but the last dimension of a and b must match.

#### Example:

  import tensorflow as tf  
  import collections  
  
  # Represent the following array of sets as a sparse tensor:  
  # a = np.array([[{1, 2}, {3}], [{4}, {5, 6}]])  
  a = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((0, 0, 1), 2),  
      ((0, 1, 0), 3),  
      ((1, 0, 0), 4),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
  ])  
  a = tf.SparseTensor(list(a.keys()), list(a.values()), dense\_shape=[2, 2, 2])  
  
  # np.array([[{1, 3}, {2}], [{4, 5}, {5, 6, 7, 8}]])  
  b = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((0, 0, 1), 3),  
      ((0, 1, 0), 2),  
      ((1, 0, 0), 4),  
      ((1, 0, 1), 5),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
      ((1, 1, 2), 7),  
      ((1, 1, 3), 8),  
  ])  
  b = tf.SparseTensor(list(b.keys()), list(b.values()), dense\_shape=[2, 2, 4])  
  
  # `set\_difference` is applied to each aligned pair of sets.  
  tf.sets.difference(a, b)  
  
  # The result will be equivalent to either of:  
  #  
  # np.array([[{2}, {3}], [{}, {}]])  
  #  
  # collections.OrderedDict([  
  #     ((0, 0, 0), 2),  
  #     ((0, 1, 0), 3),  
  # ])

#### Args:

* **a**: Tensor or SparseTensor of the same type as b. If sparse, indices must be sorted in row-major order.
* **b**: Tensor or SparseTensor of the same type as a. If sparse, indices must be sorted in row-major order.
* **aminusb**: Whether to subtract b from a, vs vice versa.
* **validate\_indices**: Whether to validate the order and range of sparse indices in a and b.

#### Returns:

A SparseTensor whose shape is the same rank as a and b, and all but the last dimension the same. Elements along the last dimension contain the differences.

# tf.sets.intersection

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

Compute set intersection of elements in last dimension of a and b.

### Aliases:

* tf.compat.v1.sets.intersection
* tf.compat.v1.sets.set\_intersection
* tf.compat.v2.sets.intersection
* tf.sets.intersection

tf.sets.intersection(  
    a,  
    b,  
    validate\_indices=True  
)

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

All but the last dimension of a and b must match.

#### Example:

  import tensorflow as tf  
  import collections  
  
  # Represent the following array of sets as a sparse tensor:  
  # a = np.array([[{1, 2}, {3}], [{4}, {5, 6}]])  
  a = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((0, 0, 1), 2),  
      ((0, 1, 0), 3),  
      ((1, 0, 0), 4),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
  ])  
  a = tf.SparseTensor(list(a.keys()), list(a.values()), dense\_shape=[2,2,2])  
  
  # b = np.array([[{1}, {}], [{4}, {5, 6, 7, 8}]])  
  b = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((1, 0, 0), 4),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
      ((1, 1, 2), 7),  
      ((1, 1, 3), 8),  
  ])  
  b = tf.SparseTensor(list(b.keys()), list(b.values()), dense\_shape=[2, 2, 4])  
  
  # `tf.sets.intersection` is applied to each aligned pair of sets.  
  tf.sets.intersection(a, b)  
  
  # The result will be equivalent to either of:  
  #  
  # np.array([[{1}, {}], [{4}, {5, 6}]])  
  #  
  # collections.OrderedDict([  
  #     ((0, 0, 0), 1),  
  #     ((1, 0, 0), 4),  
  #     ((1, 1, 0), 5),  
  #     ((1, 1, 1), 6),  
  # ])

#### Args:

* **a**: Tensor or SparseTensor of the same type as b. If sparse, indices must be sorted in row-major order.
* **b**: Tensor or SparseTensor of the same type as a. If sparse, indices must be sorted in row-major order.
* **validate\_indices**: Whether to validate the order and range of sparse indices in a and b.

#### Returns:

A SparseTensor whose shape is the same rank as a and b, and all but the last dimension the same. Elements along the last dimension contain the intersections.

# tf.sets.size

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

Compute number of unique elements along last dimension of a.

### Aliases:

* tf.compat.v1.sets.set\_size
* tf.compat.v1.sets.size
* tf.compat.v2.sets.size
* tf.sets.size

tf.sets.size(  
    a,  
    validate\_indices=True  
)

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

#### Args:

* **a**: SparseTensor, with indices sorted in row-major order.
* **validate\_indices**: Whether to validate the order and range of sparse indices in a.

#### Returns:

int32 Tensor of set sizes. For a ranked n, this is a Tensor with rank n-1, and the same 1st n-1dimensions as a. Each value is the number of unique elements in the corresponding [0...n-1]dimension of a.

#### Raises:

* **TypeError**: If a is an invalid types.

# tf.sets.union

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

Compute set union of elements in last dimension of a and b.

### Aliases:

* tf.compat.v1.sets.set\_union
* tf.compat.v1.sets.union
* tf.compat.v2.sets.union
* tf.sets.union

tf.sets.union(  
    a,  
    b,  
    validate\_indices=True  
)

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

All but the last dimension of a and b must match.

#### Example:

  import tensorflow as tf  
  import collections  
  
  # [[{1, 2}, {3}], [{4}, {5, 6}]]  
  a = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((0, 0, 1), 2),  
      ((0, 1, 0), 3),  
      ((1, 0, 0), 4),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
  ])  
  a = tf.SparseTensor(list(a.keys()), list(a.values()), dense\_shape=[2, 2, 2])  
  
  # [[{1, 3}, {2}], [{4, 5}, {5, 6, 7, 8}]]  
  b = collections.OrderedDict([  
      ((0, 0, 0), 1),  
      ((0, 0, 1), 3),  
      ((0, 1, 0), 2),  
      ((1, 0, 0), 4),  
      ((1, 0, 1), 5),  
      ((1, 1, 0), 5),  
      ((1, 1, 1), 6),  
      ((1, 1, 2), 7),  
      ((1, 1, 3), 8),  
  ])  
  b = tf.SparseTensor(list(b.keys()), list(b.values()), dense\_shape=[2, 2, 4])  
  
  # `set\_union` is applied to each aligned pair of sets.  
  tf.sets.union(a, b)  
  
  # The result will be a equivalent to either of:  
  #  
  # np.array([[{1, 2, 3}, {2, 3}], [{4, 5}, {5, 6, 7, 8}]])  
  #  
  # collections.OrderedDict([  
  #     ((0, 0, 0), 1),  
  #     ((0, 0, 1), 2),  
  #     ((0, 0, 2), 3),  
  #     ((0, 1, 0), 2),  
  #     ((0, 1, 1), 3),  
  #     ((1, 0, 0), 4),  
  #     ((1, 0, 1), 5),  
  #     ((1, 1, 0), 5),  
  #     ((1, 1, 1), 6),  
  #     ((1, 1, 2), 7),  
  #     ((1, 1, 3), 8),  
  # ])

#### Args:

* **a**: Tensor or SparseTensor of the same type as b. If sparse, indices must be sorted in row-major order.
* **b**: Tensor or SparseTensor of the same type as a. If sparse, indices must be sorted in row-major order.
* **validate\_indices**: Whether to validate the order and range of sparse indices in a and b.

#### Returns:

A SparseTensor whose shape is the same rank as a and b, and all but the last dimension the same. Elements along the last dimension contain the unions.

Module: tf.signal

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

Signal processing operations.

See the [tf.signal](https://tensorflow.org/api_guides/python/contrib.signal) guide.

Functions

[dct(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/dct): Computes the 1D [Discrete Cosine Transform (DCT)][dct] of input.

[fft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft): Fast Fourier transform.

[fft2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft2d): 2D fast Fourier transform.

[fft3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft3d): 3D fast Fourier transform.

[fftshift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fftshift): Shift the zero-frequency component to the center of the spectrum.

[frame(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/frame): Expands signal's axis dimension into frames of frame\_length.

[hamming\_window(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/hamming_window): Generate a [Hamming](https://en.wikipedia.org/wiki/Window_function#Hamming_window) window.

[hann\_window(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/hann_window): Generate a [Hann window](https://en.wikipedia.org/wiki/Window_function#Hann_window).

[idct(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/idct): Computes the 1D [Inverse Discrete Cosine Transform (DCT)][idct] of input.

[ifft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft): Inverse fast Fourier transform.

[ifft2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft2d): Inverse 2D fast Fourier transform.

[ifft3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft3d): Inverse 3D fast Fourier transform.

[ifftshift(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifftshift): The inverse of fftshift.

[inverse\_stft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/inverse_stft): Computes the inverse [Short-time Fourier Transform](https://en.wikipedia.org/wiki/Short-time_Fourier_transform) of stfts.

[inverse\_stft\_window\_fn(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/inverse_stft_window_fn): Generates a window function that can be used in inverse\_stft.

[irfft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft): Inverse real-valued fast Fourier transform.

[irfft2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft2d): Inverse 2D real-valued fast Fourier transform.

[irfft3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft3d): Inverse 3D real-valued fast Fourier transform.

[linear\_to\_mel\_weight\_matrix(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/linear_to_mel_weight_matrix): Returns a matrix to warp linear scale spectrograms to the [mel scale](https://en.wikipedia.org/wiki/Mel_scale).

[mfccs\_from\_log\_mel\_spectrograms(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/mfccs_from_log_mel_spectrograms): Computes [MFCCs](https://en.wikipedia.org/wiki/Mel-frequency_cepstrum) of log\_mel\_spectrograms.

[overlap\_and\_add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/overlap_and_add): Reconstructs a signal from a framed representation.

[rfft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft): Real-valued fast Fourier transform.

[rfft2d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft2d): 2D real-valued fast Fourier transform.

[rfft3d(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft3d): 3D real-valued fast Fourier transform.

[stft(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/stft): Computes the [Short-time Fourier Transform](https://en.wikipedia.org/wiki/Short-time_Fourier_transform) of signals.

# tf.signal.dct

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

Computes the 1D [Discrete Cosine Transform (DCT)](https://en.wikipedia.org/wiki/Discrete_cosine_transform) of input.

### Aliases:

* tf.compat.v1.signal.dct
* tf.compat.v1.spectral.dct
* tf.compat.v2.signal.dct
* tf.signal.dct

tf.signal.dct(  
    input,  
    type=2,  
    n=None,  
    axis=-1,  
    norm=None,  
    name=None  
)

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

Currently only Types I, II and III are supported. Type I is implemented using a length 2N padded [tf.signal.rfft](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft). Type II is implemented using a length 2N padded [tf.signal.rfft](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft), as described here: [Type 2 DCT using 2N FFT padded (Makhoul)](https://dsp.stackexchange.com/a/10606). Type III is a fairly straightforward inverse of Type II (i.e. using a length 2N padded [tf.signal.irfft](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft)).

#### Args:

* **input**: A [..., samples] float32 Tensor containing the signals to take the DCT of.
* **type**: The DCT type to perform. Must be 1, 2 or 3.
* **n**: The length of the transform. If length is less than sequence length, only the first n elements of the sequence are considered for the DCT. If n is greater than the sequence length, zeros are padded and then the DCT is computed as usual.
* **axis**: For future expansion. The axis to compute the DCT along. Must be -1.
* **norm**: The normalization to apply. None for no normalization or 'ortho' for orthonormal normalization.
* **name**: An optional name for the operation.

#### Returns:

A [..., samples] float32 Tensor containing the DCT of input.

#### Raises:

* **ValueError**: If type is not 1, 2 or 3, axis is not -1, n is not None or greater than 0, or norm is not None or 'ortho'.
* **ValueError**: If type is 1 and norm is ortho.

#### Scipy Compatibility

Equivalent to [scipy.fftpack.dct](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.fftpack.dct.html) for Type-I, Type-II and Type-III DCT.

# tf.signal.fft

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

Fast Fourier transform.

### Aliases:

* tf.compat.v1.fft
* tf.compat.v1.signal.fft
* tf.compat.v1.spectral.fft
* tf.compat.v2.signal.fft
* tf.signal.fft

tf.signal.fft(  
    input,  
    name=None  
)

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

Computes the 1-dimensional discrete Fourier transform over the inner-most dimension of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.fft2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft2d#aliases)

2D fast Fourier transform.

### Aliases:

* tf.compat.v1.fft2d
* tf.compat.v1.signal.fft2d
* tf.compat.v1.spectral.fft2d
* tf.compat.v2.signal.fft2d
* tf.signal.fft2d

tf.signal.fft2d(  
    input,  
    name=None  
)

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

Computes the 2-dimensional discrete Fourier transform over the inner-most 2 dimensions of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.fft3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/fft3d#aliases)

3D fast Fourier transform.

### Aliases:

* tf.compat.v1.fft3d
* tf.compat.v1.signal.fft3d
* tf.compat.v1.spectral.fft3d
* tf.compat.v2.signal.fft3d
* tf.signal.fft3d

tf.signal.fft3d(  
    input,  
    name=None  
)

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

Computes the 3-dimensional discrete Fourier transform over the inner-most 3 dimensions of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex64 tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.fftshift

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

Shift the zero-frequency component to the center of the spectrum.

### Aliases:

* tf.compat.v1.signal.fftshift
* tf.compat.v2.signal.fftshift
* tf.signal.fftshift

tf.signal.fftshift(  
    x,  
    axes=None,  
    name=None  
)

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

This function swaps half-spaces for all axes listed (defaults to all). Note that y[0] is the Nyquist component only if len(x) is even.

#### For example:

x = tf.signal.fftshift([ 0.,  1.,  2.,  3.,  4., -5., -4., -3., -2., -1.])  
x.numpy() # array([-5., -4., -3., -2., -1.,  0.,  1.,  2.,  3.,  4.])

#### Args:

* **x**: Tensor, input tensor.
* **axes**: int or shape tuple, optional Axes over which to shift. Default is None, which shifts all axes.
* **name**: An optional name for the operation.

#### Returns:

A Tensor, The shifted tensor.

#### Numpy Compatibility

Equivalent to numpy.fft.fftshift. https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fftshift.html

# tf.signal.frame

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

Expands signal's axis dimension into frames of frame\_length.

### Aliases:

* tf.compat.v1.signal.frame
* tf.compat.v2.signal.frame
* tf.signal.frame

tf.signal.frame(  
    signal,  
    frame\_length,  
    frame\_step,  
    pad\_end=False,  
    pad\_value=0,  
    axis=-1,  
    name=None  
)

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

Slides a window of size frame\_length over signal's axis dimension with a stride of frame\_step, replacing the axis dimension with [frames, frame\_length] frames.

If pad\_end is True, window positions that are past the end of the axis dimension are padded with pad\_value until the window moves fully past the end of the dimension. Otherwise, only window positions that fully overlap the axis dimension are produced.

#### For example:

pcm = tf.compat.v1.placeholder(tf.float32, [None, 9152])  
frames = tf.signal.frame(pcm, 512, 180)  
magspec = tf.abs(tf.signal.rfft(frames, [512]))  
image = tf.expand\_dims(magspec, 3)

#### Args:

* **signal**: A [..., samples, ...] Tensor. The rank and dimensions may be unknown. Rank must be at least 1.
* **frame\_length**: The frame length in samples. An integer or scalar Tensor.
* **frame\_step**: The frame hop size in samples. An integer or scalar Tensor.
* **pad\_end**: Whether to pad the end of signal with pad\_value.
* **pad\_value**: An optional scalar Tensor to use where the input signal does not exist when pad\_end is True.
* **axis**: A scalar integer Tensor indicating the axis to frame. Defaults to the last axis. Supports negative values for indexing from the end.
* **name**: An optional name for the operation.

#### Returns:

A Tensor of frames with shape [..., frames, frame\_length, ...].

#### Raises:

* **ValueError**: If frame\_length, frame\_step, pad\_value, or axis are not scalar.

# tf.signal.hamming\_window

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

Generate a [Hamming](https://en.wikipedia.org/wiki/Window_function#Hann_and_Hamming_windows) window.

### Aliases:

* tf.compat.v1.signal.hamming\_window
* tf.compat.v2.signal.hamming\_window
* tf.signal.hamming\_window

tf.signal.hamming\_window(  
    window\_length,  
    periodic=True,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

#### Args:

* **window\_length**: A scalar Tensor indicating the window length to generate.
* **periodic**: A bool Tensor indicating whether to generate a periodic or symmetric window. Periodic windows are typically used for spectral analysis while symmetric windows are typically used for digital filter design.
* **dtype**: The data type to produce. Must be a floating point type.
* **name**: An optional name for the operation.

#### Returns:

A Tensor of shape [window\_length] of type dtype.

#### Raises:

* **ValueError**: If dtype is not a floating point type.

# tf.signal.hann\_window

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

Generate a [Hann window](https://en.wikipedia.org/wiki/Window_function#Hann_and_Hamming_windows).

### Aliases:

* tf.compat.v1.signal.hann\_window
* tf.compat.v2.signal.hann\_window
* tf.signal.hann\_window

tf.signal.hann\_window(  
    window\_length,  
    periodic=True,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

#### Args:

* **window\_length**: A scalar Tensor indicating the window length to generate.
* **periodic**: A bool Tensor indicating whether to generate a periodic or symmetric window. Periodic windows are typically used for spectral analysis while symmetric windows are typically used for digital filter design.
* **dtype**: The data type to produce. Must be a floating point type.
* **name**: An optional name for the operation.

#### Returns:

A Tensor of shape [window\_length] of type dtype.

#### Raises:

* **ValueError**: If dtype is not a floating point type.

# tf.signal.idct

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

Computes the 1D [Inverse Discrete Cosine Transform (DCT)](https://en.wikipedia.org/wiki/Discrete_cosine_transform#Inverse_transforms) of input.

### Aliases:

* tf.compat.v1.signal.idct
* tf.compat.v1.spectral.idct
* tf.compat.v2.signal.idct
* tf.signal.idct

tf.signal.idct(  
    input,  
    type=2,  
    n=None,  
    axis=-1,  
    norm=None,  
    name=None  
)

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

Currently only Types I, II and III are supported. Type III is the inverse of Type II, and vice versa.

Note that you must re-normalize by 1/(2n) to obtain an inverse if norm is not 'ortho'. That is:signal == idct(dct(signal)) \* 0.5 / signal.shape[-1]. When norm='ortho', we have:signal == idct(dct(signal, norm='ortho'), norm='ortho').

#### Args:

* **input**: A [..., samples] float32 Tensor containing the signals to take the DCT of.
* **type**: The IDCT type to perform. Must be 1, 2 or 3.
* **n**: For future expansion. The length of the transform. Must be None.
* **axis**: For future expansion. The axis to compute the DCT along. Must be -1.
* **norm**: The normalization to apply. None for no normalization or 'ortho' for orthonormal normalization.
* **name**: An optional name for the operation.

#### Returns:

A [..., samples] float32 Tensor containing the IDCT of input.

#### Raises:

* **ValueError**: If type is not 1, 2 or 3, n is not None,axisis not-1, ornormis notNoneor'ortho'`.

#### Scipy Compatibility

Equivalent to [scipy.fftpack.idct](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.fftpack.idct.html) for Type-I, Type-II and Type-III DCT.

# tf.signal.ifft

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

Inverse fast Fourier transform.

### Aliases:

* tf.compat.v1.ifft
* tf.compat.v1.signal.ifft
* tf.compat.v1.spectral.ifft
* tf.compat.v2.signal.ifft
* tf.signal.ifft

tf.signal.ifft(  
    input,  
    name=None  
)

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

Computes the inverse 1-dimensional discrete Fourier transform over the inner-most dimension of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.ifft2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft2d#aliases)

Inverse 2D fast Fourier transform.

### Aliases:

* tf.compat.v1.ifft2d
* tf.compat.v1.signal.ifft2d
* tf.compat.v1.spectral.ifft2d
* tf.compat.v2.signal.ifft2d
* tf.signal.ifft2d

tf.signal.ifft2d(  
    input,  
    name=None  
)

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

Computes the inverse 2-dimensional discrete Fourier transform over the inner-most 2 dimensions of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.ifft3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/ifft3d#aliases)

Inverse 3D fast Fourier transform.

### Aliases:

* tf.compat.v1.ifft3d
* tf.compat.v1.signal.ifft3d
* tf.compat.v1.spectral.ifft3d
* tf.compat.v2.signal.ifft3d
* tf.signal.ifft3d

tf.signal.ifft3d(  
    input,  
    name=None  
)

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

Computes the inverse 3-dimensional discrete Fourier transform over the inner-most 3 dimensions of input.

#### Args:

* **input**: A Tensor. Must be one of the following types: complex64, complex128. A complex64 tensor.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as input.

# tf.signal.ifftshift

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

The inverse of fftshift.

### Aliases:

* tf.compat.v1.signal.ifftshift
* tf.compat.v2.signal.ifftshift
* tf.signal.ifftshift

tf.signal.ifftshift(  
    x,  
    axes=None,  
    name=None  
)

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

Although identical for even-length x, the functions differ by one sample for odd-length x.

#### For example:

x = tf.signal.ifftshift([[ 0.,  1.,  2.],[ 3.,  4., -4.],[-3., -2., -1.]])  
x.numpy() # array([[ 4., -4.,  3.],[-2., -1., -3.],[ 1.,  2.,  0.]])

#### Args:

* **x**: Tensor, input tensor.
* **axes**: int or shape tuple Axes over which to calculate. Defaults to None, which shifts all axes.
* **name**: An optional name for the operation.

#### Returns:

A Tensor, The shifted tensor.

#### Numpy Compatibility

Equivalent to numpy.fft.ifftshift. https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.ifftshift.html

# tf.signal.inverse\_stft

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

Computes the inverse [Short-time Fourier Transform](https://en.wikipedia.org/wiki/Short-time_Fourier_transform) of stfts.

### Aliases:

* tf.compat.v1.signal.inverse\_stft
* tf.compat.v2.signal.inverse\_stft
* tf.signal.inverse\_stft

tf.signal.inverse\_stft(  
    stfts,  
    frame\_length,  
    frame\_step,  
    fft\_length=None,  
    window\_fn=tf.signal.hann\_window,  
    name=None  
)

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

To reconstruct an original waveform, a complimentary window function should be used in inverse\_stft. Such a window function can be constructed with tf.signal.inverse\_stft\_window\_fn.

#### Example:

frame\_length = 400  
frame\_step = 160  
waveform = tf.compat.v1.placeholder(dtype=tf.float32, shape=[1000])  
stft = tf.signal.stft(waveform, frame\_length, frame\_step)  
inverse\_stft = tf.signal.inverse\_stft(  
    stft, frame\_length, frame\_step,  
    window\_fn=tf.signal.inverse\_stft\_window\_fn(frame\_step))

if a custom window\_fn is used in stft, it must be passed to inverse\_stft\_window\_fn:

frame\_length = 400  
frame\_step = 160  
window\_fn = functools.partial(window\_ops.hamming\_window, periodic=True),  
waveform = tf.compat.v1.placeholder(dtype=tf.float32, shape=[1000])  
stft = tf.signal.stft(  
    waveform, frame\_length, frame\_step, window\_fn=window\_fn)  
inverse\_stft = tf.signal.inverse\_stft(  
    stft, frame\_length, frame\_step,  
    window\_fn=tf.signal.inverse\_stft\_window\_fn(  
       frame\_step, forward\_window\_fn=window\_fn))

Implemented with GPU-compatible ops and supports gradients.

#### Args:

* **stfts**: A complex64 [..., frames, fft\_unique\_bins] Tensor of STFT bins representing a batch of fft\_length-point STFTs where fft\_unique\_bins is fft\_length // 2 + 1
* **frame\_length**: An integer scalar Tensor. The window length in samples.
* **frame\_step**: An integer scalar Tensor. The number of samples to step.
* **fft\_length**: An integer scalar Tensor. The size of the FFT that produced stfts. If not provided, uses the smallest power of 2 enclosing frame\_length.
* **window\_fn**: A callable that takes a window length and a dtype keyword argument and returns a [window\_length] Tensor of samples in the provided datatype. If set to None, no windowing is used.
* **name**: An optional name for the operation.

#### Returns:

A [..., samples] Tensor of float32 signals representing the inverse STFT for each input STFT in stfts.

#### Raises:

* **ValueError**: If stfts is not at least rank 2, frame\_length is not scalar, frame\_step is not scalar, or fft\_length is not scalar.

# tf.signal.inverse\_stft\_window\_fn

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

Generates a window function that can be used in inverse\_stft.

### Aliases:

* tf.compat.v1.signal.inverse\_stft\_window\_fn
* tf.compat.v2.signal.inverse\_stft\_window\_fn
* tf.signal.inverse\_stft\_window\_fn

tf.signal.inverse\_stft\_window\_fn(  
    frame\_step,  
    forward\_window\_fn=tf.signal.hann\_window,  
    name=None  
)

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

Constructs a window that is equal to the forward window with a further pointwise amplitude correction. inverse\_stft\_window\_fn is equivalent to forward\_window\_fn in the case where it would produce an exact inverse.

See examples in inverse\_stft documentation for usage.

#### Args:

* **frame\_step**: An integer scalar Tensor. The number of samples to step.
* **forward\_window\_fn**: window\_fn used in the forward transform, stft.
* **name**: An optional name for the operation.

#### Returns:

A callable that takes a window length and a dtype keyword argument and returns a [window\_length] Tensor of samples in the provided datatype. The returned window is suitable for reconstructing original waveform in inverse\_stft.

# tf.signal.irfft

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

Inverse real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.irfft
* tf.compat.v1.spectral.irfft
* tf.compat.v2.signal.irfft
* tf.signal.irfft

tf.signal.irfft(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the inverse 1-dimensional discrete Fourier transform of a real-valued signal over the inner-most dimension of input.

The inner-most dimension of input is assumed to be the result of RFFT: the fft\_length / 2 + 1unique components of the DFT of a real-valued signal. If fft\_length is not provided, it is computed from the size of the inner-most dimension of input (fft\_length = 2 \* (inner - 1)). If the FFT length used to compute input is odd, it should be provided since it cannot be inferred properly.

Along the axis IRFFT is computed on, if fft\_length / 2 + 1 is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type complex64. A complex64 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [1]. The FFT length.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.signal.irfft2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft2d#aliases)

Inverse 2D real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.irfft2d
* tf.compat.v1.spectral.irfft2d
* tf.compat.v2.signal.irfft2d
* tf.signal.irfft2d

tf.signal.irfft2d(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the inverse 2-dimensional discrete Fourier transform of a real-valued signal over the inner-most 2 dimensions of input.

The inner-most 2 dimensions of input are assumed to be the result of RFFT2D: The inner-most dimension contains the fft\_length / 2 + 1 unique components of the DFT of a real-valued signal. If fft\_length is not provided, it is computed from the size of the inner-most 2 dimensions of input. If the FFT length used to compute input is odd, it should be provided since it cannot be inferred properly.

Along each axis IRFFT2D is computed on, if fft\_length (or fft\_length / 2 + 1 for the inner-most dimension) is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type complex64. A complex64 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [2]. The FFT length for each dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.signal.irfft3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/irfft3d#aliases)

Inverse 3D real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.irfft3d
* tf.compat.v1.spectral.irfft3d
* tf.compat.v2.signal.irfft3d
* tf.signal.irfft3d

tf.signal.irfft3d(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the inverse 3-dimensional discrete Fourier transform of a real-valued signal over the inner-most 3 dimensions of input.

The inner-most 3 dimensions of input are assumed to be the result of RFFT3D: The inner-most dimension contains the fft\_length / 2 + 1 unique components of the DFT of a real-valued signal. If fft\_length is not provided, it is computed from the size of the inner-most 3 dimensions of input. If the FFT length used to compute input is odd, it should be provided since it cannot be inferred properly.

Along each axis IRFFT3D is computed on, if fft\_length (or fft\_length / 2 + 1 for the inner-most dimension) is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type complex64. A complex64 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [3]. The FFT length for each dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.signal.linear\_to\_mel\_weight\_matrix

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

Returns a matrix to warp linear scale spectrograms to the [mel scale](https://en.wikipedia.org/wiki/Mel_scale).

### Aliases:

* tf.compat.v1.signal.linear\_to\_mel\_weight\_matrix
* tf.compat.v2.signal.linear\_to\_mel\_weight\_matrix
* tf.signal.linear\_to\_mel\_weight\_matrix

tf.signal.linear\_to\_mel\_weight\_matrix(  
    num\_mel\_bins=20,  
    num\_spectrogram\_bins=129,  
    sample\_rate=8000,  
    lower\_edge\_hertz=125.0,  
    upper\_edge\_hertz=3800.0,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

Returns a weight matrix that can be used to re-weight a Tensor containing num\_spectrogram\_binslinearly sampled frequency information from [0, sample\_rate / 2] into num\_mel\_bins frequency information from [lower\_edge\_hertz, upper\_edge\_hertz] on the [mel scale](https://en.wikipedia.org/wiki/Mel_scale).

For example, the returned matrix A can be used to right-multiply a spectrogram S of shape [frames, num\_spectrogram\_bins] of linear scale spectrum values (e.g. STFT magnitudes) to generate a "mel spectrogram" M of shape [frames, num\_mel\_bins].

# `S` has shape [frames, num\_spectrogram\_bins]  
# `M` has shape [frames, num\_mel\_bins]  
M = tf.matmul(S, A)

The matrix can be used with [tf.tensordot](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tensordot) to convert an arbitrary rank Tensor of linear-scale spectral bins into the mel scale.

# S has shape [..., num\_spectrogram\_bins].  
# M has shape [..., num\_mel\_bins].  
M = tf.tensordot(S, A, 1)  
# tf.tensordot does not support shape inference for this case yet.  
M.set\_shape(S.shape[:-1].concatenate(A.shape[-1:]))

#### Args:

* **num\_mel\_bins**: Python int. How many bands in the resulting mel spectrum.
* **num\_spectrogram\_bins**: An integer Tensor. How many bins there are in the source spectrogram data, which is understood to be fft\_size // 2 + 1, i.e. the spectrogram only contains the nonredundant FFT bins.
* **sample\_rate**: Python float. Samples per second of the input signal used to create the spectrogram. We need this to figure out the actual frequencies for each spectrogram bin, which dictates how they are mapped into the mel scale.
* **lower\_edge\_hertz**: Python float. Lower bound on the frequencies to be included in the mel spectrum. This corresponds to the lower edge of the lowest triangular band.
* **upper\_edge\_hertz**: Python float. The desired top edge of the highest frequency band.
* **dtype**: The DType of the result matrix. Must be a floating point type.
* **name**: An optional name for the operation.

#### Returns:

A Tensor of shape [num\_spectrogram\_bins, num\_mel\_bins].

#### Raises:

* **ValueError**: If num\_mel\_bins/num\_spectrogram\_bins/sample\_rate are not positive, lower\_edge\_hertz is negative, frequency edges are incorrectly ordered, or upper\_edge\_hertzis larger than the Nyquist frequency.

# tf.signal.mfccs\_from\_log\_mel\_spectrograms

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

Computes [MFCCs](https://en.wikipedia.org/wiki/Mel-frequency_cepstrum) of log\_mel\_spectrograms.

### Aliases:

* tf.compat.v1.signal.mfccs\_from\_log\_mel\_spectrograms
* tf.compat.v2.signal.mfccs\_from\_log\_mel\_spectrograms
* tf.signal.mfccs\_from\_log\_mel\_spectrograms

tf.signal.mfccs\_from\_log\_mel\_spectrograms(  
    log\_mel\_spectrograms,  
    name=None  
)

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

Implemented with GPU-compatible ops and supports gradients.

[Mel-Frequency Cepstral Coefficient (MFCC)](https://en.wikipedia.org/wiki/Mel-frequency_cepstrum) calculation consists of taking the DCT-II of a log-magnitude mel-scale spectrogram. [HTK](https://en.wikipedia.org/wiki/HTK_(software))'s MFCCs use a particular scaling of the DCT-II which is almost orthogonal normalization. We follow this convention.

All num\_mel\_bins MFCCs are returned and it is up to the caller to select a subset of the MFCCs based on their application. For example, it is typical to only use the first few for speech recognition, as this results in an approximately pitch-invariant representation of the signal.

#### For example:

sample\_rate = 16000.0  
# A Tensor of [batch\_size, num\_samples] mono PCM samples in the range [-1, 1].  
pcm = tf.compat.v1.placeholder(tf.float32, [None, None])  
  
# A 1024-point STFT with frames of 64 ms and 75% overlap.  
stfts = tf.signal.stft(pcm, frame\_length=1024, frame\_step=256,  
                       fft\_length=1024)  
spectrograms = tf.abs(stfts)  
  
# Warp the linear scale spectrograms into the mel-scale.  
num\_spectrogram\_bins = stfts.shape[-1].value  
lower\_edge\_hertz, upper\_edge\_hertz, num\_mel\_bins = 80.0, 7600.0, 80  
linear\_to\_mel\_weight\_matrix = tf.signal.linear\_to\_mel\_weight\_matrix(  
  num\_mel\_bins, num\_spectrogram\_bins, sample\_rate, lower\_edge\_hertz,  
  upper\_edge\_hertz)  
mel\_spectrograms = tf.tensordot(  
  spectrograms, linear\_to\_mel\_weight\_matrix, 1)  
mel\_spectrograms.set\_shape(spectrograms.shape[:-1].concatenate(  
  linear\_to\_mel\_weight\_matrix.shape[-1:]))  
  
# Compute a stabilized log to get log-magnitude mel-scale spectrograms.  
log\_mel\_spectrograms = tf.math.log(mel\_spectrograms + 1e-6)  
  
# Compute MFCCs from log\_mel\_spectrograms and take the first 13.  
mfccs = tf.signal.mfccs\_from\_log\_mel\_spectrograms(  
  log\_mel\_spectrograms)[..., :13]

#### Args:

* **log\_mel\_spectrograms**: A [..., num\_mel\_bins] float32 Tensor of log-magnitude mel-scale spectrograms.
* **name**: An optional name for the operation.

#### Returns:

A [..., num\_mel\_bins] float32 Tensor of the MFCCs of log\_mel\_spectrograms.

#### Raises:

* **ValueError**: If num\_mel\_bins is not positive.

# tf.signal.overlap\_and\_add

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

Reconstructs a signal from a framed representation.

### Aliases:

* tf.compat.v1.signal.overlap\_and\_add
* tf.compat.v2.signal.overlap\_and\_add
* tf.signal.overlap\_and\_add

tf.signal.overlap\_and\_add(  
    signal,  
    frame\_step,  
    name=None  
)

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

Adds potentially overlapping frames of a signal with shape [..., frames, frame\_length], offsetting subsequent frames by frame\_step. The resulting tensor has shape [..., output\_size]where

output\_size = (frames - 1) \* frame\_step + frame\_length

#### Args:

* **signal**: A [..., frames, frame\_length] Tensor. All dimensions may be unknown, and rank must be at least 2.
* **frame\_step**: An integer or scalar Tensor denoting overlap offsets. Must be less than or equal to frame\_length.
* **name**: An optional name for the operation.

#### Returns:

A Tensor with shape [..., output\_size] containing the overlap-added frames of signal's inner-most two dimensions.

#### Raises:

* **ValueError**: If signal's rank is less than 2, or frame\_step is not a scalar integer.

# tf.signal.rfft

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

Real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.rfft
* tf.compat.v1.spectral.rfft
* tf.compat.v2.signal.rfft
* tf.signal.rfft

tf.signal.rfft(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the 1-dimensional discrete Fourier transform of a real-valued signal over the inner-most dimension of input.

Since the DFT of a real signal is Hermitian-symmetric, RFFT only returns the fft\_length / 2 + 1unique components of the FFT: the zero-frequency term, followed by the fft\_length / 2 positive-frequency terms.

Along the axis RFFT is computed on, if fft\_length is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type float32. A float32 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [1]. The FFT length.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type complex64.

# tf.signal.rfft2d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft2d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft2d#aliases)

2D real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.rfft2d
* tf.compat.v1.spectral.rfft2d
* tf.compat.v2.signal.rfft2d
* tf.signal.rfft2d

tf.signal.rfft2d(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the 2-dimensional discrete Fourier transform of a real-valued signal over the inner-most 2 dimensions of input.

Since the DFT of a real signal is Hermitian-symmetric, RFFT2D only returns the fft\_length / 2 + 1unique components of the FFT for the inner-most dimension of output: the zero-frequency term, followed by the fft\_length / 2 positive-frequency terms.

Along each axis RFFT2D is computed on, if fft\_length is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type float32. A float32 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [2]. The FFT length for each dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type complex64.

# tf.signal.rfft3d

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft3d#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/signal/rfft3d#aliases)

3D real-valued fast Fourier transform.

### Aliases:

* tf.compat.v1.signal.rfft3d
* tf.compat.v1.spectral.rfft3d
* tf.compat.v2.signal.rfft3d
* tf.signal.rfft3d

tf.signal.rfft3d(  
    input\_tensor,  
    fft\_length=None,  
    name=None  
)

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

Computes the 3-dimensional discrete Fourier transform of a real-valued signal over the inner-most 3 dimensions of input.

Since the DFT of a real signal is Hermitian-symmetric, RFFT3D only returns the fft\_length / 2 + 1unique components of the FFT for the inner-most dimension of output: the zero-frequency term, followed by the fft\_length / 2 positive-frequency terms.

Along each axis RFFT3D is computed on, if fft\_length is smaller than the corresponding dimension of input, the dimension is cropped. If it is larger, the dimension is padded with zeros.

#### Args:

* **input**: A Tensor of type float32. A float32 tensor.
* **fft\_length**: A Tensor of type int32. An int32 tensor of shape [3]. The FFT length for each dimension.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type complex64.

# tf.signal.stft

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

Computes the [Short-time Fourier Transform](https://en.wikipedia.org/wiki/Short-time_Fourier_transform) of signals.

### Aliases:

* tf.compat.v1.signal.stft
* tf.compat.v2.signal.stft
* tf.signal.stft

tf.signal.stft(  
    signals,  
    frame\_length,  
    frame\_step,  
    fft\_length=None,  
    window\_fn=tf.signal.hann\_window,  
    pad\_end=False,  
    name=None  
)

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

Implemented with GPU-compatible ops and supports gradients.

#### Args:

* **signals**: A [..., samples] float32 Tensor of real-valued signals.
* **frame\_length**: An integer scalar Tensor. The window length in samples.
* **frame\_step**: An integer scalar Tensor. The number of samples to step.
* **fft\_length**: An integer scalar Tensor. The size of the FFT to apply. If not provided, uses the smallest power of 2 enclosing frame\_length.
* **window\_fn**: A callable that takes a window length and a dtype keyword argument and returns a [window\_length] Tensor of samples in the provided datatype. If set to None, no windowing is used.
* **pad\_end**: Whether to pad the end of signals with zeros when the provided frame length and step produces a frame that lies partially past its end.
* **name**: An optional name for the operation.

#### Returns:

A [..., frames, fft\_unique\_bins] Tensor of complex64 STFT values wherefft\_unique\_bins is fft\_length // 2 + 1 (the unique components of the FFT).

#### Raises:

* **ValueError**: If signals is not at least rank 1, frame\_length is not scalar, or frame\_step is not scalar.

Module: tf.sparse

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

Sparse Tensor Representation.

See also [tf.SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor).

Classes

[class SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor): Represents a sparse tensor.

Functions

[add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/add): Adds two tensors, at least one of each is a SparseTensor.

[concat(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/concat): Concatenates a list of SparseTensor along the specified dimension. (deprecated arguments)

[cross(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/cross): Generates sparse cross from a list of sparse and dense tensors.

[cross\_hashed(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/cross_hashed): Generates hashed sparse cross from a list of sparse and dense tensors.

[expand\_dims(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/expand_dims): Inserts a dimension of 1 into a tensor's shape.

[eye(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/eye): Creates a two-dimensional sparse tensor with ones along the diagonal.

[fill\_empty\_rows(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/fill_empty_rows): Fills empty rows in the input 2-D SparseTensor with a default value.

[mask(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/mask): Masks elements of IndexedSlices.

[maximum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/maximum): Returns the element-wise max of two SparseTensors.

[minimum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/minimum): Returns the element-wise min of two SparseTensors.

[reduce\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/reduce_max): Computes the max of elements across dimensions of a SparseTensor.

[reduce\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/reduce_sum): Computes the sum of elements across dimensions of a SparseTensor.

[reorder(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/reorder): Reorders a SparseTensor into the canonical, row-major ordering.

[reset\_shape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/reset_shape): Resets the shape of a SparseTensor with indices and values unchanged.

[reshape(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/reshape): Reshapes a SparseTensor to represent values in a new dense shape.

[retain(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/retain): Retains specified non-empty values within a SparseTensor.

[segment\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/segment_mean): Computes the mean along sparse segments of a tensor.

[segment\_sqrt\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/segment_sqrt_n): Computes the sum along sparse segments of a tensor divided by the sqrt(N).

[segment\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/segment_sum): Computes the sum along sparse segments of a tensor.

[slice(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/slice): Slice a SparseTensor based on the start and `size.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/softmax): Applies softmax to a batched N-D SparseTensor.

[sparse\_dense\_matmul(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/sparse_dense_matmul): Multiply SparseTensor (of rank 2) "A" by dense matrix "B".

[split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/split): Split a SparseTensor into num\_split tensors along axis.

[to\_dense(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/to_dense): Converts a SparseTensor into a dense tensor.

[to\_indicator(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/to_indicator): Converts a SparseTensor of ids into a dense bool indicator tensor.

[transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/transpose): Transposes a SparseTensor

# tf.sparse.add

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

Adds two tensors, at least one of each is a SparseTensor.

### Aliases:

* tf.compat.v2.sparse.add
* tf.sparse.add

tf.sparse.add(  
    a,  
    b,  
    threshold=0  
)

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

If one SparseTensor and one Tensor are passed in, returns a Tensor. If both arguments are SparseTensors, this returns a SparseTensor. The order of arguments does not matter. Use vanilla tf.add() for adding two dense Tensors.

The shapes of the two operands must match: broadcasting is not supported.

The indices of any input SparseTensor are assumed ordered in standard lexicographic order. If this is not the case, before this step run SparseReorder to restore index ordering.

If both arguments are sparse, we perform "clipping" as follows. By default, if two values sum to zero at some index, the output SparseTensor would still include that particular location in its index, storing a zero in the corresponding value slot. To override this, callers can specify threshold, indicating that if the sum has a magnitude strictly smaller than threshold, its corresponding value and index would then not be included. In particular, threshold == 0.0 (default) means everything is kept and actual thresholding happens only for a positive value.

For example, suppose the logical sum of two sparse operands is (densified):

[       2]  
[.1     0]  
[ 6   -.2]

Then,

* threshold == 0 (the default): all 5 index/value pairs will be returned.
* threshold == 0.11: only .1 and 0 will vanish, and the remaining three index/value pairs will be returned.
* threshold == 0.21: .1, 0, and -.2 will vanish.

#### Args:

* **a**: The first operand; SparseTensor or Tensor.
* **b**: The second operand; SparseTensor or Tensor. At least one operand must be sparse.
* **threshold**: A 0-D Tensor. The magnitude threshold that determines if an output value/index pair takes space. Its dtype should match that of the values if they are real; if the latter are complex64/complex128, then the dtype should be float32/float64, correspondingly.

#### Returns:

A SparseTensor or a Tensor, representing the sum.

#### Raises:

* **TypeError**: If both a and b are Tensors. Use tf.add() instead.

# tf.sparse.concat

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/concat#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/concat#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/concat#used_in_the_guide)

Concatenates a list of SparseTensor along the specified dimension. (deprecated arguments)

### Aliases:

* tf.compat.v2.sparse.concat
* tf.sparse.concat

tf.sparse.concat(  
    axis,  
    sp\_inputs,  
    expand\_nonconcat\_dims=False,  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(concat\_dim)**. They will be removed in a future version. Instructions for updating: concat\_dim is deprecated, use axis instead

Concatenation is with respect to the dense versions of each sparse input. It is assumed that each inputs is a SparseTensor whose elements are ordered along increasing dimension number.

If expand\_nonconcat\_dim is False, all inputs' shapes must match, except for the concat dimension. If expand\_nonconcat\_dim is True, then inputs' shapes are allowed to vary among all inputs.

The indices, values, and shapes lists must have the same length.

If expand\_nonconcat\_dim is False, then the output shape is identical to the inputs', except along the concat dimension, where it is the sum of the inputs' sizes along that dimension.

If expand\_nonconcat\_dim is True, then the output shape along the non-concat dimensions will be expand to be the largest among all inputs, and it is the sum of the inputs sizes along the concat dimension.

The output elements will be resorted to preserve the sort order along increasing dimension number.

This op runs in O(M log M) time, where M is the total number of non-empty values across all inputs. This is due to the need for an internal sort in order to concatenate efficiently across an arbitrary dimension.

For example, if axis = 1 and the inputs are

sp\_inputs[0]: shape = [2, 3]  
[0, 2]: "a"  
[1, 0]: "b"  
[1, 1]: "c"  
  
sp\_inputs[1]: shape = [2, 4]  
[0, 1]: "d"  
[0, 2]: "e"

then the output will be

shape = [2, 7]  
[0, 2]: "a"  
[0, 4]: "d"  
[0, 5]: "e"  
[1, 0]: "b"  
[1, 1]: "c"

Graphically this is equivalent to doing

[    a] concat [  d e  ] = [    a   d e  ]  
[b c  ]        [       ]   [b c          ]

Another example, if 'axis = 1' and the inputs are

sp\_inputs[0]: shape = [3, 3]  
[0, 2]: "a"  
[1, 0]: "b"  
[2, 1]: "c"  
  
sp\_inputs[1]: shape = [2, 4]  
[0, 1]: "d"  
[0, 2]: "e"

if expand\_nonconcat\_dim = False, this will result in an error. But if expand\_nonconcat\_dim = True, this will result in:

shape = [3, 7]  
[0, 2]: "a"  
[0, 4]: "d"  
[0, 5]: "e"  
[1, 0]: "b"  
[2, 1]: "c"

Graphically this is equivalent to doing

[    a] concat [  d e  ] = [    a   d e  ]  
[b    ]        [       ]   [b            ]  
[  c  ]                    [  c          ]

#### Args:

* **axis**: Dimension to concatenate along. Must be in range [-rank, rank), where rank is the number of dimensions in each input SparseTensor.
* **sp\_inputs**: List of SparseTensor to concatenate.
* **name**: A name prefix for the returned tensors (optional).
* **expand\_nonconcat\_dim**: Whether to allow the expansion in the non-concat dimensions. Defaulted to False.
* **concat\_dim**: The old (deprecated) name for axis.
* **expand\_nonconcat\_dims**: alias for expand\_nonconcat\_dim

#### Returns:

A SparseTensor with the concatenated output.

#### Raises:

* **TypeError**: If sp\_inputs is not a list of SparseTensor.

# tf.sparse.cross

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

Generates sparse cross from a list of sparse and dense tensors.

### Aliases:

* tf.compat.v1.sparse.cross
* tf.compat.v2.sparse.cross
* tf.sparse.cross

tf.sparse.cross(  
    inputs,  
    name=None  
)

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

For example, if the inputs are

\* inputs[0]: SparseTensor with shape = [2, 2]  
  [0, 0]: "a"  
  [1, 0]: "b"  
  [1, 1]: "c"  
\* inputs[1]: SparseTensor with shape = [2, 1]  
  [0, 0]: "d"  
  [1, 0]: "e"  
\* inputs[2]: Tensor [["f"], ["g"]]

then the output will be:

shape = [2, 2]  
[0, 0]: "a\_X\_d\_X\_f"  
[1, 0]: "b\_X\_e\_X\_g"  
[1, 1]: "c\_X\_e\_X\_g"

#### Args:

* **inputs**: An iterable of Tensor or SparseTensor.
* **name**: Optional name for the op.

#### Returns:

A SparseTensor of type string.

# tf.sparse.cross\_hashed

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

Generates hashed sparse cross from a list of sparse and dense tensors.

### Aliases:

* tf.compat.v1.sparse.cross\_hashed
* tf.compat.v2.sparse.cross\_hashed
* tf.sparse.cross\_hashed

tf.sparse.cross\_hashed(  
    inputs,  
    num\_buckets=0,  
    hash\_key=None,  
    name=None  
)

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

For example, if the inputs are

\* inputs[0]: SparseTensor with shape = [2, 2]  
  [0, 0]: "a"  
  [1, 0]: "b"  
  [1, 1]: "c"  
\* inputs[1]: SparseTensor with shape = [2, 1]  
  [0, 0]: "d"  
  [1, 0]: "e"  
\* inputs[2]: Tensor [["f"], ["g"]]

then the output will be:

shape = [2, 2]  
[0, 0]: FingerprintCat64(  
            Fingerprint64("f"), FingerprintCat64(  
                Fingerprint64("d"), Fingerprint64("a")))  
[1, 0]: FingerprintCat64(  
            Fingerprint64("g"), FingerprintCat64(  
                Fingerprint64("e"), Fingerprint64("b")))  
[1, 1]: FingerprintCat64(  
            Fingerprint64("g"), FingerprintCat64(  
                Fingerprint64("e"), Fingerprint64("c")))

#### Args:

* **inputs**: An iterable of Tensor or SparseTensor.
* **num\_buckets**: An int that is >= 0. output = hashed\_value%num\_buckets if num\_buckets > 0 else hashed\_value.
* **hash\_key**: Integer hash\_key that will be used by the FingerprintCat64 function. If not given, will use a default key.
* **name**: Optional name for the op.

#### Returns:

A SparseTensor of type int64.

# tf.sparse.expand\_dims

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

Inserts a dimension of 1 into a tensor's shape.

### Aliases:

* tf.compat.v1.sparse.expand\_dims
* tf.compat.v2.sparse.expand\_dims
* tf.sparse.expand\_dims

tf.sparse.expand\_dims(  
    sp\_input,  
    axis=None,  
    name=None  
)

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

Given a tensor sp\_input, this operation inserts a dimension of 1 at the dimension index axis of sp\_input's shape. The dimension index axis starts at zero; if you specify a negative number for axis it is counted backwards from the end.

#### Args:

* **sp\_input**: A SparseTensor.
* **axis**: 0-D (scalar). Specifies the dimension index at which to expand the shape of input. Must be in the range [-rank(sp\_input) - 1, rank(sp\_input)].
* **name**: The name of the output SparseTensor.

#### Returns:

A SparseTensor with the same data as sp\_input, but its shape has an additional dimension of size 1 added.

# tf.sparse.eye

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

Creates a two-dimensional sparse tensor with ones along the diagonal.

### Aliases:

* tf.compat.v1.sparse.eye
* tf.compat.v2.sparse.eye
* tf.sparse.eye

tf.sparse.eye(  
    num\_rows,  
    num\_columns=None,  
    dtype=tf.dtypes.float32,  
    name=None  
)

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

#### Args:

* **num\_rows**: Non-negative integer or int32 scalar tensor giving the number of rows in the resulting matrix.
* **num\_columns**: Optional non-negative integer or int32 scalar tensor giving the number of columns in the resulting matrix. Defaults to num\_rows.
* **dtype**: The type of element in the resulting Tensor.
* **name**: A name for this Op. Defaults to "eye".

#### Returns:

A SparseTensor of shape [num\_rows, num\_columns] with ones along the diagonal.

# tf.sparse.fill\_empty\_rows

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

Fills empty rows in the input 2-D SparseTensor with a default value.

### Aliases:

* tf.compat.v1.sparse.fill\_empty\_rows
* tf.compat.v1.sparse\_fill\_empty\_rows
* tf.compat.v2.sparse.fill\_empty\_rows
* tf.sparse.fill\_empty\_rows

tf.sparse.fill\_empty\_rows(  
    sp\_input,  
    default\_value,  
    name=None  
)

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

This op adds entries with the specified default\_value at index [row, 0] for any row in the input that does not already have a value.

For example, suppose sp\_input has shape [5, 6] and non-empty values:

[0, 1]: a  
[0, 3]: b  
[2, 0]: c  
[3, 1]: d

Rows 1 and 4 are empty, so the output will be of shape [5, 6] with values:

[0, 1]: a  
[0, 3]: b  
[1, 0]: default\_value  
[2, 0]: c  
[3, 1]: d  
[4, 0]: default\_value

Note that the input may have empty columns at the end, with no effect on this op.

The output SparseTensor will be in row-major order and will have the same shape as the input.

This op also returns an indicator vector such that

empty\_row\_indicator[i] = True iff row i was an empty row.

#### Args:

* **sp\_input**: A SparseTensor with shape [N, M].
* **default\_value**: The value to fill for empty rows, with the same type as sp\_input.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

* **sp\_ordered\_output**: A SparseTensor with shape [N, M], and with all empty rows filled in with default\_value.
* **empty\_row\_indicator**: A bool vector of length N indicating whether each input row was empty.

#### Raises:

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

# tf.sparse.mask

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

Masks elements of IndexedSlices.

### Aliases:

* tf.compat.v1.sparse.mask
* tf.compat.v1.sparse\_mask
* tf.compat.v2.sparse.mask
* tf.sparse.mask

tf.sparse.mask(  
    a,  
    mask\_indices,  
    name=None  
)

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

Given an IndexedSlices instance a, returns another IndexedSlices that contains a subset of the slices of a. Only the slices at indices not specified in mask\_indices are returned.

This is useful when you need to extract a subset of slices in an IndexedSlices object.

#### For example:

# `a` contains slices at indices [12, 26, 37, 45] from a large tensor  
# with shape [1000, 10]  
a.indices  # [12, 26, 37, 45]  
tf.shape(a.values)  # [4, 10]  
  
# `b` will be the subset of `a` slices at its second and third indices, so  
# we want to mask its first and last indices (which are at absolute  
# indices 12, 45)  
b = tf.sparse.mask(a, [12, 45])  
  
b.indices  # [26, 37]  
tf.shape(b.values)  # [2, 10]

#### Args:

* **a**: An IndexedSlices instance.
* **mask\_indices**: Indices of elements to mask.
* **name**: A name for the operation (optional).

#### Returns:

The masked IndexedSlices instance.

# tf.sparse.maximum

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

Returns the element-wise max of two SparseTensors.

### Aliases:

* tf.compat.v1.sparse.maximum
* tf.compat.v1.sparse\_maximum
* tf.compat.v2.sparse.maximum
* tf.sparse.maximum

tf.sparse.maximum(  
    sp\_a,  
    sp\_b,  
    name=None  
)

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

Assumes the two SparseTensors have the same shape, i.e., no broadcasting. Example:

sp\_zero = sparse\_tensor.SparseTensor([[0]], [0], [7])  
sp\_one = sparse\_tensor.SparseTensor([[1]], [1], [7])  
res = tf.sparse.maximum(sp\_zero, sp\_one).eval()  
# "res" should be equal to SparseTensor([[0], [1]], [0, 1], [7]).

#### Args:

* **sp\_a**: a SparseTensor operand whose dtype is real, and indices lexicographically ordered.
* **sp\_b**: the other SparseTensor operand with the same requirements (and the same shape).
* **name**: optional name of the operation.

#### Returns:

* **output**: the output SparseTensor.

# tf.sparse.minimum

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

Returns the element-wise min of two SparseTensors.

### Aliases:

* tf.compat.v1.sparse.minimum
* tf.compat.v1.sparse\_minimum
* tf.compat.v2.sparse.minimum
* tf.sparse.minimum

tf.sparse.minimum(  
    sp\_a,  
    sp\_b,  
    name=None  
)

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

Assumes the two SparseTensors have the same shape, i.e., no broadcasting. Example:

sp\_zero = sparse\_tensor.SparseTensor([[0]], [0], [7])  
sp\_one = sparse\_tensor.SparseTensor([[1]], [1], [7])  
res = tf.sparse.minimum(sp\_zero, sp\_one).eval()  
# "res" should be equal to SparseTensor([[0], [1]], [0, 0], [7]).

#### Args:

* **sp\_a**: a SparseTensor operand whose dtype is real, and indices lexicographically ordered.
* **sp\_b**: the other SparseTensor operand with the same requirements (and the same shape).
* **name**: optional name of the operation.

#### Returns:

* **output**: the output SparseTensor.

# tf.sparse.reduce\_max

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

Computes the max of elements across dimensions of a SparseTensor.

### Aliases:

* tf.compat.v2.sparse.reduce\_max
* tf.sparse.reduce\_max

tf.sparse.reduce\_max(  
    sp\_input,  
    axis=None,  
    keepdims=None,  
    output\_is\_sparse=False,  
    name=None  
)

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

This Op takes a SparseTensor and is the sparse counterpart to tf.reduce\_max(). In particular, this Op also returns a dense Tensor if output\_is\_sparse is False, or a SparseTensor if output\_is\_sparse is True.

**Note:** A gradient is not defined for this function, so it can't be used in training models that need gradient descent.

Reduces sp\_input along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned. Additionally, the axes can be negative, similar to the indexing rules in Python.

The values not defined in sp\_input don't participate in the reduce max, as opposed to be implicitly assumed 0 -- hence it can return negative values for sparse axis. But, in case there are no values inaxis, it will reduce to 0. See second example below.

#### For example:

# 'x' represents [[1, ?, 2]  
#                 [?, 3, ?]]  
# where ? is implicitly-zero.  
tf.sparse.reduce\_max(x) ==> 3  
tf.sparse.reduce\_max(x, 0) ==> [1, 3, 2]  
tf.sparse.reduce\_max(x, 1) ==> [2, 3]  # Can also use -1 as the axis.  
tf.sparse.reduce\_max(x, 1, keepdims=True) ==> [[2], [3]]  
tf.sparse.reduce\_max(x, [0, 1]) ==> 3  
  
# 'y' represents [[-7, ?]  
#                 [ 4, 3]  
#                 [ ?, ?]  
tf.sparse.reduce\_max(x, 1) ==> [-7, 4, 0]

#### Args:

* **sp\_input**: The SparseTensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce; list or scalar. If None (the default), reduces all dimensions.
* **keepdims**: If true, retain reduced dimensions with length 1.
* **output\_is\_sparse**: If true, returns a SparseTensor instead of a dense Tensor (the default).
* **name**: A name for the operation (optional).

#### Returns:

The reduced Tensor or the reduced SparseTensor if output\_is\_sparse is True.

# tf.sparse.reduce\_sum

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

Computes the sum of elements across dimensions of a SparseTensor.

### Aliases:

* tf.compat.v2.sparse.reduce\_sum
* tf.sparse.reduce\_sum

tf.sparse.reduce\_sum(  
    sp\_input,  
    axis=None,  
    keepdims=None,  
    output\_is\_sparse=False,  
    name=None  
)

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

This Op takes a SparseTensor and is the sparse counterpart to tf.reduce\_sum(). In particular, this Op also returns a dense Tensor if output\_is\_sparse is False, or a SparseTensor if output\_is\_sparse is True.

**Note:** if **output\_is\_sparse** is True, a gradient is not defined for this function, so it can't be used in training models that need gradient descent.

Reduces sp\_input along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned. Additionally, the axes can be negative, similar to the indexing rules in Python.

#### For example:

# 'x' represents [[1, ?, 1]  
#                 [?, 1, ?]]  
# where ? is implicitly-zero.  
tf.sparse.reduce\_sum(x) ==> 3  
tf.sparse.reduce\_sum(x, 0) ==> [1, 1, 1]  
tf.sparse.reduce\_sum(x, 1) ==> [2, 1]  # Can also use -1 as the axis.  
tf.sparse.reduce\_sum(x, 1, keepdims=True) ==> [[2], [1]]  
tf.sparse.reduce\_sum(x, [0, 1]) ==> 3

#### Args:

* **sp\_input**: The SparseTensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce; list or scalar. If None (the default), reduces all dimensions.
* **keepdims**: If true, retain reduced dimensions with length 1.
* **output\_is\_sparse**: If true, returns a SparseTensor instead of a dense Tensor (the default).
* **name**: A name for the operation (optional).

#### Returns:

The reduced Tensor or the reduced SparseTensor if output\_is\_sparse is True.

# tf.sparse.reorder

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

Reorders a SparseTensor into the canonical, row-major ordering.

### Aliases:

* tf.compat.v1.sparse.reorder
* tf.compat.v1.sparse\_reorder
* tf.compat.v2.sparse.reorder
* tf.sparse.reorder

tf.sparse.reorder(  
    sp\_input,  
    name=None  
)

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

Note that by convention, all sparse ops preserve the canonical ordering along increasing dimension number. The only time ordering can be violated is during manual manipulation of the indices and values to add entries.

Reordering does not affect the shape of the SparseTensor.

For example, if sp\_input has shape [4, 5] and indices / values:

[0, 3]: b  
[0, 1]: a  
[3, 1]: d  
[2, 0]: c

then the output will be a SparseTensor of shape [4, 5] and indices / values:

[0, 1]: a  
[0, 3]: b  
[2, 0]: c  
[3, 1]: d

#### Args:

* **sp\_input**: The input SparseTensor.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A SparseTensor with the same shape and non-empty values, but in canonical ordering.

#### Raises:

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

# tf.sparse.reset\_shape

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

Resets the shape of a SparseTensor with indices and values unchanged.

### Aliases:

* tf.compat.v1.sparse.reset\_shape
* tf.compat.v1.sparse\_reset\_shape
* tf.compat.v2.sparse.reset\_shape
* tf.sparse.reset\_shape

tf.sparse.reset\_shape(  
    sp\_input,  
    new\_shape=None  
)

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

If new\_shape is None, returns a copy of sp\_input with its shape reset to the tight bounding box of sp\_input. This will be a shape consisting of all zeros if sp\_input has no values.

If new\_shape is provided, then it must be larger or equal in all dimensions compared to the shape of sp\_input. When this condition is met, the returned SparseTensor will have its shape reset to new\_shape and its indices and values unchanged from that of sp\_input.

#### For example:

Consider a sp\_input with shape [2, 3, 5]:

* It is an error to set new\_shape as [3, 7] since this represents a rank-2 tensor while sp\_input is rank-3. This is either a ValueError during graph construction (if both shapes are known) or an OpError during run time.
* Setting new\_shape as [2, 3, 6] will be fine as this shape is larger or equal in every dimension compared to the original shape [2, 3, 5].
* On the other hand, setting new\_shape as [2, 3, 4] is also an error: The third dimension is smaller than the original shape [2, 3, 5](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/and%20an%0A%60InvalidArgumentError%60%20will%20be%20raised).
* If new\_shape is None, the returned SparseTensor will have a shape [2, 3, 4], which is the tight bounding box of sp\_input.

#### Args:

* **sp\_input**: The input SparseTensor.
* **new\_shape**: None or a vector representing the new shape for the returned SparseTensor.

#### Returns:

A SparseTensor indices and values unchanged from input\_sp. Its shape is new\_shape if that is set. Otherwise it is the tight bounding box of input\_sp

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.
* **ValueError**: If new\_shape represents a tensor with a different rank from that of sp\_input (if shapes are known when graph is constructed).
* **ValueError**: If new\_shape is determined during graph build to have dimension sizes that are too small.
* **OpError**: - If new\_shape has dimension sizes that are too small.
  + If shapes are not known during graph construction time, and during run time it is found out that the ranks do not match.

# tf.sparse.reshape

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

Reshapes a SparseTensor to represent values in a new dense shape.

### Aliases:

* tf.compat.v1.sparse.reshape
* tf.compat.v1.sparse\_reshape
* tf.compat.v2.sparse.reshape
* tf.sparse.reshape

tf.sparse.reshape(  
    sp\_input,  
    shape,  
    name=None  
)

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

This operation has the same semantics as reshape on the represented dense tensor. The indices of non-empty values in sp\_input are recomputed based on the new dense shape, and a new SparseTensor is returned containing the new indices and new shape. The order of non-empty values in sp\_input is unchanged.

If one component of shape is the special value -1, the size of that dimension is computed so that the total dense size remains constant. At most one component of shape can be -1. The number of dense elements implied by shape must be the same as the number of dense elements originally represented by sp\_input.

For example, if sp\_input has shape [2, 3, 6] and indices / values:

[0, 0, 0]: a  
[0, 0, 1]: b  
[0, 1, 0]: c  
[1, 0, 0]: d  
[1, 2, 3]: e

and shape is [9, -1], then the output will be a SparseTensor of shape [9, 4] and indices / values:

[0, 0]: a  
[0, 1]: b  
[1, 2]: c  
[4, 2]: d  
[8, 1]: e

#### Args:

* **sp\_input**: The input SparseTensor.
* **shape**: A 1-D (vector) int64 Tensor specifying the new dense shape of the represented SparseTensor.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A SparseTensor with the same non-empty values but with indices calculated by the new dense shape.

#### Raises:

* **TypeError**: If sp\_input is not a SparseTensor.
* **ValueError**: If argument shape requests a SparseTensor with a different number of elements than sp\_input.
* **ValueError**: If shape has more than one inferred (== -1) dimension.

# tf.sparse.retain

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

Retains specified non-empty values within a SparseTensor.

### Aliases:

* tf.compat.v1.sparse.retain
* tf.compat.v1.sparse\_retain
* tf.compat.v2.sparse.retain
* tf.sparse.retain

tf.sparse.retain(  
    sp\_input,  
    to\_retain  
)

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

For example, if sp\_input has shape [4, 5] and 4 non-empty string values:

[0, 1]: a  
[0, 3]: b  
[2, 0]: c  
[3, 1]: d

and to\_retain = [True, False, False, True], then the output will be a SparseTensor of shape [4, 5] with 2 non-empty values:

[0, 1]: a  
[3, 1]: d

#### Args:

* **sp\_input**: The input SparseTensor with N non-empty elements.
* **to\_retain**: A bool vector of length N with M true values.

#### Returns:

A SparseTensor with the same shape as the input and M non-empty elements corresponding to the true positions in to\_retain.

#### Raises:

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

# tf.sparse.segment\_mean

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

Computes the mean along sparse segments of a tensor.

### Aliases:

* tf.compat.v2.sparse.segment\_mean
* tf.sparse.segment\_mean

tf.sparse.segment\_mean(  
    data,  
    indices,  
    segment\_ids,  
    num\_segments=None,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Like [tf.math.segment\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_mean), but segment\_ids can have rank less than data's first dimension, selecting a subset of dimension 0, specified by indices. segment\_ids is allowed to have missing ids, in which case the output will be zeros at those indices. In those cases num\_segments is used to determine the size of the output.

#### Args:

* **data**: A Tensor with data that will be assembled in the output.
* **indices**: A 1-D Tensor with indices into data. Has same rank as segment\_ids.
* **segment\_ids**: A 1-D Tensor with indices into the output Tensor. Values should be sorted and can be repeated.
* **num\_segments**: An optional int32 scalar. Indicates the size of the output Tensor.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the shape as data, except for dimension 0 which has size k, the number of segments specified via num\_segments or inferred for the last element in segments\_ids.

# tf.sparse.segment\_sqrt\_n

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

Computes the sum along sparse segments of a tensor divided by the sqrt(N).

### Aliases:

* tf.compat.v2.sparse.segment\_sqrt\_n
* tf.sparse.segment\_sqrt\_n

tf.sparse.segment\_sqrt\_n(  
    data,  
    indices,  
    segment\_ids,  
    num\_segments=None,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Like [tf.sparse.segment\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/segment_mean), but instead of dividing by the size of the segment, N, divide by sqrt(N) instead.

#### Args:

* **data**: A Tensor with data that will be assembled in the output.
* **indices**: A 1-D Tensor with indices into data. Has same rank as segment\_ids.
* **segment\_ids**: A 1-D Tensor with indices into the output Tensor. Values should be sorted and can be repeated.
* **num\_segments**: An optional int32 scalar. Indicates the size of the output Tensor.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the shape as data, except for dimension 0 which has size k, the number of segments specified via num\_segments or inferred for the last element in segments\_ids.

# tf.sparse.segment\_sum

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

Computes the sum along sparse segments of a tensor.

### Aliases:

* tf.compat.v2.sparse.segment\_sum
* tf.sparse.segment\_sum

tf.sparse.segment\_sum(  
    data,  
    indices,  
    segment\_ids,  
    num\_segments=None,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Like [tf.math.segment\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum), but segment\_ids can have rank less than data's first dimension, selecting a subset of dimension 0, specified by indices. segment\_ids is allowed to have missing ids, in which case the output will be zeros at those indices. In those cases num\_segments is used to determine the size of the output.

#### For example:

c = tf.constant([[1,2,3,4], [-1,-2,-3,-4], [5,6,7,8]])  
  
# Select two rows, one segment.  
tf.sparse.segment\_sum(c, tf.constant([0, 1]), tf.constant([0, 0]))  
# => [[0 0 0 0]]  
  
# Select two rows, two segment.  
tf.sparse.segment\_sum(c, tf.constant([0, 1]), tf.constant([0, 1]))  
# => [[ 1  2  3  4]  
#     [-1 -2 -3 -4]]  
  
# With missing segment ids.  
tf.sparse.segment\_sum(c, tf.constant([0, 1]), tf.constant([0, 2]),  
                      num\_segments=4)  
# => [[ 1  2  3  4]  
#     [ 0  0  0  0]  
#     [-1 -2 -3 -4]  
#     [ 0  0  0  0]]  
  
# Select all rows, two segments.  
tf.sparse.segment\_sum(c, tf.constant([0, 1, 2]), tf.constant([0, 0, 1]))  
# => [[0 0 0 0]  
#     [5 6 7 8]]  
  
# Which is equivalent to:  
tf.math.segment\_sum(c, tf.constant([0, 0, 1]))

#### Args:

* **data**: A Tensor with data that will be assembled in the output.
* **indices**: A 1-D Tensor with indices into data. Has same rank as segment\_ids.
* **segment\_ids**: A 1-D Tensor with indices into the output Tensor. Values should be sorted and can be repeated.
* **num\_segments**: An optional int32 scalar. Indicates the size of the output Tensor.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the shape as data, except for dimension 0 which has size k, the number of segments specified via num\_segments or inferred for the last element in segments\_ids.

# tf.sparse.slice

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

Slice a SparseTensor based on the start and `size.

### Aliases:

* tf.compat.v1.sparse.slice
* tf.compat.v1.sparse\_slice
* tf.compat.v2.sparse.slice
* tf.sparse.slice

tf.sparse.slice(  
    sp\_input,  
    start,  
    size,  
    name=None  
)

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

For example, if the input is

input\_tensor = shape = [2, 7]  
[    a   d e  ]  
[b c          ]

Graphically the output tensors are:

sparse.slice([0, 0], [2, 4]) = shape = [2, 4]  
[    a  ]  
[b c    ]  
  
sparse.slice([0, 4], [2, 3]) = shape = [2, 3]  
[ d e  ]  
[      ]

#### Args:

* **sp\_input**: The SparseTensor to split.
* **start**: 1-D. tensor represents the start of the slice.
* **size**: 1-D. tensor represents the size of the slice.
* **name**: A name for the operation (optional).

#### Returns:

A SparseTensor objects resulting from splicing.

#### Raises:

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

# tf.sparse.softmax

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

Applies softmax to a batched N-D SparseTensor.

### Aliases:

* tf.compat.v1.sparse.softmax
* tf.compat.v1.sparse\_softmax
* tf.compat.v2.sparse.softmax
* tf.sparse.softmax

tf.sparse.softmax(  
    sp\_input,  
    name=None  
)

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

The inputs represent an N-D SparseTensor with logical shape [..., B, C] (where N >= 2), and with indices sorted in the canonical lexicographic order.

This op is equivalent to applying the normal tf.nn.softmax() to each innermost logical submatrix with shape [B, C], but with the catch that the implicitly zero elements do not participate. Specifically, the algorithm is equivalent to:

(1) Applies tf.nn.softmax() to a densified view of each innermost submatrix with shape [B, C], along the size-C dimension; (2) Masks out the original implicitly-zero locations; (3) Renormalizes the remaining elements.

Hence, the SparseTensor result has exactly the same non-zero indices and shape.

#### Example:

# First batch:  
# [?   e.]  
# [1.  ? ]  
# Second batch:  
# [e   ? ]  
# [e   e ]  
shape = [2, 2, 2]  # 3-D SparseTensor  
values = np.asarray([[[0., np.e], [1., 0.]], [[np.e, 0.], [np.e, np.e]]])  
indices = np.vstack(np.where(values)).astype(np.int64).T  
  
result = tf.sparse.softmax(tf.SparseTensor(indices, values, shape))  
# ...returning a 3-D SparseTensor, equivalent to:  
# [?   1.]     [1    ?]  
# [1.  ? ] and [.5  .5]  
# where ? means implicitly zero.

#### Args:

* **sp\_input**: N-D SparseTensor, where N >= 2.
* **name**: optional name of the operation.

#### Returns:

* **output**: N-D SparseTensor representing the results.

# tf.sparse.SparseTensor

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor#top_of_page)
* [Class SparseTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor#class_sparsetensor)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor#aliases)
  + [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor#used_in_the_guide)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/SparseTensor#__init__)

## Class SparseTensor

Represents a sparse tensor.

### Aliases:

* Class tf.SparseTensor
* Class tf.compat.v1.SparseTensor
* Class tf.compat.v1.sparse.SparseTensor
* Class tf.compat.v2.SparseTensor
* Class tf.compat.v2.sparse.SparseTensor
* Class tf.sparse.SparseTensor

Defined in [python/framework/sparse\_tensor.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/framework/sparse_tensor.py).

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

TensorFlow represents a sparse tensor as three separate dense tensors: indices, values, and dense\_shape. In Python, the three tensors are collected into a SparseTensor class for ease of use. If you have separate indices, values, and dense\_shape tensors, wrap them in a SparseTensorobject before passing to the ops below.

Concretely, the sparse tensor SparseTensor(indices, values, dense\_shape) comprises the following components, where N and ndims are the number of values and number of dimensions in the SparseTensor, respectively:

* indices: A 2-D int64 tensor of dense\_shape [N, ndims], which specifies the indices of the elements in the sparse tensor that contain nonzero values (elements are zero-indexed). For example, indices=[[1,3], [2,4]] specifies that the elements with indexes of [1,3] and [2,4] have nonzero values.
* values: A 1-D tensor of any type and dense\_shape [N], which supplies the values for each element in indices. For example, given indices=[[1,3], [2,4]], the parameter values=[18, 3.6] specifies that element [1,3] of the sparse tensor has a value of 18, and element [2,4] of the tensor has a value of 3.6.
* dense\_shape: A 1-D int64 tensor of dense\_shape [ndims], which specifies the dense\_shape of the sparse tensor. Takes a list indicating the number of elements in each dimension. For example, dense\_shape=[3,6] specifies a two-dimensional 3x6 tensor, dense\_shape=[2,3,4]specifies a three-dimensional 2x3x4 tensor, and dense\_shape=[9] specifies a one-dimensional tensor with 9 elements.

The corresponding dense tensor satisfies:

dense.shape = dense\_shape  
dense[tuple(indices[i])] = values[i]

By convention, indices should be sorted in row-major order (or equivalently lexicographic order on the tuples indices[i]). This is not enforced when SparseTensor objects are constructed, but most ops assume correct ordering. If the ordering of sparse tensor st is wrong, a fixed version can be obtained by calling tf.sparse.reorder(st).

Example: The sparse tensor

SparseTensor(indices=[[0, 0], [1, 2]], values=[1, 2], dense\_shape=[3, 4])

represents the dense tensor

[[1, 0, 0, 0]  
 [0, 0, 2, 0]  
 [0, 0, 0, 0]]

## \_\_init\_\_

\_\_init\_\_(  
    indices,  
    values,  
    dense\_shape  
)

Creates a SparseTensor.

#### Args:

* **indices**: A 2-D int64 tensor of shape [N, ndims].
* **values**: A 1-D tensor of any type and shape [N].
* **dense\_shape**: A 1-D int64 tensor of shape [ndims].

## Properties

### dense\_shape

A 1-D Tensor of int64 representing the shape of the dense tensor.

### dtype

The DType of elements in this tensor.

### graph

The Graph that contains the index, value, and dense\_shape tensors.

### indices

The indices of non-zero values in the represented dense tensor.

#### Returns:

A 2-D Tensor of int64 with dense\_shape [N, ndims], where N is the number of non-zero values in the tensor, and ndims is the rank.

### op

The Operation that produces values as an output.

### shape

Get the TensorShape representing the shape of the dense tensor.

#### Returns:

A TensorShape object.

### values

The non-zero values in the represented dense tensor.

#### Returns:

A 1-D Tensor of any data type.

## Methods

### \_\_div\_\_

\_\_div\_\_(  
    sp\_x,  
    y  
)

Component-wise divides a SparseTensor by a dense Tensor.

Limitation: this Op only broadcasts the dense side to the sparse side, but not the other direction.

#### Args:

* **sp\_indices**: A Tensor of type int64. 2-D. N x R matrix with the indices of non-empty values in a SparseTensor, possibly not in canonical ordering.
* **sp\_values**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64. 1-D. N non-empty values corresponding to sp\_indices.
* **sp\_shape**: A Tensor of type int64. 1-D. Shape of the input SparseTensor.
* **dense**: A Tensor. Must have the same type as sp\_values. R-D. The dense Tensor operand.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as sp\_values.

### \_\_mul\_\_

\_\_mul\_\_(  
    sp\_x,  
    y  
)

Component-wise multiplies a SparseTensor by a dense Tensor.

The output locations corresponding to the implicitly zero elements in the sparse tensor will be zero (i.e., will not take up storage space), regardless of the contents of the dense tensor (even if it's +/-INF and that INF\*0 == NaN).

Limitation: this Op only broadcasts the dense side to the sparse side, but not the other direction.

#### Args:

* **sp\_indices**: A Tensor of type int64. 2-D. N x R matrix with the indices of non-empty values in a SparseTensor, possibly not in canonical ordering.
* **sp\_values**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64. 1-D. N non-empty values corresponding to sp\_indices.
* **sp\_shape**: A Tensor of type int64. 1-D. Shape of the input SparseTensor.
* **dense**: A Tensor. Must have the same type as sp\_values. R-D. The dense Tensor operand.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as sp\_values.

### \_\_truediv\_\_

\_\_truediv\_\_(  
    sp\_x,  
    y  
)

Internal helper function for 'sp\_t / dense\_t'.

### consumers

consumers()

### eval

eval(  
    feed\_dict=None,  
    session=None  
)

Evaluates this sparse tensor in a Session.

Calling this method will execute all preceding operations that produce the inputs needed for the operation that produces this tensor.

N.B. Before invoking SparseTensor.eval(), its graph must have been launched in a session, and either a default session must be available, or session must be specified explicitly.

#### Args:

* **feed\_dict**: A dictionary that maps Tensor objects to feed values. See tf.Session.run for a description of the valid feed values.
* **session**: (Optional.) The Session to be used to evaluate this sparse tensor. If none, the default session will be used.

#### Returns:

A SparseTensorValue object.

### from\_value

@classmethod  
from\_value(  
    cls,  
    sparse\_tensor\_value  
)

### get\_shape

get\_shape()

Get the TensorShape representing the shape of the dense tensor.

#### Returns:

A TensorShape object.

# tf.sparse.sparse\_dense\_matmul

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

Multiply SparseTensor (of rank 2) "A" by dense matrix "B".

### Aliases:

* tf.compat.v1.sparse.matmul
* tf.compat.v1.sparse.sparse\_dense\_matmul
* tf.compat.v1.sparse\_tensor\_dense\_matmul
* tf.compat.v2.sparse.sparse\_dense\_matmul
* tf.sparse.sparse\_dense\_matmul

tf.sparse.sparse\_dense\_matmul(  
    sp\_a,  
    b,  
    adjoint\_a=False,  
    adjoint\_b=False,  
    name=None  
)

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

No validity checking is performed on the indices of A. However, the following input format is recommended for optimal behavior:

* If adjoint\_a == false: A should be sorted in lexicographically increasing order. Use sparse.reorder if you're not sure.
* If adjoint\_a == true: A should be sorted in order of increasing dimension 1 (i.e., "column major" order instead of "row major" order).

Using [tf.nn.embedding\_lookup\_sparse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/embedding_lookup_sparse) for sparse multiplication:

It's not obvious but you can consider embedding\_lookup\_sparse as another sparse and dense multiplication. In some situations, you may prefer to use embedding\_lookup\_sparse even though you're not dealing with embeddings.

There are two questions to ask in the decision process: Do you need gradients computed as sparse too? Is your sparse data represented as two SparseTensors: ids and values? There is more explanation about data format below. If you answer any of these questions as yes, consider using[tf.nn.embedding\_lookup\_sparse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/embedding_lookup_sparse).

Following explains differences between the expected SparseTensors: For example if dense form of your sparse data has shape [3, 5] and values:

[[  a      ]  
 [b       c]  
 [    d    ]]

SparseTensor format expected by sparse\_tensor\_dense\_matmul: sp\_a (indices, values):

[0, 1]: a  
[1, 0]: b  
[1, 4]: c  
[2, 2]: d

SparseTensor format expected by embedding\_lookup\_sparse: sp\_ids sp\_weights

[0, 0]: 1                [0, 0]: a  
[1, 0]: 0                [1, 0]: b  
[1, 1]: 4                [1, 1]: c  
[2, 0]: 2                [2, 0]: d

Deciding when to use sparse\_tensor\_dense\_matmul vs. matmul(a\_is\_sparse=True):

There are a number of questions to ask in the decision process, including:

* Will the SparseTensor A fit in memory if densified?
* Is the column count of the product large (>> 1)?
* Is the density of A larger than approximately 15%?

If the answer to several of these questions is yes, consider converting the SparseTensor to a dense one and using [tf.matmul](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/linalg/matmul) with a\_is\_sparse=True.

This operation tends to perform well when A is more sparse, if the column size of the product is small (e.g. matrix-vector multiplication), if sp\_a.dense\_shape takes on large values.

Below is a rough speed comparison between sparse\_tensor\_dense\_matmul, labeled 'sparse', and matmul(a\_is\_sparse=True), labeled 'dense'. For purposes of the comparison, the time spent converting from a SparseTensor to a dense Tensor is not included, so it is overly conservative with respect to the time ratio.

#### Benchmark system:

CPU: Intel Ivybridge with HyperThreading (6 cores) dL1:32KB dL2:256KB dL3:12MB GPU: NVidia Tesla k40c

#### Compiled with:

-c opt --config=cuda --copt=-mavx

tensorflow/python/sparse\_tensor\_dense\_matmul\_op\_test --benchmarks  
A sparse [m, k] with % nonzero values between 1% and 80%  
B dense [k, n]  
  
% nnz  n   gpu   m     k     dt(dense)     dt(sparse)   dt(sparse)/dt(dense)  
0.01   1   True  100   100   0.000221166   0.00010154   0.459112  
0.01   1   True  100   1000  0.00033858    0.000109275  0.322745  
0.01   1   True  1000  100   0.000310557   9.85661e-05  0.317385  
0.01   1   True  1000  1000  0.0008721     0.000100875  0.115669  
0.01   1   False 100   100   0.000208085   0.000107603  0.51711  
0.01   1   False 100   1000  0.000327112   9.51118e-05  0.290762  
0.01   1   False 1000  100   0.000308222   0.00010345   0.335635  
0.01   1   False 1000  1000  0.000865721   0.000101397  0.117124  
0.01   10  True  100   100   0.000218522   0.000105537  0.482958  
0.01   10  True  100   1000  0.000340882   0.000111641  0.327506  
0.01   10  True  1000  100   0.000315472   0.000117376  0.372064  
0.01   10  True  1000  1000  0.000905493   0.000123263  0.136128  
0.01   10  False 100   100   0.000221529   9.82571e-05  0.44354  
0.01   10  False 100   1000  0.000330552   0.000112615  0.340687  
0.01   10  False 1000  100   0.000341277   0.000114097  0.334324  
0.01   10  False 1000  1000  0.000819944   0.000120982  0.147549  
0.01   25  True  100   100   0.000207806   0.000105977  0.509981  
0.01   25  True  100   1000  0.000322879   0.00012921   0.400181  
0.01   25  True  1000  100   0.00038262    0.00014158   0.370035  
0.01   25  True  1000  1000  0.000865438   0.000202083  0.233504  
0.01   25  False 100   100   0.000209401   0.000104696  0.499979  
0.01   25  False 100   1000  0.000321161   0.000130737  0.407076  
0.01   25  False 1000  100   0.000377012   0.000136801  0.362856  
0.01   25  False 1000  1000  0.000861125   0.00020272   0.235413  
0.2    1   True  100   100   0.000206952   9.69219e-05  0.46833  
0.2    1   True  100   1000  0.000348674   0.000147475  0.422959  
0.2    1   True  1000  100   0.000336908   0.00010122   0.300439  
0.2    1   True  1000  1000  0.001022      0.000203274  0.198898  
0.2    1   False 100   100   0.000207532   9.5412e-05   0.459746  
0.2    1   False 100   1000  0.000356127   0.000146824  0.41228  
0.2    1   False 1000  100   0.000322664   0.000100918  0.312764  
0.2    1   False 1000  1000  0.000998987   0.000203442  0.203648  
0.2    10  True  100   100   0.000211692   0.000109903  0.519165  
0.2    10  True  100   1000  0.000372819   0.000164321  0.440753  
0.2    10  True  1000  100   0.000338651   0.000144806  0.427596  
0.2    10  True  1000  1000  0.00108312    0.000758876  0.70064  
0.2    10  False 100   100   0.000215727   0.000110502  0.512231  
0.2    10  False 100   1000  0.000375419   0.0001613    0.429653  
0.2    10  False 1000  100   0.000336999   0.000145628  0.432132  
0.2    10  False 1000  1000  0.00110502    0.000762043  0.689618  
0.2    25  True  100   100   0.000218705   0.000129913  0.594009  
0.2    25  True  100   1000  0.000394794   0.00029428   0.745402  
0.2    25  True  1000  100   0.000404483   0.0002693    0.665788  
0.2    25  True  1000  1000  0.0012002     0.00194494   1.62052  
0.2    25  False 100   100   0.000221494   0.0001306    0.589632  
0.2    25  False 100   1000  0.000396436   0.000297204  0.74969  
0.2    25  False 1000  100   0.000409346   0.000270068  0.659754  
0.2    25  False 1000  1000  0.00121051    0.00193737   1.60046  
0.5    1   True  100   100   0.000214981   9.82111e-05  0.456836  
0.5    1   True  100   1000  0.000415328   0.000223073  0.537101  
0.5    1   True  1000  100   0.000358324   0.00011269   0.314492  
0.5    1   True  1000  1000  0.00137612    0.000437401  0.317851  
0.5    1   False 100   100   0.000224196   0.000101423  0.452386  
0.5    1   False 100   1000  0.000400987   0.000223286  0.556841  
0.5    1   False 1000  100   0.000368825   0.00011224   0.304318  
0.5    1   False 1000  1000  0.00136036    0.000429369  0.31563  
0.5    10  True  100   100   0.000222125   0.000112308  0.505608  
0.5    10  True  100   1000  0.000461088   0.00032357   0.701753  
0.5    10  True  1000  100   0.000394624   0.000225497  0.571422  
0.5    10  True  1000  1000  0.00158027    0.00190898   1.20801  
0.5    10  False 100   100   0.000232083   0.000114978  0.495418  
0.5    10  False 100   1000  0.000454574   0.000324632  0.714146  
0.5    10  False 1000  100   0.000379097   0.000227768  0.600817  
0.5    10  False 1000  1000  0.00160292    0.00190168   1.18638  
0.5    25  True  100   100   0.00023429    0.000151703  0.647501  
0.5    25  True  100   1000  0.000497462   0.000598873  1.20386  
0.5    25  True  1000  100   0.000460778   0.000557038  1.20891  
0.5    25  True  1000  1000  0.00170036    0.00467336   2.74845  
0.5    25  False 100   100   0.000228981   0.000155334  0.678371  
0.5    25  False 100   1000  0.000496139   0.000620789  1.25124  
0.5    25  False 1000  100   0.00045473    0.000551528  1.21287  
0.5    25  False 1000  1000  0.00171793    0.00467152   2.71927  
0.8    1   True  100   100   0.000222037   0.000105301  0.47425  
0.8    1   True  100   1000  0.000410804   0.000329327  0.801664  
0.8    1   True  1000  100   0.000349735   0.000131225  0.375212  
0.8    1   True  1000  1000  0.00139219    0.000677065  0.48633  
0.8    1   False 100   100   0.000214079   0.000107486  0.502085  
0.8    1   False 100   1000  0.000413746   0.000323244  0.781261  
0.8    1   False 1000  100   0.000348983   0.000131983  0.378193  
0.8    1   False 1000  1000  0.00136296    0.000685325  0.50282  
0.8    10  True  100   100   0.000229159   0.00011825   0.516017  
0.8    10  True  100   1000  0.000498845   0.000532618  1.0677  
0.8    10  True  1000  100   0.000383126   0.00029935   0.781336  
0.8    10  True  1000  1000  0.00162866    0.00307312   1.88689  
0.8    10  False 100   100   0.000230783   0.000124958  0.541452  
0.8    10  False 100   1000  0.000493393   0.000550654  1.11606  
0.8    10  False 1000  100   0.000377167   0.000298581  0.791642  
0.8    10  False 1000  1000  0.00165795    0.00305103   1.84024  
0.8    25  True  100   100   0.000233496   0.000175241  0.75051  
0.8    25  True  100   1000  0.00055654    0.00102658   1.84458  
0.8    25  True  1000  100   0.000463814   0.000783267  1.68875  
0.8    25  True  1000  1000  0.00186905    0.00755344   4.04132  
0.8    25  False 100   100   0.000240243   0.000175047  0.728625  
0.8    25  False 100   1000  0.000578102   0.00104499   1.80763  
0.8    25  False 1000  100   0.000485113   0.000776849  1.60138  
0.8    25  False 1000  1000  0.00211448    0.00752736   3.55992

#### Args:

* **sp\_a**: SparseTensor A, of rank 2.
* **b**: A dense Matrix with the same dtype as sp\_a.
* **adjoint\_a**: Use the adjoint of A in the matrix multiply. If A is complex, this is transpose(conj(A)). Otherwise it's transpose(A).
* **adjoint\_b**: Use the adjoint of B in the matrix multiply. If B is complex, this is transpose(conj(B)). Otherwise it's transpose(B).
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A dense matrix (pseudo-code in dense np.matrix notation): A = A.H if adjoint\_a else A B = B.H if adjoint\_b else B return A\*B

# tf.sparse.split

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

Split a SparseTensor into num\_split tensors along axis.

### Aliases:

* tf.compat.v2.sparse.split
* tf.sparse.split

tf.sparse.split(  
    sp\_input=None,  
    num\_split=None,  
    axis=None,  
    name=None  
)

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

If the sp\_input.dense\_shape[axis] is not an integer multiple of num\_split each slice starting from 0:shape[axis] % num\_split gets extra one dimension. For example, if axis = 1 and num\_split = 2 and the input is:

input\_tensor = shape = [2, 7]  
[    a   d e  ]  
[b c          ]

Graphically the output tensors are:

output\_tensor[0] =  
[    a ]  
[b c   ]  
  
output\_tensor[1] =  
[ d e  ]  
[      ]

#### Args:

* **sp\_input**: The SparseTensor to split.
* **num\_split**: A Python integer. The number of ways to split.
* **axis**: A 0-D int32 Tensor. The dimension along which to split.
* **name**: A name for the operation (optional).

#### Returns:

num\_split SparseTensor objects resulting from splitting value.

#### Raises:

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

# tf.sparse.to\_dense

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/to_dense#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/to_dense#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sparse/to_dense#used_in_the_guide)

Converts a SparseTensor into a dense tensor.

### Aliases:

* tf.compat.v1.sparse.to\_dense
* tf.compat.v1.sparse\_tensor\_to\_dense
* tf.compat.v2.sparse.to\_dense
* tf.sparse.to\_dense

tf.sparse.to\_dense(  
    sp\_input,  
    default\_value=0,  
    validate\_indices=True,  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

This op is a convenience wrapper around sparse\_to\_dense for SparseTensors.

For example, if sp\_input has shape [3, 5] and non-empty string values:

[0, 1]: a  
[0, 3]: b  
[2, 0]: c

and default\_value is x, then the output will be a dense [3, 5] string tensor with values:

[[x a x b x]  
 [x x x x x]  
 [c x x x x]]

Indices must be without repeats. This is only tested if validate\_indices is True.

#### Args:

* **sp\_input**: The input SparseTensor.
* **default\_value**: Scalar value to set for indices not specified in sp\_input. Defaults to zero.
* **validate\_indices**: A boolean value. If True, indices are checked to make sure they are sorted in lexicographic order and that there are no repeats.
* **name**: A name prefix for the returned tensors (optional).

#### Returns:

A dense tensor with shape sp\_input.dense\_shape and values specified by the non-empty values in sp\_input. Indices not in sp\_input are assigned default\_value.

#### Raises:

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

# tf.sparse.to\_indicator

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

Converts a SparseTensor of ids into a dense bool indicator tensor.

### Aliases:

* tf.compat.v1.sparse.to\_indicator
* tf.compat.v1.sparse\_to\_indicator
* tf.compat.v2.sparse.to\_indicator
* tf.sparse.to\_indicator

tf.sparse.to\_indicator(  
    sp\_input,  
    vocab\_size,  
    name=None  
)

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

The last dimension of sp\_input.indices is discarded and replaced with the values of sp\_input. If sp\_input.dense\_shape = [D0, D1, ..., Dn, K], then output.shape = [D0, D1, ..., Dn, vocab\_size], where

output[d\_0, d\_1, ..., d\_n, sp\_input[d\_0, d\_1, ..., d\_n, k]] = True

and False elsewhere in output.

For example, if sp\_input.dense\_shape = [2, 3, 4] with non-empty values:

[0, 0, 0]: 0  
[0, 1, 0]: 10  
[1, 0, 3]: 103  
[1, 1, 2]: 150  
[1, 1, 3]: 149  
[1, 1, 4]: 150  
[1, 2, 1]: 121

and vocab\_size = 200, then the output will be a [2, 3, 200] dense bool tensor with False everywhere except at positions

(0, 0, 0), (0, 1, 10), (1, 0, 103), (1, 1, 149), (1, 1, 150),  
(1, 2, 121).

Note that repeats are allowed in the input SparseTensor. This op is useful for converting SparseTensors into dense formats for compatibility with ops that expect dense tensors.

The input SparseTensor must be in row-major order.

#### Args:

* **sp\_input**: A SparseTensor with values property of type int32 or int64.
* **vocab\_size**: A scalar int64 Tensor (or Python int) containing the new size of the last dimension, all(0 <= sp\_input.values < vocab\_size).
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A dense bool indicator tensor representing the indices with specified value.

#### Raises:

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

# tf.sparse.transpose

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

Transposes a SparseTensor

### Aliases:

* tf.compat.v1.sparse.transpose
* tf.compat.v1.sparse\_transpose
* tf.compat.v2.sparse.transpose
* tf.sparse.transpose

tf.sparse.transpose(  
    sp\_input,  
    perm=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 returned tensor's dimension i will correspond to the input dimension perm[i]. If perm is not given, it is set to (n-1...0), where n is the rank of the input tensor. Hence by default, this operation performs a regular matrix transpose on 2-D input Tensors.

For example, if sp\_input has shape [4, 5] and indices / values:

[0, 3]: b  
[0, 1]: a  
[3, 1]: d  
[2, 0]: c

then the output will be a SparseTensor of shape [5, 4] and indices / values:

[0, 2]: c  
[1, 0]: a  
[1, 3]: d  
[3, 0]: b

#### Args:

* **sp\_input**: The input SparseTensor.
* **perm**: A permutation of the dimensions of sp\_input.
* **name**: A name prefix for the returned tensors (optional)

#### Returns:

A transposed SparseTensor.

#### Raises:

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

Module: tf.compat.v1.strings / tf.strings

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

Operations for working with string Tensors.

Functions

[as\_string(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/as_string): Converts each entry in the given tensor to strings. Supports many numeric

[bytes\_split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/bytes_split): Split string elements of input into bytes.

[format(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/format): Formats a string template using a list of tensors.

[join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join): Joins the strings in the given list of string tensors into one tensor;

[length(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/length): String lengths of input.

[lower(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/lower): TODO: add doc.

[reduce\_join(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/reduce_join): Joins a string Tensor across the given dimensions.

[regex\_full\_match(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_full_match): Check if the input matches the regex pattern.

[regex\_replace(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/regex_replace): Replace elements of input matching regex pattern with rewrite.

[split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/split): Split elements of input based on sep.

[strip(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/strip): Strip leading and trailing whitespaces from the Tensor.

[substr(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/strings/substr): Return substrings from Tensor of strings.

[to\_hash\_bucket(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/string_to_hash_bucket): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_hash\_bucket\_fast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_hash\_bucket\_strong(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_strong): Converts each string in the input Tensor to its hash mod by a number of buckets.

[to\_number(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/string_to_number): Converts each string in the input Tensor to the specified numeric type.

[unicode\_decode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_decode): Decodes each string in input into a sequence of Unicode code points.

[unicode\_decode\_with\_offsets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_decode_with_offsets): Decodes each string into a sequence of code points with start offsets.

[unicode\_encode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_encode): Encodes each sequence of Unicode code points in input into a string.

[unicode\_script(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_script): Determine the script codes of a given tensor of Unicode integer code points.

[unicode\_split(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split): Splits each string in input into a sequence of Unicode code points.

[unicode\_split\_with\_offsets(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split_with_offsets): Splits each string into a sequence of code points with start offsets.

[unicode\_transcode(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_transcode): Transcode the input text from a source encoding to a destination encoding.

[upper(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/upper): TODO: add doc.

# tf.compat.v1.strings.length

String lengths of input.

tf.compat.v1.strings.length(  
    input,  
    name=None,  
    unit='BYTE'  
)

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

Computes the length of each string given in the input tensor.

#### Args:

* **input**: A Tensor of type string. The string for which to compute the length.
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is counted to compute string length. One of: "BYTE" (for the number of bytes in each string) or "UTF8\_CHAR" (for the number of UTF-8 encoded Unicode code points in each string). Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int32.

# tf.compat.v1.strings.split

Split elements of input based on sep.

tf.compat.v1.strings.split(  
    input=None,  
    sep=None,  
    maxsplit=-1,  
    result\_type='SparseTensor',  
    source=None,  
    name=None  
)

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

Let N be the size of input (typically N will be the batch size). Split each element of input based on sep and return a SparseTensor or RaggedTensor containing the split tokens. Empty tokens are ignored.

#### Examples:

>>> tf.strings.split(['hello world', 'a b c'])  
tf.SparseTensor(indices=[[0, 0], [0, 1], [1, 0], [1, 1], [1, 2]],  
                values=['hello', 'world', 'a', 'b', 'c']  
                dense\_shape=[2, 3])  
  
>>> tf.strings.split(['hello world', 'a b c'], result\_type="RaggedTensor")  
<tf.RaggedTensor [['hello', 'world'], ['a', 'b', 'c']]>

If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings. For example, input of "1<>2<><>3" and sep of "<>" returns ["1", "2", "", "3"]. If sep is None or an empty string, consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

Note that the above mentioned behavior matches python's str.split.

#### Args:

* **input**: A string Tensor of rank N, the strings to split. If rank(input) is not known statically, then it is assumed to be 1.
* **sep**: 0-D string Tensor, the delimiter character.
* **maxsplit**: An int. If maxsplit > 0, limit of the split of the result.
* **result\_type**: The tensor type for the result: one of "RaggedTensor" or "SparseTensor".
* **source**: alias for "input" argument.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: If sep is not a string.

#### Returns:

A SparseTensor or RaggedTensor of rank N+1, the strings split according to the delimiter.

# tf.compat.v1.strings.substr

Return substrings from Tensor of strings.

tf.compat.v1.strings.substr(  
    input,  
    pos,  
    len,  
    name=None,  
    unit='BYTE'  
)

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

For each string in the input Tensor, creates a substring starting at index pos with a total length of len.

If len defines a substring that would extend beyond the length of the input string, then as many characters as possible are used.

A negative pos indicates distance within the string backwards from the end.

If pos specifies an index which is out of range for any of the input strings, then an InvalidArgumentError is thrown.

pos and len must have the same shape, otherwise a ValueError is thrown on Op creation.

NOTE: Substr supports broadcasting up to two dimensions. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

Examples

Using scalar pos and len:

input = [b'Hello', b'World']  
position = 1  
length = 3  
  
output = [b'ell', b'orl']

Using pos and len with same shape as input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen']]  
position = [[1, 2, 3],  
            [1, 2, 3],  
            [1, 2, 3]]  
length =   [[2, 3, 4],  
            [4, 3, 2],  
            [5, 5, 5]]  
  
output = [[b'en', b'eve', b'lve'],  
          [b'hirt', b'urt', b'te'],  
          [b'ixtee', b'vente', b'hteen']]

Broadcasting pos and len onto input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen'],  
         [b'nineteen', b'twenty', b'twentyone']]  
position = [1, 2, 3]  
length =   [1, 2, 3]  
  
output = [[b'e', b'ev', b'lve'],  
          [b'h', b'ur', b'tee'],  
          [b'i', b've', b'hte'],  
          [b'i', b'en', b'nty']]

Broadcasting input onto pos and len:

input = b'thirteen'  
position = [1, 5, 7]  
length =   [3, 2, 1]  
  
output = [b'hir', b'ee', b'n']

#### Args:

* **input**: A Tensor of type string. Tensor of strings
* **pos**: A Tensor. Must be one of the following types: int32, int64. Scalar defining the position of first character in each substring
* **len**: A Tensor. Must have the same type as pos. Scalar defining the number of characters to include in each substring
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is used to create the substring. One of: "BYTE" (for defining position and length by bytes) or "UTF8\_CHAR" (for the UTF-8 encoded Unicode code points). The default is "BYTE". Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.as\_string

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

Converts each entry in the given tensor to strings. Supports many numeric

### Aliases:

* tf.as\_string
* tf.compat.v1.as\_string
* tf.compat.v1.dtypes.as\_string
* tf.compat.v1.strings.as\_string
* tf.compat.v2.as\_string
* tf.compat.v2.strings.as\_string
* tf.strings.as\_string

tf.strings.as\_string(  
    input,  
    precision=-1,  
    scientific=False,  
    shortest=False,  
    width=-1,  
    fill='',  
    name=None  
)

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

types and boolean.

#### Args:

* **input**: A Tensor. Must be one of the following types: int8, int16, int32, int64, complex64, complex128, float32, float64, bool.
* **precision**: An optional int. Defaults to -1. The post-decimal precision to use for floating point numbers. Only used if precision > -1.
* **scientific**: An optional bool. Defaults to False. Use scientific notation for floating point numbers.
* **shortest**: An optional bool. Defaults to False. Use shortest representation (either scientific or standard) for floating point numbers.
* **width**: An optional int. Defaults to -1. Pad pre-decimal numbers to this width. Applies to both floating point and integer numbers. Only used if width > -1.
* **fill**: An optional string. Defaults to "". The value to pad if width > -1. If empty, pads with spaces. Another typical value is '0'. String cannot be longer than 1 character.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.bytes\_split

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

Split string elements of input into bytes.

### Aliases:

* tf.compat.v1.strings.bytes\_split
* tf.compat.v2.strings.bytes\_split
* tf.strings.bytes\_split

tf.strings.bytes\_split(  
    input,  
    name=None  
)

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

#### Examples:

>>> tf.strings.to\_bytes('hello')  
['h', 'e', 'l', 'l', 'o']  
>>> tf.strings.to\_bytes(['hello', '123'])  
<RaggedTensor [['h', 'e', 'l', 'l', 'o'], ['1', '2', '3']]>

Note that this op splits strings into bytes, not unicode characters. To split strings into unicode characters, use [tf.strings.unicode\_split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split).

See also: [tf.io.decode\_raw](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/io/decode_raw), [tf.strings.split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/split), [tf.strings.unicode\_split](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/unicode_split).

#### Args:

* **input**: A string Tensor or RaggedTensor: the strings to split. Must have a statically known rank (N).
* **name**: A name for the operation (optional).

#### Returns:

A RaggedTensor of rank N+1: the bytes that make up the soruce strings.

# tf.strings.format

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

Formats a string template using a list of tensors.

### Aliases:

* tf.compat.v1.strings.format
* tf.compat.v2.strings.format
* tf.strings.format

tf.strings.format(  
    template,  
    inputs,  
    placeholder='{}',  
    summarize=3,  
    name=None  
)

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

Formats a string template using a list of tensors, abbreviating tensors by only printing the first and last summarize elements of each dimension (recursively). If formatting only one tensor into a template, the tensor does not have to be wrapped in a list.

#### Example:

Formatting a single-tensor template:

sess = tf.compat.v1.Session()  
with sess.as\_default():  
    tensor = tf.range(10)  
    formatted = tf.strings.format("tensor: {}, suffix", tensor)  
    out = sess.run(formatted)  
    expected = "tensor: [0 1 2 ... 7 8 9], suffix"  
  
    assert(out.decode() == expected)

Formatting a multi-tensor template:

sess = tf.compat.v1.Session()  
with sess.as\_default():  
    tensor\_one = tf.reshape(tf.range(100), [10, 10])  
    tensor\_two = tf.range(10)  
    formatted = tf.strings.format("first: {}, second: {}, suffix",  
      (tensor\_one, tensor\_two))  
  
    out = sess.run(formatted)  
    expected = ("first: [[0 1 2 ... 7 8 9]\n"  
          " [10 11 12 ... 17 18 19]\n"  
          " [20 21 22 ... 27 28 29]\n"  
          " ...\n"  
          " [70 71 72 ... 77 78 79]\n"  
          " [80 81 82 ... 87 88 89]\n"  
          " [90 91 92 ... 97 98 99]], second: [0 1 2 ... 7 8 9], suffix")  
  
    assert(out.decode() == expected)

#### Args:

* **template**: A string template to format tensor values into.
* **inputs**: A list of Tensor objects, or a single Tensor. The list of tensors to format into the template string. If a solitary tensor is passed in, the input tensor will automatically be wrapped as a list.
* **placeholder**: An optional string. Defaults to {}. At each placeholder occurring in the template, a subsequent tensor will be inserted.
* **summarize**: An optional int. Defaults to 3. When formatting the tensors, show the first and last summarize entries of each tensor dimension (recursively). If set to -1, all elements of the tensor will be shown.
* **name**: A name for the operation (optional).

#### Returns:

A scalar Tensor of type string.

#### Raises:

* **ValueError**: if the number of placeholders does not match the number of inputs.

# tf.strings.join

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/join#used_in_the_guide)

Joins the strings in the given list of string tensors into one tensor;

### Aliases:

* tf.compat.v1.string\_join
* tf.compat.v1.strings.join
* tf.compat.v2.strings.join
* tf.strings.join

tf.strings.join(  
    inputs,  
    separator='',  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

with the given separator (default is an empty separator).

#### Args:

* **inputs**: A list of at least 1 Tensor objects with type string. A list of string tensors. The tensors must all have the same shape, or be scalars. Scalars may be mixed in; these will be broadcast to the shape of non-scalar inputs.
* **separator**: An optional string. Defaults to "". string, an optional join separator.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.length

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

### Aliases:

* tf.compat.v2.strings.length
* tf.strings.length

tf.strings.length(  
    input,  
    unit='BYTE',  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

# tf.strings.lower

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

TODO: add doc.

### Aliases:

* tf.compat.v1.strings.lower
* tf.compat.v2.strings.lower
* tf.strings.lower

tf.strings.lower(  
    input,  
    encoding='',  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string.
* **encoding**: An optional string. Defaults to "".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.reduce\_join

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

### Aliases:

* tf.compat.v2.strings.reduce\_join
* tf.strings.reduce\_join

tf.strings.reduce\_join(  
    inputs,  
    axis=None,  
    keepdims=False,  
    separator='',  
    name=None  
)

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

# tf.strings.regex\_full\_match

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

Check if the input matches the regex pattern.

### Aliases:

* tf.compat.v1.strings.regex\_full\_match
* tf.compat.v2.strings.regex\_full\_match
* tf.strings.regex\_full\_match

tf.strings.regex\_full\_match(  
    input,  
    pattern,  
    name=None  
)

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

The input is a string tensor of any shape. The pattern is a scalar string tensor which is applied to every element of the input tensor. The boolean values (True or False) of the output tensor indicate if the input matches the regex pattern provided.

The pattern follows the re2 syntax (https://github.com/google/re2/wiki/Syntax)

#### Args:

* **input**: A Tensor of type string. A string tensor of the text to be processed.
* **pattern**: A Tensor of type string. A scalar string tensor containing the regular expression to match the input.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.strings.regex\_replace

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

Replace elements of input matching regex pattern with rewrite.

### Aliases:

* tf.compat.v1.regex\_replace
* tf.compat.v1.strings.regex\_replace
* tf.compat.v2.strings.regex\_replace
* tf.strings.regex\_replace

tf.strings.regex\_replace(  
    input,  
    pattern,  
    rewrite,  
    replace\_global=True,  
    name=None  
)

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

### Used in the tutorials:

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

#### Args:

* **input**: string Tensor, the source strings to process.
* **pattern**: string or scalar string Tensor, regular expression to use, see more details at https://github.com/google/re2/wiki/Syntax
* **rewrite**: string or scalar string Tensor, value to use in match replacement, supports backslash-escaped digits (\1 to \9) can be to insert text matching corresponding parenthesized group.
* **replace\_global**: bool, if True replace all non-overlapping matches, else replace only the first match.
* **name**: A name for the operation (optional).

#### Returns:

string Tensor of the same shape as input with specified replacements.

# tf.strings.split

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

Split elements of input based on sep into a RaggedTensor.

### Aliases:

* tf.compat.v2.strings.split
* tf.strings.split

tf.strings.split(  
    input,  
    sep=None,  
    maxsplit=-1,  
    name=None  
)

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

Let N be the size of input (typically N will be the batch size). Split each element of input based on sep and return a SparseTensor or RaggedTensor containing the split tokens. Empty tokens are ignored.

#### Example:

>>> tf.strings.split('hello world')  
<Tensor ['hello', 'world']>  
>>> tf.strings.split(['hello world', 'a b c'])  
<tf.RaggedTensor [['hello', 'world'], ['a', 'b', 'c']]>

If sep is given, consecutive delimiters are not grouped together and are deemed to delimit empty strings. For example, input of "1<>2<><>3" and sep of "<>" returns ["1", "2", "", "3"]. If sep is None or an empty string, consecutive whitespace are regarded as a single separator, and the result will contain no empty strings at the start or end if the string has leading or trailing whitespace.

Note that the above mentioned behavior matches python's str.split.

#### Args:

* **input**: A string Tensor of rank N, the strings to split. If rank(input) is not known statically, then it is assumed to be 1.
* **sep**: 0-D string Tensor, the delimiter string.
* **maxsplit**: An int. If maxsplit > 0, limit of the split of the result.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: If sep is not a string.

#### Returns:

A RaggedTensor of rank N+1, the strings split according to the delimiter.

# tf.strings.strip

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

Strip leading and trailing whitespaces from the Tensor.

### Aliases:

* tf.compat.v1.string\_strip
* tf.compat.v1.strings.strip
* tf.compat.v2.strings.strip
* tf.strings.strip

tf.strings.strip(  
    input,  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string. A string Tensor of any shape.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.substr

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

Return substrings from Tensor of strings.

### Aliases:

* tf.compat.v2.strings.substr
* tf.strings.substr

tf.strings.substr(  
    input,  
    pos,  
    len,  
    unit='BYTE',  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

For each string in the input Tensor, creates a substring starting at index pos with a total length of len.

If len defines a substring that would extend beyond the length of the input string, then as many characters as possible are used.

A negative pos indicates distance within the string backwards from the end.

If pos specifies an index which is out of range for any of the input strings, then an InvalidArgumentError is thrown.

pos and len must have the same shape, otherwise a ValueError is thrown on Op creation.

NOTE: Substr supports broadcasting up to two dimensions. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

Examples

Using scalar pos and len:

input = [b'Hello', b'World']  
position = 1  
length = 3  
  
output = [b'ell', b'orl']

Using pos and len with same shape as input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen']]  
position = [[1, 2, 3],  
            [1, 2, 3],  
            [1, 2, 3]]  
length =   [[2, 3, 4],  
            [4, 3, 2],  
            [5, 5, 5]]  
  
output = [[b'en', b'eve', b'lve'],  
          [b'hirt', b'urt', b'te'],  
          [b'ixtee', b'vente', b'hteen']]

Broadcasting pos and len onto input:

input = [[b'ten', b'eleven', b'twelve'],  
         [b'thirteen', b'fourteen', b'fifteen'],  
         [b'sixteen', b'seventeen', b'eighteen'],  
         [b'nineteen', b'twenty', b'twentyone']]  
position = [1, 2, 3]  
length =   [1, 2, 3]  
  
output = [[b'e', b'ev', b'lve'],  
          [b'h', b'ur', b'tee'],  
          [b'i', b've', b'hte'],  
          [b'i', b'en', b'nty']]

Broadcasting input onto pos and len:

input = b'thirteen'  
position = [1, 5, 7]  
length =   [3, 2, 1]  
  
output = [b'hir', b'ee', b'n']

#### Args:

* **input**: A Tensor of type string. Tensor of strings
* **pos**: A Tensor. Must be one of the following types: int32, int64. Scalar defining the position of first character in each substring
* **len**: A Tensor. Must have the same type as pos. Scalar defining the number of characters to include in each substring
* **unit**: An optional string from: "BYTE", "UTF8\_CHAR". Defaults to "BYTE". The unit that is used to create the substring. One of: "BYTE" (for defining position and length by bytes) or "UTF8\_CHAR" (for the UTF-8 encoded Unicode code points). The default is "BYTE". Results are undefined if unit=UTF8\_CHAR and the input strings do not contain structurally valid UTF-8.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.to\_hash\_bucket

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

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v2.strings.to\_hash\_bucket
* tf.strings.to\_hash\_bucket

tf.strings.to\_hash\_bucket(  
    input,  
    num\_buckets,  
    name=None  
)

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

The hash function is deterministic on the content of the string within the process.

Note that the hash function may change from time to time. This functionality will be deprecated and it's recommended to use tf.strings.to\_hash\_bucket\_fast() or tf.strings.to\_hash\_bucket\_strong().

#### Args:

* **input**: A Tensor of type string.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_hash\_bucket\_fast

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_hash_bucket_fast#used_in_the_guide)

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v1.string\_to\_hash\_bucket\_fast
* tf.compat.v1.strings.to\_hash\_bucket\_fast
* tf.compat.v2.strings.to\_hash\_bucket\_fast
* tf.strings.to\_hash\_bucket\_fast

tf.strings.to\_hash\_bucket\_fast(  
    input,  
    num\_buckets,  
    name=None  
)

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

### Used in the guide:

* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

The hash function is deterministic on the content of the string within the process and will never change. However, it is not suitable for cryptography. This function may be used when CPU time is scarce and inputs are trusted or unimportant. There is a risk of adversaries constructing inputs that all hash to the same bucket. To prevent this problem, use a strong hash function withtf.string\_to\_hash\_bucket\_strong.

#### Args:

* **input**: A Tensor of type string. The strings to assign a hash bucket.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_hash\_bucket\_strong

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

Converts each string in the input Tensor to its hash mod by a number of buckets.

### Aliases:

* tf.compat.v1.string\_to\_hash\_bucket\_strong
* tf.compat.v1.strings.to\_hash\_bucket\_strong
* tf.compat.v2.strings.to\_hash\_bucket\_strong
* tf.strings.to\_hash\_bucket\_strong

tf.strings.to\_hash\_bucket\_strong(  
    input,  
    num\_buckets,  
    key,  
    name=None  
)

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

The hash function is deterministic on the content of the string within the process. The hash function is a keyed hash function, where attribute key defines the key of the hash function. key is an array of 2 elements.

A strong hash is important when inputs may be malicious, e.g. URLs with additional components. Adversaries could try to make their inputs hash to the same bucket for a denial-of-service attack or to skew the results. A strong hash can be used to make it difficult to find inputs with a skewed hash value distribution over buckets. This requires that the hash function is seeded by a high-entropy (random) "key" unknown to the adversary.

The additional robustness comes at a cost of roughly 4x higher compute time than tf.string\_to\_hash\_bucket\_fast.

#### Args:

* **input**: A Tensor of type string. The strings to assign a hash bucket.
* **num\_buckets**: An int that is >= 1. The number of buckets.
* **key**: A list of ints. The key used to seed the hash function, passed as a list of two uint64 elements.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int64.

# tf.strings.to\_number

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_number#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_number#aliases)
* [Used in the guide:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/strings/to_number#used_in_the_guide)

Converts each string in the input Tensor to the specified numeric type.

### Aliases:

* tf.compat.v2.strings.to\_number
* tf.strings.to\_number

tf.strings.to\_number(  
    input,  
    out\_type=tf.dtypes.float32,  
    name=None  
)

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

### Used in the guide:

* [Using the SavedModel format](https://www.tensorflow.org/beta/guide/saved_model)

(Note that int32 overflow results in an error while float overflow results in a rounded value.)

#### Args:

* **input**: A Tensor of type string.
* **out\_type**: An optional [tf.DType](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/dtypes/DType) from: tf.float32, tf.float64, tf.int32, tf.int64. Defaults to [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32). The numeric type to interpret each string in string\_tensor as.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type out\_type.

# tf.strings.unicode\_decode

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

Decodes each string in input into a sequence of Unicode code points.

### Aliases:

* tf.compat.v1.strings.unicode\_decode
* tf.compat.v2.strings.unicode\_decode
* tf.strings.unicode\_decode

tf.strings.unicode\_decode(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

result[i1...iN, j] is the Unicode codepoint for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'; and in place of C0 control characters in input when replace\_control\_characters=True.
* **replace\_control\_characters**: Whether to replace the C0 control characters (U+0000 - U+001F) with the replacement\_char.
* **name**: A name for the operation (optional).

#### Returns:

A N+1 dimensional int32 tensor with shape [D1...DN, (num\_chars)]. The returned tensor is a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) if input is a scalar, or a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> tf.strings.unicode\_decode(input, 'UTF-8').tolist()  
[[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]

# tf.strings.unicode\_decode\_with\_offsets

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

Decodes each string into a sequence of code points with start offsets.

### Aliases:

* tf.compat.v1.strings.unicode\_decode\_with\_offsets
* tf.compat.v2.strings.unicode\_decode\_with\_offsets
* tf.strings.unicode\_decode\_with\_offsets

tf.strings.unicode\_decode\_with\_offsets(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

This op is similar to tf.strings.decode(...), but it also returns the start offset for each character in its respective string. This information can be used to align the characters with the original byte sequence.

Returns a tuple (codepoints, start\_offsets) where:

* codepoints[i1...iN, j] is the Unicode codepoint for the jth character in input[i1...iN], when decoded using input\_encoding.
* start\_offsets[i1...iN, j] is the start byte offset for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'; and in place of C0 control characters in input when replace\_control\_characters=True.
* **replace\_control\_characters**: Whether to replace the C0 control characters (U+0000 - U+001F) with the replacement\_char.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of N+1 dimensional tensors (codepoints, start\_offsets).

* codepoints is an int32 tensor with shape [D1...DN, (num\_chars)].
* offsets is an int64 tensor with shape [D1...DN, (num\_chars)].

The returned tensors are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor)s if input is a scalar, or [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor)s otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> result = tf.strings.unicode\_decode\_with\_offsets(input, 'UTF-8')  
>>> result[0].tolist()  # codepoints  
[[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]  
>>> result[1].tolist()  # offsets  
[0, 1, 3, 5, 6, 7, 8, 9, 10], [0]]

# tf.strings.unicode\_encode

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

Encodes each sequence of Unicode code points in input into a string.

### Aliases:

* tf.compat.v1.strings.unicode\_encode
* tf.compat.v2.strings.unicode\_encode
* tf.strings.unicode\_encode

tf.strings.unicode\_encode(  
    input,  
    output\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

result[i1...iN] is the string formed by concatenating the Unicode codepoints input[1...iN, :], encoded using output\_encoding.

#### Args:

* **input**: An N+1 dimensional potentially ragged integer tensor with shape [D1...DN, num\_chars].
* **output\_encoding**: Unicode encoding that should be used to encode each codepoint sequence. Can be "UTF-8", "UTF-16-BE", or "UTF-32-BE".
* **errors**: Specifies the response when an invalid codepoint is encountered (optional). One of: \* 'replace': Replace invalid codepoint with the replacement\_char. (default) \* 'ignore': Skip invalid codepoints. \* 'strict': Raise an exception for any invalid codepoint.
* **replacement\_char**: The replacement character codepoint to be used in place of any invalid input when errors='replace'. Any valid unicode codepoint may be used. The default value is the default unicode replacement character which is 0xFFFD (U+65533).
* **name**: A name for the operation (optional).

#### Returns:

A N dimensional string tensor with shape [D1...DN].

#### Example:

  >>> input = [[71, 246, 246, 100, 110, 105, 103, 104, 116], [128522]]  
  >>> unicode\_encode(input, 'UTF-8')  
  ['G\xc3\xb6\xc3\xb6dnight', '\xf0\x9f\x98\x8a']

# tf.strings.unicode\_script

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

Determine the script codes of a given tensor of Unicode integer code points.

### Aliases:

* tf.compat.v1.strings.unicode\_script
* tf.compat.v2.strings.unicode\_script
* tf.strings.unicode\_script

tf.strings.unicode\_script(  
    input,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

This operation converts Unicode code points to script codes corresponding to each code point. Script codes correspond to International Components for Unicode (ICU) UScriptCode values. See http://icu-project.org/apiref/icu4c/uscript\_8h.html. Returns -1 (USCRIPT\_INVALID\_CODE) for invalid codepoints. Output shape will match input shape.

#### Args:

* **input**: A Tensor of type int32. A Tensor of int32 Unicode code points.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type int32.

# tf.strings.unicode\_split

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

Splits each string in input into a sequence of Unicode code points.

### Aliases:

* tf.compat.v1.strings.unicode\_split
* tf.compat.v2.strings.unicode\_split
* tf.strings.unicode\_split

tf.strings.unicode\_split(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

result[i1...iN, j] is the substring of input[i1...iN] that encodes its jth character, when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'.
* **name**: A name for the operation (optional).

#### Returns:

A N+1 dimensional int32 tensor with shape [D1...DN, (num\_chars)]. The returned tensor is a [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) if input is a scalar, or a [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor) otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> tf.strings.unicode\_split(input, 'UTF-8').tolist()  
[['G', '\xc3\xb6', '\xc3\xb6', 'd', 'n', 'i', 'g', 'h', 't'],  
 ['\xf0\x9f\x98\x8a']]

# tf.strings.unicode\_split\_with\_offsets

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

Splits each string into a sequence of code points with start offsets.

### Aliases:

* tf.compat.v1.strings.unicode\_split\_with\_offsets
* tf.compat.v2.strings.unicode\_split\_with\_offsets
* tf.strings.unicode\_split\_with\_offsets

tf.strings.unicode\_split\_with\_offsets(  
    input,  
    input\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    name=None  
)

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

This op is similar to tf.strings.decode(...), but it also returns the start offset for each character in its respective string. This information can be used to align the characters with the original byte sequence.

Returns a tuple (chars, start\_offsets) where:

* chars[i1...iN, j] is the substring of input[i1...iN] that encodes its jth character, when decoded using input\_encoding.
* start\_offsets[i1...iN, j] is the start byte offset for the jth character in input[i1...iN], when decoded using input\_encoding.

#### Args:

* **input**: An N dimensional potentially ragged string tensor with shape [D1...DN]. N must be statically known.
* **input\_encoding**: String name for the unicode encoding that should be used to decode each string.
* **errors**: Specifies the response when an input string can't be converted using the indicated encoding. One of:
  + 'strict': Raise an exception for any illegal substrings.
  + 'replace': Replace illegal substrings with replacement\_char.
  + 'ignore': Skip illegal substrings.
* **replacement\_char**: The replacement codepoint to be used in place of invalid substrings in input when errors='replace'.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of N+1 dimensional tensors (codepoints, start\_offsets).

* codepoints is an int32 tensor with shape [D1...DN, (num\_chars)].
* offsets is an int64 tensor with shape [D1...DN, (num\_chars)].

The returned tensors are [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor)s if input is a scalar, or [tf.RaggedTensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/RaggedTensor)s otherwise.

#### Example:

>>> input = [s.encode('utf8') for s in (u'G\xf6\xf6dnight', u'\U0001f60a')]  
>>> result = tf.strings.unicode\_split\_with\_offsets(input, 'UTF-8')  
>>> result[0].tolist()  # character substrings  
[['G', '\xc3\xb6', '\xc3\xb6', 'd', 'n', 'i', 'g', 'h', 't'],  
 ['\xf0\x9f\x98\x8a']]  
>>> result[1].tolist()  # offsets  
[0, 1, 3, 5, 6, 7, 8, 9, 10], [0]]

# tf.strings.unicode\_transcode

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

Transcode the input text from a source encoding to a destination encoding.

### Aliases:

* tf.compat.v1.strings.unicode\_transcode
* tf.compat.v2.strings.unicode\_transcode
* tf.strings.unicode\_transcode

tf.strings.unicode\_transcode(  
    input,  
    input\_encoding,  
    output\_encoding,  
    errors='replace',  
    replacement\_char=65533,  
    replace\_control\_characters=False,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

The input is a string tensor of any shape. The output is a string tensor of the same shape containing the transcoded strings. Output strings are always valid unicode. If the input contains invalid encoding positions, the errors attribute sets the policy for how to deal with them. If the default error-handling policy is used, invalid formatting will be substituted in the output by the replacement\_char. If the errors policy is to ignore, any invalid encoding positions in the input are skipped and not included in the output. If it set to strict then any invalid formatting will result in an InvalidArgument error.

This operation can be used with output\_encoding = input\_encoding to enforce correct formatting for inputs even if they are already in the desired encoding.

If the input is prefixed by a Byte Order Mark needed to determine encoding (e.g. if the encoding is UTF-16 and the BOM indicates big-endian), then that BOM will be consumed and not emitted into the output. If the input encoding is marked with an explicit endianness (e.g. UTF-16-BE), then the BOM is interpreted as a non-breaking-space and is preserved in the output (including always for UTF-8).

The end result is that if the input is marked as an explicit endianness the transcoding is faithful to all codepoints in the source. If it is not marked with an explicit endianness, the BOM is not considered part of the string itself but as metadata, and so is not preserved in the output.

#### Args:

* **input**: A Tensor of type string. The text to be processed. Can have any shape.
* **input\_encoding**: A string. Text encoding of the input strings. This is any of the encodings supported by ICU ucnv algorithmic converters. Examples: "UTF-16", "US ASCII", "UTF-8".
* **output\_encoding**: A string from: "UTF-8", "UTF-16-BE", "UTF-32-BE". The unicode encoding to use in the output. Must be one of "UTF-8", "UTF-16-BE", "UTF-32-BE". Multi-byte encodings will be big-endian.
* **errors**: An optional string from: "strict", "replace", "ignore". Defaults to "replace". Error handling policy when there is invalid formatting found in the input. The value of 'strict' will cause the operation to produce a InvalidArgument error on any invalid input formatting. A value of 'replace' (the default) will cause the operation to replace any invalid formatting in the input with the replacement\_char codepoint. A value of 'ignore' will cause the operation to skip any invalid formatting in the input and produce no corresponding output character.
* **replacement\_char**: An optional int. Defaults to 65533. The replacement character codepoint to be used in place of any invalid formatting in the input when errors='replace'. Any valid unicode codepoint may be used. The default value is the default unicode replacement character is 0xFFFD or U+65533.)

Note that for UTF-8, passing a replacement character expressible in 1 byte, such as ' ', will preserve string alignment to the source since invalid bytes will be replaced with a 1-byte replacement. For UTF-16-BE and UTF-16-LE, any 1 or 2 byte replacement character will preserve byte alignment to the source.

* **replace\_control\_characters**: An optional bool. Defaults to False. Whether to replace the C0 control characters (00-1F) with the replacement\_char. Default is false.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.strings.upper

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

TODO: add doc.

### Aliases:

* tf.compat.v1.strings.upper
* tf.compat.v2.strings.upper
* tf.strings.upper

tf.strings.upper(  
    input,  
    encoding='',  
    name=None  
)

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

#### Args:

* **input**: A Tensor of type string.
* **encoding**: An optional string. Defaults to "".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

Module: tf.summary

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

Operations for writing summary data, for use in analysis and visualization.

Aliases:

* Module tf.compat.v2.summary
* Module tf.summary

Defined in [summary/\_tf/summary/\_\_init\_\_.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/summary/_tf/summary/__init__.py).

The [tf.summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary) module provides APIs for writing summary data. This data can be visualized in TensorBoard, the visualization toolkit that comes with TensorFlow. See the [TensorBoard website](https://www.tensorflow.org/tensorboard) for more detailed tutorials about how to use these APIs, or some quick examples below.

Example usage with eager execution, the default in TF 2.0:

writer = tf.summary.create\_file\_writer("/tmp/mylogs")  
with writer.as\_default():  
  for step in range(100):  
    # other model code would go here  
    tf.summary.scalar("my\_metric", 0.5, step=step)  
    writer.flush()

Example usage with [tf.function](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/function) graph execution:

writer = tf.summary.create\_file\_writer("/tmp/mylogs")  
  
@tf.function  
def my\_func(step):  
  # other model code would go here  
  with writer.as\_default():  
    tf.summary.scalar("my\_metric", 0.5, step=step)  
  
for step in range(100):  
  my\_func(step)  
  writer.flush()

Example usage with legacy TF 1.x graph execution:

with tf.compat.v1.Graph().as\_default():  
  step = tf.Variable(0, dtype=tf.int64)  
  step\_update = step.assign\_add(1)  
  writer = tf.summary.create\_file\_writer("/tmp/mylogs")  
  with writer.as\_default():  
    tf.summary.scalar("my\_metric", 0.5, step=step)  
  all\_summary\_ops = tf.compat.v1.summary.all\_v2\_summary\_ops()  
  writer\_flush = writer.flush()  
  
  sess = tf.compat.v1.Session()  
  sess.run([writer.init(), step.initializer])  
  for i in range(100):  
    sess.run(all\_summary\_ops)  
    sess.run(step\_update)  
    sess.run(writer\_flush)

Modules

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

Classes

[class SummaryWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter): Interface representing a stateful summary writer object.

Functions

[audio(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/audio): Write an audio summary.

[create\_file\_writer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/create_file_writer): Creates a summary file writer for the given log directory.

[create\_noop\_writer(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/create_noop_writer): Returns a summary writer that does nothing.

[flush(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/flush): Forces summary writer to send any buffered data to storage.

[histogram(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/histogram): Write a histogram summary.

[image(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/image): Write an image summary.

[record\_if(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/record_if): Sets summary recording on or off per the provided boolean value.

[scalar(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/scalar): Write a scalar summary.

[text(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/text): Write a text summary.

[trace\_export(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/trace_export): Stops and exports the active trace as a Summary and/or profile file.

[trace\_off(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/trace_off): Stops the current trace and discards any collected information.

[trace\_on(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/trace_on): Starts a trace to record computation graphs and profiling information.

[write(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/write): Writes a generic summary to the default SummaryWriter if one exists.

# tf.summary.audio

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

Write an audio summary.

### Aliases:

* tf.compat.v2.summary.audio
* tf.summary.audio

tf.summary.audio(  
    name,  
    data,  
    sample\_rate,  
    step=None,  
    max\_outputs=3,  
    encoding=None,  
    description=None  
)

Defined in [plugins/audio/summary\_v2.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/audio/summary_v2.py).

#### Arguments:

* **name**: A name for this summary. The summary tag used for TensorBoard will be this name prefixed by any active name scopes.
* **data**: A Tensor representing audio data with shape [k, t, c], where k is the number of audio clips, t is the number of frames, and c is the number of channels. Elements should be floating-point values in [-1.0, 1.0]. Any of the dimensions may be statically unknown (i.e., None).
* **sample\_rate**: An int or rank-0 int32 Tensor that represents the sample rate, in Hz. Must be positive.
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **max\_outputs**: Optional int or rank-0 integer Tensor. At most this many audio clips will be emitted at each step. When more than max\_outputs many clips are provided, the first max\_outputs many clips will be used and the rest silently discarded.
* **encoding**: Optional constant str for the desired encoding. Only "wav" is currently supported, but this is not guaranteed to remain the default, so if you want "wav" in particular, set this explicitly.
* **description**: Optional long-form description for this summary, as a constant str. Markdown is supported. Defaults to empty.

#### Returns:

True on success, or false if no summary was emitted because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

# tf.summary.create\_file\_writer

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

Creates a summary file writer for the given log directory.

### Aliases:

* tf.compat.v2.summary.create\_file\_writer
* tf.summary.create\_file\_writer

tf.summary.create\_file\_writer(  
    logdir,  
    max\_queue=None,  
    flush\_millis=None,  
    filename\_suffix=None,  
    name=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

#### Args:

* **logdir**: a string specifying the directory in which to write an event file.
* **max\_queue**: the largest number of summaries to keep in a queue; will flush once the queue gets bigger than this. Defaults to 10.
* **flush\_millis**: the largest interval between flushes. Defaults to 120,000.
* **filename\_suffix**: optional suffix for the event file name. Defaults to .v2.
* **name**: a name for the op that creates the writer.

#### Returns:

A SummaryWriter object.

tf.summary.create\_noop\_writer

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

Returns a summary writer that does nothing.

Aliases:

* tf.compat.v2.summary.create\_noop\_writer
* tf.summary.create\_noop\_writer

tf.summary.create\_noop\_writer()

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This is useful as a placeholder in code that expects a context manager.

# tf.summary.flush

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

Forces summary writer to send any buffered data to storage.

### Aliases:

* tf.compat.v2.summary.flush
* tf.summary.flush

tf.summary.flush(  
    writer=None,  
    name=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This operation blocks until that finishes.

#### Args:

* **writer**: The [tf.summary.SummaryWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter) resource to flush. The thread default will be used if this parameter is None. Otherwise a [tf.no\_op](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/no_op) is returned.
* **name**: A name for the operation (optional).

#### Returns:

The created [tf.Operation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Operation).

# tf.summary.histogram

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

Write a histogram summary.

### Aliases:

* tf.compat.v2.summary.histogram
* tf.summary.histogram

tf.summary.histogram(  
    name,  
    data,  
    step=None,  
    buckets=None,  
    description=None  
)

Defined in [plugins/histogram/summary\_v2.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/histogram/summary_v2.py).

#### Arguments:

* **name**: A name for this summary. The summary tag used for TensorBoard will be this name prefixed by any active name scopes.
* **data**: A Tensor of any shape. Must be castable to float64.
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **buckets**: Optional positive int. The output will have this many buckets, except in two edge cases. If there is no data, then there are no buckets. If there is data but all points have the same value, then there is one bucket whose left and right endpoints are the same.
* **description**: Optional long-form description for this summary, as a constant str. Markdown is supported. Defaults to empty.

#### Returns:

True on success, or false if no summary was emitted because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

# tf.summary.image

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

Write an image summary.

### Aliases:

* tf.compat.v2.summary.image
* tf.summary.image

tf.summary.image(  
    name,  
    data,  
    step=None,  
    max\_outputs=3,  
    description=None  
)

Defined in [plugins/image/summary\_v2.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/image/summary_v2.py).

#### Arguments:

* **name**: A name for this summary. The summary tag used for TensorBoard will be this name prefixed by any active name scopes.
* **data**: A Tensor representing pixel data with shape [k, h, w, c], where k is the number of images, h and w are the height and width of the images, and c is the number of channels, which should be 1, 2, 3, or 4 (grayscale, grayscale with alpha, RGB, RGBA). Any of the dimensions may be statically unknown (i.e., None). Floating point data will be clipped to the range [0,1).
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **max\_outputs**: Optional int or rank-0 integer Tensor. At most this many images will be emitted at each step. When more than max\_outputs many images are provided, the first max\_outputs many images will be used and the rest silently discarded.
* **description**: Optional long-form description for this summary, as a constant str. Markdown is supported. Defaults to empty.

#### Returns:

True on success, or false if no summary was emitted because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

# tf.summary.record\_if

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

Sets summary recording on or off per the provided boolean value.

### Aliases:

* tf.compat.v2.summary.record\_if
* tf.summary.record\_if

tf.summary.record\_if(condition)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

The provided value can be a python boolean, a scalar boolean Tensor, or or a callable providing such a value; if a callable is passed it will be invoked on-demand to determine whether summary writing will occur.

#### Args:

* **condition**: can be True, False, a bool Tensor, or a callable providing such.

#### Yields:

Returns a context manager that sets this value on enter and restores the previous value on exit.

# tf.summary.scalar

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

Write a scalar summary.

### Aliases:

* tf.compat.v2.summary.scalar
* tf.summary.scalar

tf.summary.scalar(  
    name,  
    data,  
    step=None,  
    description=None  
)

Defined in [plugins/scalar/summary\_v2.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/scalar/summary_v2.py).

#### Arguments:

* **name**: A name for this summary. The summary tag used for TensorBoard will be this name prefixed by any active name scopes.
* **data**: A real numeric scalar value, convertible to a float32 Tensor.
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **description**: Optional long-form description for this summary, as a constant str. Markdown is supported. Defaults to empty.

#### Returns:

True on success, or false if no summary was written because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

tf.summary.SummaryWriter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter#top_of_page)
* [Class SummaryWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter#class_summarywriter)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter#aliases)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter#methods)
  + [as\_default](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/SummaryWriter#as_default)

Class SummaryWriter

Interface representing a stateful summary writer object.

Aliases:

* Class tf.compat.v2.summary.SummaryWriter
* Class tf.summary.SummaryWriter

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

Methods

as\_default

as\_default()

Returns a context manager that enables summary writing.

close

close()

Flushes and closes the summary writer.

flush

flush()

Flushes any buffered data.

init

init()

Initializes the summary writer.

set\_as\_default

set\_as\_default()

Enables this summary writer for the current thread.

# tf.summary.text

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

Write a text summary.

### Aliases:

* tf.compat.v2.summary.text
* tf.summary.text

tf.summary.text(  
    name,  
    data,  
    step=None,  
    description=None  
)

Defined in [plugins/text/summary\_v2.py](https://github.com/tensorflow/tensorboard/tree/master/tensorboard/plugins/text/summary_v2.py).

#### Arguments:

* **name**: A name for this summary. The summary tag used for TensorBoard will be this name prefixed by any active name scopes.
* **data**: A UTF-8 string tensor value.
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **description**: Optional long-form description for this summary, as a constant str. Markdown is supported. Defaults to empty.

#### Returns:

True on success, or false if no summary was emitted because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

# tf.summary.trace\_export

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

Stops and exports the active trace as a Summary and/or profile file.

### Aliases:

* tf.compat.v2.summary.trace\_export
* tf.summary.trace\_export

tf.summary.trace\_export(  
    name,  
    step=None,  
    profiler\_outdir=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

Stops the trace and exports all metadata collected during the trace to the default SummaryWriter, if one has been set.

#### Args:

* **name**: A name for the summary to be written.
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **profiler\_outdir**: Output directory for profiler. It is required when profiler is enabled when trace was started. Otherwise, it is ignored.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

tf.summary.trace\_off

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

Stops the current trace and discards any collected information.

Aliases:

* tf.compat.v2.summary.trace\_off
* tf.summary.trace\_off

tf.summary.trace\_off()

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

# tf.summary.trace\_on

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

Starts a trace to record computation graphs and profiling information.

### Aliases:

* tf.compat.v2.summary.trace\_on
* tf.summary.trace\_on

tf.summary.trace\_on(  
    graph=True,  
    profiler=False  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

Must be invoked in eager mode.

When enabled, TensorFlow runtime will collection information that can later be exported and consumed by TensorBoard. The trace is activated across the entire TensorFlow runtime and affects all threads of execution.

To stop the trace and export the collected information, use [tf.summary.trace\_export](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/trace_export). To stop the trace without exporting, use [tf.summary.trace\_off](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/trace_off).

#### Args:

* **graph**: If True, enables collection of executed graphs. It includes ones from tf.function invocation and ones from the legacy graph mode. The default is True.
* **profiler**: If True, enables the advanced profiler. Enabling profiler implicitly enables the graph collection. The profiler may incur a high memory overhead. The default is False.

# tf.summary.write

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

Writes a generic summary to the default SummaryWriter if one exists.

### Aliases:

* tf.compat.v2.summary.write
* tf.summary.write

tf.summary.write(  
    tag,  
    tensor,  
    step=None,  
    metadata=None,  
    name=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This exists primarily to support the definition of type-specific summary ops like scalar() and image(), and is not intended for direct use unless defining a new type-specific summary op.

#### Args:

* **tag**: string tag used to identify the summary (e.g. in TensorBoard), usually generated with tf.summary.summary\_scope
* **tensor**: the Tensor holding the summary data to write
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **metadata**: Optional SummaryMetadata, as a proto or serialized bytes
* **name**: Optional string name for this op.

#### Returns:

True on success, or false if no summary was written because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

Module: tf.summary.experimental

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental#aliases)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental#functions)

Public API for tf.summary.experimental namespace.

Aliases:

* Module tf.compat.v2.summary.experimental
* Module tf.summary.experimental

Functions

[get\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental/get_step): Returns the default summary step for the current thread.

[set\_step(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental/set_step): Sets the default summary step for the current thread.

[summary\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental/summary_scope): Experimental context manager for use when defining a custom summary op.

[write\_raw\_pb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/summary/experimental/write_raw_pb): Writes a summary using raw [tf.compat.v1.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffers.

# tf.summary.experimental.get\_step

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

Returns the default summary step for the current thread.

### Aliases:

* tf.compat.v2.summary.experimental.get\_step
* tf.summary.experimental.get\_step

tf.summary.experimental.get\_step()

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

#### Returns:

The step set by tf.summary.experimental.set\_step() if one has been set, otherwise None.

# tf.summary.experimental.set\_step

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

Sets the default summary step for the current thread.

### Aliases:

* tf.compat.v2.summary.experimental.set\_step
* tf.summary.experimental.set\_step

tf.summary.experimental.set\_step(step)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

For convenience, this function sets a default value for the step parameter used in summary-writing functions elsewhere in the API so that it need not be explicitly passed in every such invocation. The value can be a constant or a variable, and can be retrieved via tf.summary.experimental.get\_step().

**Note:** when using this with @tf.functions, the step value will be captured at the time the function is traced, so changes to the step outside the function will not be reflected inside the function unless using a [**tf.Variable**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable)step.

#### Args:

* **step**: An int64-castable default step value, or None to unset.

# tf.summary.experimental.summary\_scope

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

Experimental context manager for use when defining a custom summary op.

### Aliases:

* tf.compat.v2.summary.experimental.summary\_scope
* tf.summary.experimental.summary\_scope

tf.summary.experimental.summary\_scope(  
    name,  
    default\_name='summary',  
    values=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This behaves similarly to [tf.name\_scope](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/name_scope), except that it returns a generated summary tag in addition to the scope name. The tag is structurally similar to the scope name - derived from the user-provided name, prefixed with enclosing name scopes if any - but we relax the constraint that it be uniquified, as well as the character set limitation (so the user-provided name can contain characters not legal for scope names; in the scope name these are removed).

This makes the summary tag more predictable and consistent for the user.

For example, to define a new summary op called my\_op:

def my\_op(name, my\_value, step):  
  with tf.summary.summary\_scope(name, "MyOp", [my\_value]) as (tag, scope):  
    my\_value = tf.convert\_to\_tensor(my\_value)  
    return tf.summary.write(tag, my\_value, step=step)

#### Args:

* **name**: string name for the summary.
* **default\_name**: Optional; if provided, used as default name of the summary.
* **values**: Optional; passed as values parameter to name\_scope.

#### Yields:

A tuple (tag, scope) as described above.

# tf.summary.experimental.write\_raw\_pb

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

Writes a summary using raw [tf.compat.v1.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffers.

### Aliases:

* tf.compat.v2.summary.experimental.write\_raw\_pb
* tf.summary.experimental.write\_raw\_pb

tf.summary.experimental.write\_raw\_pb(  
    tensor,  
    step=None,  
    name=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

Experimental: this exists to support the usage of V1-style manual summary writing (via the construction of a [tf.compat.v1.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffer) with the V2 summary writing API.

#### Args:

* **tensor**: the string Tensor holding one or more serialized Summary protobufs
* **step**: Explicit int64-castable monotonic step value for this summary. If omitted, this defaults to tf.summary.experimental.get\_step(), which must not be None.
* **name**: Optional string name for this op.

#### Returns:

True on success, or false if no summary was written because no default summary writer was available.

#### Raises:

* **ValueError**: if a default writer exists, but no step was provided andtf.summary.experimental.get\_step() is None.

Module: tf.compat.v1.summary

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

Operations for writing summary data, for use in analysis and visualization.

See the [Summaries and TensorBoard](https://www.tensorflow.org/guide/summaries_and_tensorboard) guide.

Classes

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

[class FileWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter): Writes Summary protocol buffers to event files.

[class FileWriterCache](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache): Cache for file writers.

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

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

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

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

Functions

[all\_v2\_summary\_ops(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/all_v2_summary_ops): Returns all V2-style summary ops defined in the current default graph.

[audio(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/audio): Outputs a Summary protocol buffer with audio.

[get\_summary\_description(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/get_summary_description): Given a TensorSummary node\_def, retrieve its SummaryDescription.

[histogram(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/histogram): Outputs a Summary protocol buffer with a histogram.

[image(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/image): Outputs a Summary protocol buffer with images.

[initialize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/initialize): Initializes summary writing for graph execution mode.

[merge(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/merge): Merges summaries.

[merge\_all(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/merge_all): Merges all summaries collected in the default graph.

[scalar(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/scalar): Outputs a Summary protocol buffer containing a single scalar value.

[tensor\_summary(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/tensor_summary): Outputs a Summary protocol buffer with a serialized tensor.proto.

[text(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/text): Summarizes textual data.

# tf.compat.v1.summary.all\_v2\_summary\_ops

Returns all V2-style summary ops defined in the current default graph.

tf.compat.v1.summary.all\_v2\_summary\_ops()

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This includes ops from TF 2.0 tf.summary and TF 1.x tf.contrib.summary (except for tf.contrib.summary.graph and tf.contrib.summary.import\_event), but does not include TF 1.x tf.summary ops.

#### Returns:

List of summary ops, or None if called under eager execution.

# tf.compat.v1.summary.audio

Outputs a Summary protocol buffer with audio.

tf.compat.v1.summary.audio(  
    name,  
    tensor,  
    sample\_rate,  
    max\_outputs=3,  
    collections=None,  
    family=None  
)

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

The summary has up to max\_outputs summary values containing audio. The audio is built from tensor which must be 3-D with shape [batch\_size, frames, channels] or 2-D with shape [batch\_size, frames]. The values are assumed to be in the range of [-1.0, 1.0] with a sample rate of sample\_rate.

The tag in the outputted Summary.Value protobufs is generated based on the name, with a suffix depending on the max\_outputs setting:

* If max\_outputs is 1, the summary value tag is 'name/audio'.
* If max\_outputs is greater than 1, the summary value tags are generated sequentially as 'name/audio/0', 'name/audio/1', etc

#### Args:

* **name**: A name for the generated node. Will also serve as a series name in TensorBoard.
* **tensor**: A 3-D float32 Tensor of shape [batch\_size, frames, channels] or a 2-D float32 Tensor of shape [batch\_size, frames].
* **sample\_rate**: A Scalar float32 Tensor indicating the sample rate of the signal in hertz.
* **max\_outputs**: Max number of batch elements to generate audio for.
* **collections**: Optional list of ops.GraphKeys. The collections to add the summary to. Defaults to [\_ops.GraphKeys.SUMMARIES]
* **family**: Optional; if provided, used as the prefix of the summary tag name, which controls the tab name used for display on Tensorboard.

#### Returns:

A scalar Tensor of type string. The serialized Summary protocol buffer.

# tf.compat.v1.summary.FileWriter

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter#top_of_page)
* [Class FileWriter](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter#class_filewriter)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter#methods)
  + [\_\_enter\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriter#__enter__)

## Class FileWriter

Writes Summary protocol buffers to event files.

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

The FileWriter class provides a mechanism to create an event file in a given directory and add summaries and events to it. The class updates the file contents asynchronously. This allows a training program to call methods to add data to the file directly from the training loop, without slowing down training.

When constructed with a [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) parameter, a FileWriter instead forms a compatibility layer over new graph-based summaries (tf.contrib.summary) to facilitate the use of new summary writing with pre-existing code that expects a FileWriter instance.

## \_\_init\_\_

\_\_init\_\_(  
    logdir,  
    graph=None,  
    max\_queue=10,  
    flush\_secs=120,  
    graph\_def=None,  
    filename\_suffix=None,  
    session=None  
)

Creates a FileWriter, optionally shared within the given session.

Typically, constructing a file writer creates a new event file in logdir. This event file will contain Event protocol buffers constructed when you call one of the following functions: add\_summary(), add\_session\_log(), add\_event(), or add\_graph().

If you pass a Graph to the constructor it is added to the event file. (This is equivalent to calling add\_graph() later).

TensorBoard will pick the graph from the file and display it graphically so you can interactively explore the graph you built. You will usually pass the graph from the session in which you launched it:

...create a graph...  
# Launch the graph in a session.  
sess = tf.compat.v1.Session()  
# Create a summary writer, add the 'graph' to the event file.  
writer = tf.compat.v1.summary.FileWriter(<some-directory>, sess.graph)

The session argument to the constructor makes the returned FileWriter a compatibility layer over new graph-based summaries (tf.contrib.summary). Crucially, this means the underlying writer resource and events file will be shared with any other FileWriter using the same session and logdir, and with any tf.contrib.summary.SummaryWriter in this session using the the same shared resource name (which by default scoped to the logdir). If no such resource exists, one will be created using the remaining arguments to this constructor, but if one already exists those arguments are ignored. In either case, ops will be added to session.graph to control the underlying file writer resource. See tf.contrib.summary for more details.

#### Args:

* **logdir**: A string. Directory where event file will be written.
* **graph**: A Graph object, such as sess.graph.
* **max\_queue**: Integer. Size of the queue for pending events and summaries.
* **flush\_secs**: Number. How often, in seconds, to flush the pending events and summaries to disk.
* **graph\_def**: DEPRECATED: Use the graph argument instead.
* **filename\_suffix**: A string. Every event file's name is suffixed with suffix.
* **session**: A [tf.compat.v1.Session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Session) object. See details above.

#### Raises:

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

#### Eager Compatibility

FileWriter is not compatible with eager execution. To write TensorBoard summaries under eager execution, use tf.contrib.summary instead.

## Methods

### \_\_enter\_\_

\_\_enter\_\_()

Make usable with "with" statement.

### \_\_exit\_\_

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

Make usable with "with" statement.

### add\_event

add\_event(event)

Adds an event to the event file.

#### Args:

* **event**: An Event protocol buffer.

### add\_graph

add\_graph(  
    graph,  
    global\_step=None,  
    graph\_def=None  
)

Adds a Graph to the event file.

The graph described by the protocol buffer will be displayed by TensorBoard. Most users pass a graph in the constructor instead.

#### Args:

* **graph**: A Graph object, such as sess.graph.
* **global\_step**: Number. Optional global step counter to record with the graph.
* **graph\_def**: DEPRECATED. Use the graph parameter instead.

#### Raises:

* **ValueError**: If both graph and graph\_def are passed to the method.

### add\_meta\_graph

add\_meta\_graph(  
    meta\_graph\_def,  
    global\_step=None  
)

Adds a MetaGraphDef to the event file.

The MetaGraphDef allows running the given graph via saver.import\_meta\_graph().

#### Args:

* **meta\_graph\_def**: A MetaGraphDef object, often as returned bysaver.export\_meta\_graph().
* **global\_step**: Number. Optional global step counter to record with the graph.

#### Raises:

* **TypeError**: If both meta\_graph\_def is not an instance of MetaGraphDef.

### add\_run\_metadata

add\_run\_metadata(  
    run\_metadata,  
    tag,  
    global\_step=None  
)

Adds a metadata information for a single session.run() call.

#### Args:

* **run\_metadata**: A RunMetadata protobuf object.
* **tag**: The tag name for this metadata.
* **global\_step**: Number. Optional global step counter to record with the StepStats.

#### Raises:

* **ValueError**: If the provided tag was already used for this type of event.

### add\_session\_log

add\_session\_log(  
    session\_log,  
    global\_step=None  
)

Adds a SessionLog protocol buffer to the event file.

This method wraps the provided session in an Event protocol buffer and adds it to the event file.

#### Args:

* **session\_log**: A SessionLog protocol buffer.
* **global\_step**: Number. Optional global step value to record with the summary.

### add\_summary

add\_summary(  
    summary,  
    global\_step=None  
)

Adds a Summary protocol buffer to the event file.

This method wraps the provided summary in an Event protocol buffer and adds it to the event file.

You can pass the result of evaluating any summary op, using tf.Session.run or [tf.Tensor.eval](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor#eval), to this function. Alternatively, you can pass a [tf.compat.v1.Summary](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/Summary) protocol buffer that you populate with your own data. The latter is commonly done to report evaluation results in event files.

#### Args:

* **summary**: A Summary protocol buffer, optionally serialized as a string.
* **global\_step**: Number. Optional global step value to record with the summary.

### close

close()

Flushes the event file to disk and close the file.

Call this method when you do not need the summary writer anymore.

### flush

flush()

Flushes the event file to disk.

Call this method to make sure that all pending events have been written to disk.

### get\_logdir

get\_logdir()

Returns the directory where event file will be written.

### reopen

reopen()

Reopens the EventFileWriter.

Can be called after close() to add more events in the same directory. The events will go into a new events file.

Does nothing if the EventFileWriter was not closed.

# tf.compat.v1.summary.FileWriterCache

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache#top_of_page)
* [Class FileWriterCache](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache#class_filewritercache)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache#methods)
  + [clear](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache#clear)
  + [get](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/FileWriterCache#get)

## Class FileWriterCache

Cache for file writers.

Defined in [python/summary/writer/writer\_cache.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/summary/writer/writer_cache.py).

This class caches file writers, one per directory.

## Methods

### clear

@staticmethod  
clear()

Clear cached summary writers. Currently only used for unit tests.

### get

@staticmethod  
get(logdir)

Returns the FileWriter for the specified directory.

#### Args:

* **logdir**: str, name of the directory.

#### Returns:

A FileWriter.

# tf.compat.v1.summary.get\_summary\_description

Given a TensorSummary node\_def, retrieve its SummaryDescription.

tf.compat.v1.summary.get\_summary\_description(node\_def)

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

When a Summary op is instantiated, a SummaryDescription of associated metadata is stored in its NodeDef. This method retrieves the description.

#### Args:

* **node\_def**: the node\_def\_pb2.NodeDef of a TensorSummary op

#### Returns:

a summary\_pb2.SummaryDescription

#### Raises:

* **ValueError**: if the node is not a summary op.

#### Eager Compatibility

Not compatible with eager execution. To write TensorBoard summaries under eager execution, use tf.contrib.summary instead.

# tf.compat.v1.summary.histogram

Outputs a Summary protocol buffer with a histogram.

tf.compat.v1.summary.histogram(  
    name,  
    values,  
    collections=None,  
    family=None  
)

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

Adding a histogram summary makes it possible to visualize your data's distribution in TensorBoard. You can see a detailed explanation of the TensorBoard histogram dashboard [here](https://www.tensorflow.org/get_started/tensorboard_histograms).

The generated [Summary](https://www.tensorflow.org/code/tensorflow/core/framework/summary.proto) has one summary value containing a histogram for values.

This op reports an InvalidArgument error if any value is not finite.

#### Args:

* **name**: A name for the generated node. Will also serve as a series name in TensorBoard.
* **values**: A real numeric Tensor. Any shape. Values to use to build the histogram.
* **collections**: Optional list of graph collections keys. The new summary op is added to these collections. Defaults to [GraphKeys.SUMMARIES].
* **family**: Optional; if provided, used as the prefix of the summary tag name, which controls the tab name used for display on Tensorboard.

#### Returns:

A scalar Tensor of type string. The serialized Summary protocol buffer.

# tf.compat.v1.summary.image

Outputs a Summary protocol buffer with images.

tf.compat.v1.summary.image(  
    name,  
    tensor,  
    max\_outputs=3,  
    collections=None,  
    family=None  
)

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

The summary has up to max\_outputs summary values containing images. The images are built from tensor which must be 4-D with shape [batch\_size, height, width, channels] and where channels can be:

* 1: tensor is interpreted as Grayscale.
* 3: tensor is interpreted as RGB.
* 4: tensor is interpreted as RGBA.

The images have the same number of channels as the input tensor. For float input, the values are normalized one image at a time to fit in the range [0, 255]. uint8 values are unchanged. The op uses two different normalization algorithms:

* If the input values are all positive, they are rescaled so the largest one is 255.
* If any input value is negative, the values are shifted so input value 0.0 is at 127. They are then rescaled so that either the smallest value is 0, or the largest one is 255.

The tag in the outputted Summary.Value protobufs is generated based on the name, with a suffix depending on the max\_outputs setting:

* If max\_outputs is 1, the summary value tag is 'name/image'.
* If max\_outputs is greater than 1, the summary value tags are generated sequentially as 'name/image/0', 'name/image/1', etc.

#### Args:

* **name**: A name for the generated node. Will also serve as a series name in TensorBoard.
* **tensor**: A 4-D uint8 or float32 Tensor of shape [batch\_size, height, width, channels] where channels is 1, 3, or 4.
* **max\_outputs**: Max number of batch elements to generate images for.
* **collections**: Optional list of ops.GraphKeys. The collections to add the summary to. Defaults to [\_ops.GraphKeys.SUMMARIES]
* **family**: Optional; if provided, used as the prefix of the summary tag name, which controls the tab name used for display on Tensorboard.

#### Returns:

A scalar Tensor of type string. The serialized Summary protocol buffer.

# tf.compat.v1.summary.initialize

Initializes summary writing for graph execution mode.

tf.compat.v1.summary.initialize(  
    graph=None,  
    session=None  
)

Defined in [python/ops/summary\_ops\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/summary_ops_v2.py).

This operation is a no-op when executing eagerly.

This helper method provides a higher-level alternative to usingtf.contrib.summary.summary\_writer\_initializer\_op and tf.contrib.summary.graph.

Most users will also want to call [tf.compat.v1.train.create\_global\_step](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/train/create_global_step) which can happen before or after this function is called.

#### Args:

* **graph**: A [tf.Graph](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Graph) or [tf.compat.v1.GraphDef](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/GraphDef) to output to the writer. This function will not write the default graph by default. When writing to an event log file, the associated step will be zero.
* **session**: So this method can call tf.Session.run. This defaults to [tf.compat.v1.get\_default\_session](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/get_default_session).

#### Raises:

* **RuntimeError**: If the current thread has no default tf.contrib.summary.SummaryWriter.
* **ValueError**: If session wasn't passed and no default session.

# tf.compat.v1.summary.merge

Merges summaries.

tf.compat.v1.summary.merge(  
    inputs,  
    collections=None,  
    name=None  
)

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

This op creates a [Summary](https://www.tensorflow.org/code/tensorflow/core/framework/summary.proto) protocol buffer that contains the union of all the values in the input summaries.

When the Op is run, it reports an InvalidArgument error if multiple values in the summaries to merge use the same tag.

#### Args:

* **inputs**: A list of string Tensor objects containing serialized Summary protocol buffers.
* **collections**: Optional list of graph collections keys. The new summary op is added to these collections. Defaults to [].
* **name**: A name for the operation (optional).

#### Returns:

A scalar Tensor of type string. The serialized Summary protocol buffer resulting from the merging.

#### Raises:

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

#### Eager Compatibility

Not compatible with eager execution. To write TensorBoard summaries under eager execution, use tf.contrib.summary instead.

# tf.compat.v1.summary.merge\_all

Merges all summaries collected in the default graph.

tf.compat.v1.summary.merge\_all(  
    key=tf.GraphKeys.SUMMARIES,  
    scope=None,  
    name=None  
)

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

#### Args:

* **key**: GraphKey used to collect the summaries. Defaults to GraphKeys.SUMMARIES.
* **scope**: Optional scope used to filter the summary ops, using re.match

#### Returns:

If no summaries were collected, returns None. Otherwise returns a scalar Tensor of type stringcontaining the serialized Summary protocol buffer resulting from the merging.

#### Raises:

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

#### Eager Compatibility

Not compatible with eager execution. To write TensorBoard summaries under eager execution, use tf.contrib.summary instead.

# tf.compat.v1.summary.scalar

Outputs a Summary protocol buffer containing a single scalar value.

tf.compat.v1.summary.scalar(  
    name,  
    tensor,  
    collections=None,  
    family=None  
)

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

The generated Summary has a Tensor.proto containing the input Tensor.

#### Args:

* **name**: A name for the generated node. Will also serve as the series name in TensorBoard.
* **tensor**: A real numeric Tensor containing a single value.
* **collections**: Optional list of graph collections keys. The new summary op is added to these collections. Defaults to [GraphKeys.SUMMARIES].
* **family**: Optional; if provided, used as the prefix of the summary tag name, which controls the tab name used for display on Tensorboard.

#### Returns:

A scalar Tensor of type string. Which contains a Summary protobuf.

#### Raises:

* **ValueError**: If tensor has the wrong shape or type.

tf.compat.v1.summary.SummaryDescription

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/SummaryDescription#top_of_page)
* [Class SummaryDescription](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/SummaryDescription#class_summarydescription)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/SummaryDescription#properties)
  + [type\_hint](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/SummaryDescription#type_hint)

Class SummaryDescription

Defined in [core/framework/summary.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/framework/summary.proto).

Properties

type\_hint

string type\_hint

tf.compat.v1.summary.TaggedRunMetadata

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/TaggedRunMetadata#top_of_page)
* [Class TaggedRunMetadata](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/TaggedRunMetadata#class_taggedrunmetadata)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/summary/TaggedRunMetadata#properties)

Class TaggedRunMetadata

Defined in [core/util/event.proto](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/core/util/event.proto).

Properties

run\_metadata

bytes run\_metadata

tag

string tag

# tf.compat.v1.summary.tensor\_summary

Outputs a Summary protocol buffer with a serialized tensor.proto.

tf.compat.v1.summary.tensor\_summary(  
    name,  
    tensor,  
    summary\_description=None,  
    collections=None,  
    summary\_metadata=None,  
    family=None,  
    display\_name=None  
)

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

#### Args:

* **name**: A name for the generated node. If display\_name is not set, it will also serve as the tag name in TensorBoard. (In that case, the tag name will inherit tf name scopes.)
* **tensor**: A tensor of any type and shape to serialize.
* **summary\_description**: A long description of the summary sequence. Markdown is supported.
* **collections**: Optional list of graph collections keys. The new summary op is added to these collections. Defaults to [GraphKeys.SUMMARIES].
* **summary\_metadata**: Optional SummaryMetadata proto (which describes which plugins may use the summary value).
* **family**: Optional; if provided, used as the prefix of the summary tag, which controls the name used for display on TensorBoard when display\_name is not set.
* **display\_name**: A string used to name this data in TensorBoard. If this is not set, then the node name will be used instead.

#### Returns:

A scalar Tensor of type string. The serialized Summary protocol buffer.

# tf.compat.v1.summary.text

Summarizes textual data.

tf.compat.v1.summary.text(  
    name,  
    tensor,  
    collections=None  
)

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

Text data summarized via this plugin will be visible in the Text Dashboard in TensorBoard. The standard TensorBoard Text Dashboard will render markdown in the strings, and will automatically organize 1d and 2d tensors into tables. If a tensor with more than 2 dimensions is provided, a 2d subarray will be displayed along with a warning message. (Note that this behavior is not intrinsic to the text summary api, but rather to the default TensorBoard text plugin.)

#### Args:

* **name**: A name for the generated node. Will also serve as a series name in TensorBoard.
* **tensor**: a string-type Tensor to summarize.
* **collections**: Optional list of ops.GraphKeys. The collections to add the summary to. Defaults to [\_ops.GraphKeys.SUMMARIES]

#### Returns:

A TensorSummary op that is configured so that TensorBoard will recognize that it contains textual data. The TensorSummary is a scalar Tensor of type string which contains Summary protobufs.

#### Raises:

* **ValueError**: If tensor has the wrong type.

Module: tf.sysconfig

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig#top_of_page)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig#functions)
* [Other Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig#other_members)

System configuration library.

Functions

[get\_compile\_flags(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig/get_compile_flags): Get the compilation flags for custom operators.

[get\_include(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig/get_include): Get the directory containing the TensorFlow C++ header files.

[get\_lib(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig/get_lib): Get the directory containing the TensorFlow framework library.

[get\_link\_flags(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/sysconfig/get_link_flags): Get the link flags for custom operators.

Other Members

* CXX11\_ABI\_FLAG = 0
* MONOLITHIC\_BUILD = 0

# tf.sysconfig.get\_compile\_flags

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

Get the compilation flags for custom operators.

### Aliases:

* tf.compat.v1.sysconfig.get\_compile\_flags
* tf.compat.v2.sysconfig.get\_compile\_flags
* tf.sysconfig.get\_compile\_flags

tf.sysconfig.get\_compile\_flags()

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

#### Returns:

The compilation flags.

# tf.sysconfig.get\_include

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

Get the directory containing the TensorFlow C++ header files.

### Aliases:

* tf.compat.v1.sysconfig.get\_include
* tf.compat.v2.sysconfig.get\_include
* tf.sysconfig.get\_include

tf.sysconfig.get\_include()

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

#### Returns:

The directory as string.

# tf.sysconfig.get\_lib

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

Get the directory containing the TensorFlow framework library.

### Aliases:

* tf.compat.v1.sysconfig.get\_lib
* tf.compat.v2.sysconfig.get\_lib
* tf.sysconfig.get\_lib

tf.sysconfig.get\_lib()

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

#### Returns:

The directory as string.

# tf.sysconfig.get\_link\_flags

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

Get the link flags for custom operators.

### Aliases:

* tf.compat.v1.sysconfig.get\_link\_flags
* tf.compat.v2.sysconfig.get\_link\_flags
* tf.sysconfig.get\_link\_flags

tf.sysconfig.get\_link\_flags()

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

#### Returns:

The link flags.

Module: tf.compat.v1.test / tf.test

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

Testing.

See the [Testing](https://tensorflow.org/api_guides/python/test) guide.

**Note:** **tf.compat.v1.test.mock** is an alias to the python **mock** or **unittest.mock** depending on the python version.

Classes

[class Benchmark](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/Benchmark): Abstract class that provides helpers for TensorFlow benchmarks.

[class StubOutForTesting](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting): Support class for stubbing methods out for unit testing.

[class TestCase](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase): Base class for tests that need to test TensorFlow.

Functions

[assert\_equal\_graph\_def(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/assert_equal_graph_def): Asserts that two GraphDefs are (mostly) the same.

[benchmark\_config(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/benchmark_config): Returns a tf.compat.v1.ConfigProto for disabling the dependency optimizer.

[compute\_gradient(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/compute_gradient): Computes and returns the theoretical and numerical Jacobian.

[compute\_gradient\_error(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/compute_gradient_error): Computes the gradient error.

[create\_local\_cluster(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/create_local_cluster): Create and start local servers and return the associated Serverobjects.

[get\_temp\_dir(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/get_temp_dir): Returns a temporary directory for use during tests.

[gpu\_device\_name(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/gpu_device_name): Returns the name of a GPU device if available or the empty string.

[is\_built\_with\_cuda(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/is_built_with_cuda): Returns whether TensorFlow was built with CUDA (GPU) support.

[is\_gpu\_available(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/is_gpu_available): Returns whether TensorFlow can access a GPU.

[main(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/main): Runs all unit tests.

[test\_src\_dir\_path(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/test_src_dir_path): Creates an absolute test srcdir path given a relative path.

# tf.compat.v1.test.assert\_equal\_graph\_def

Asserts that two GraphDefs are (mostly) the same.

tf.compat.v1.test.assert\_equal\_graph\_def(  
    actual,  
    expected,  
    checkpoint\_v2=False,  
    hash\_table\_shared\_name=False  
)

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

Compares two GraphDef protos for equality, ignoring versions and ordering of nodes, attrs, and control inputs. Node names are used to match up nodes between the graphs, so the naming of nodes must be consistent.

#### Args:

* **actual**: The GraphDef we have.
* **expected**: The GraphDef we expected.
* **checkpoint\_v2**: boolean determining whether to ignore randomized attribute values that appear in V2 checkpoints.
* **hash\_table\_shared\_name**: boolean determining whether to ignore randomized shared\_names that appear in HashTableV2 op defs.

#### Raises:

* **AssertionError**: If the GraphDefs do not match.
* **TypeError**: If either argument is not a GraphDef.

# tf.compat.v1.test.compute\_gradient

Computes and returns the theoretical and numerical Jacobian.

tf.compat.v1.test.compute\_gradient(  
    x,  
    x\_shape,  
    y,  
    y\_shape,  
    x\_init\_value=None,  
    delta=0.001,  
    init\_targets=None,  
    extra\_feed\_dict=None  
)

Defined in [python/ops/gradient\_checker.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/gradient_checker.py).

If x or y is complex, the Jacobian will still be real but the corresponding Jacobian dimension(s) will be twice as large. This is required even if both input and output is complex since TensorFlow graphs are not necessarily holomorphic, and may have gradients not expressible as complex numbers. For example, if x is complex with shape [m] and y is complex with shape [n], each Jacobian J will have shape [m \* 2, n \* 2] with

J[:m, :n] = d(Re y)/d(Re x)  
J[:m, n:] = d(Im y)/d(Re x)  
J[m:, :n] = d(Re y)/d(Im x)  
J[m:, n:] = d(Im y)/d(Im x)

#### Args:

* **x**: a tensor or list of tensors
* **x\_shape**: the dimensions of x as a tuple or an array of ints. If x is a list, then this is the list of shapes.
* **y**: a tensor
* **y\_shape**: the dimensions of y as a tuple or an array of ints.
* **x\_init\_value**: (optional) a numpy array of the same shape as "x" representing the initial value of x. If x is a list, this should be a list of numpy arrays. If this is none, the function will pick a random tensor as the initial value.
* **delta**: (optional) the amount of perturbation.
* **init\_targets**: list of targets to run to initialize model params.
* **extra\_feed\_dict**: dict that allows fixing specified tensor values during the Jacobian calculation.

#### Returns:

Two 2-d numpy arrays representing the theoretical and numerical Jacobian for dy/dx. Each has "x\_size" rows and "y\_size" columns where "x\_size" is the number of elements in x and "y\_size" is the number of elements in y. If x is a list, returns a list of two numpy arrays.

# tf.compat.v1.test.compute\_gradient\_error

Computes the gradient error.

tf.compat.v1.test.compute\_gradient\_error(  
    x,  
    x\_shape,  
    y,  
    y\_shape,  
    x\_init\_value=None,  
    delta=0.001,  
    init\_targets=None,  
    extra\_feed\_dict=None  
)

Defined in [python/ops/gradient\_checker.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/gradient_checker.py).

Computes the maximum error for dy/dx between the computed Jacobian and the numerically estimated Jacobian.

This function will modify the tensors passed in as it adds more operations and hence changing the consumers of the operations of the input tensors.

This function adds operations to the current session. To compute the error using a particular device, such as a GPU, use the standard methods for setting a device (e.g. using with sess.graph.device() or setting a device function in the session constructor).

#### Args:

* **x**: a tensor or list of tensors
* **x\_shape**: the dimensions of x as a tuple or an array of ints. If x is a list, then this is the list of shapes.
* **y**: a tensor
* **y\_shape**: the dimensions of y as a tuple or an array of ints.
* **x\_init\_value**: (optional) a numpy array of the same shape as "x" representing the initial value of x. If x is a list, this should be a list of numpy arrays. If this is none, the function will pick a random tensor as the initial value.
* **delta**: (optional) the amount of perturbation.
* **init\_targets**: list of targets to run to initialize model params.
* **extra\_feed\_dict**: dict that allows fixing specified tensor values during the Jacobian calculation.

#### Returns:

The maximum error in between the two Jacobians.

# tf.compat.v1.test.get\_temp\_dir

Returns a temporary directory for use during tests.

tf.compat.v1.test.get\_temp\_dir()

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

There is no need to delete the directory after the test.

#### Returns:

The temporary directory.

# tf.compat.v1.test.StubOutForTesting

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting#top_of_page)
* [Class StubOutForTesting](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting#class_stuboutfortesting)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting#__init__)
* [Methods](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting#methods)
  + [CleanUp](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/test/StubOutForTesting#CleanUp)

## Class StubOutForTesting

Support class for stubbing methods out for unit testing.

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

#### Sample Usage:

You want os.path.exists() to always return true during testing.

stubs = StubOutForTesting() stubs.Set(os.path, 'exists', lambda x: 1) ... stubs.CleanUp()

The above changes os.path.exists into a lambda that returns 1. Once the ... part of the code finishes, the CleanUp() looks up the old value of os.path.exists and restores it.

## \_\_init\_\_

\_\_init\_\_()

## Methods

### CleanUp

CleanUp()

Undoes all SmartSet() & Set() calls, restoring original definitions.

### Set

Set(  
    parent,  
    child\_name,  
    new\_child  
)

In parent, replace child\_name's old definition with new\_child.

The parent could be a module when the child is a function at module scope. Or the parent could be a class when a class' method is being replaced. The named child is set to new\_child, while the prior definition is saved away for later, when UnsetAll() is called.

This method supports the case where child\_name is a staticmethod or a classmethod of parent.

#### Args:

* **parent**: The context in which the attribute child\_name is to be changed.
* **child\_name**: The name of the attribute to change.
* **new\_child**: The new value of the attribute.

### SmartSet

SmartSet(  
    obj,  
    attr\_name,  
    new\_attr  
)

Replace obj.attr\_name with new\_attr.

This method is smart and works at the module, class, and instance level while preserving proper inheritance. It will not stub out C types however unless that has been explicitly allowed by the type.

This method supports the case where attr\_name is a staticmethod or a classmethod of obj.

#### Notes:

* If obj is an instance, then it is its class that will actually be stubbed. Note that the method Set() does not do that: if obj is an instance, it (and not its class) will be stubbed.
* The stubbing is using the builtin getattr and setattr. So, the **get** and **set** will be called when stubbing (TODO: A better idea would probably be to manipulate obj.**dict** instead of getattr() and setattr()).

#### Args:

* **obj**: The object whose attributes we want to modify.
* **attr\_name**: The name of the attribute to modify.
* **new\_attr**: The new value for the attribute.

#### Raises:

* **AttributeError**: If the attribute cannot be found.

### SmartUnsetAll

SmartUnsetAll()

Reverses SmartSet() calls, restoring things to original definitions.

This method is automatically called when the StubOutForTesting() object is deleted; there is no need to call it explicitly.

It is okay to call SmartUnsetAll() repeatedly, as later calls have no effect if no SmartSet() calls have been made.

### UnsetAll

UnsetAll()

Reverses Set() calls, restoring things to their original definitions.

This method is automatically called when the StubOutForTesting() object is deleted; there is no need to call it explicitly.

It is okay to call UnsetAll() repeatedly, as later calls have no effect if no Set() calls have been made.

### \_\_enter\_\_

\_\_enter\_\_()

### \_\_exit\_\_

\_\_exit\_\_(  
    unused\_exc\_type,  
    unused\_exc\_value,  
    unused\_tb  
)

# tf.compat.v1.test.test\_src\_dir\_path

Creates an absolute test srcdir path given a relative path.

tf.compat.v1.test.test\_src\_dir\_path(relative\_path)

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

#### Args:

* **relative\_path**: a path relative to tensorflow root. e.g. "core/platform".

#### Returns:

An absolute path to the linked in runfiles.

# tf.test.assert\_equal\_graph\_def

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

Asserts that two GraphDefs are (mostly) the same.

### Aliases:

* tf.compat.v2.test.assert\_equal\_graph\_def
* tf.test.assert\_equal\_graph\_def

tf.test.assert\_equal\_graph\_def(  
    expected,  
    actual  
)

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

Compares two GraphDef protos for equality, ignoring versions and ordering of nodes, attrs, and control inputs. Node names are used to match up nodes between the graphs, so the naming of nodes must be consistent. This function ignores randomized attribute values that may appear in V2 checkpoints.

#### Args:

* **expected**: The GraphDef we expected.
* **actual**: The GraphDef we have.

#### Raises:

* **AssertionError**: If the GraphDefs do not match.
* **TypeError**: If either argument is not a GraphDef.

# tf.test.Benchmark

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

## Class Benchmark

Abstract class that provides helpers for TensorFlow benchmarks.

### Aliases:

* Class tf.compat.v1.test.Benchmark
* Class tf.compat.v2.test.Benchmark
* Class tf.test.Benchmark

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

## \_\_init\_\_

\_\_init\_\_()

## Methods

### evaluate

evaluate(tensors)

Evaluates tensors and returns numpy values.

#### Args:

* **tensors**: A Tensor or a nested list/tuple of Tensors.

#### Returns:

tensors numpy values.

### is\_abstract

@classmethod  
is\_abstract(cls)

### report\_benchmark

report\_benchmark(  
    iters=None,  
    cpu\_time=None,  
    wall\_time=None,  
    throughput=None,  
    extras=None,  
    name=None,  
    metrics=None  
)

Report a benchmark.

#### Args:

* **iters**: (optional) How many iterations were run
* **cpu\_time**: (optional) Median or mean cpu time in seconds.
* **wall\_time**: (optional) Median or mean wall time in seconds.
* **throughput**: (optional) Throughput (in MB/s)
* **extras**: (optional) Dict mapping string keys to additional benchmark info. Values may be either floats or values that are convertible to strings.
* **name**: (optional) Override the BenchmarkEntry name with name. Otherwise it is inferred from the top-level method name.
* **metrics**: (optional) A list of dict, where each dict has the keys below name (required), string, metric name value (required), double, metric value min\_value (optional), double, minimum acceptable metric value max\_value (optional), double, maximum acceptable metric value

### run\_op\_benchmark

run\_op\_benchmark(  
    sess,  
    op\_or\_tensor,  
    feed\_dict=None,  
    burn\_iters=2,  
    min\_iters=10,  
    store\_trace=False,  
    store\_memory\_usage=True,  
    name=None,  
    extras=None,  
    mbs=0  
)

Run an op or tensor in the given session. Report the results.

#### Args:

* **sess**: Session object to use for timing.
* **op\_or\_tensor**: Operation or Tensor to benchmark.
* **feed\_dict**: A dict of values to feed for each op iteration (see the feed\_dict parameter of Session.run).
* **burn\_iters**: Number of burn-in iterations to run.
* **min\_iters**: Minimum number of iterations to use for timing.
* **store\_trace**: Boolean, whether to run an extra untimed iteration and store the trace of iteration in returned extras. The trace will be stored as a string in Google Chrome trace format in the extras field "full\_trace\_chrome\_format". Note that trace will not be stored in test\_log\_pb2.TestResults proto.
* **store\_memory\_usage**: Boolean, whether to run an extra untimed iteration, calculate memory usage, and store that in extras fields.
* **name**: (optional) Override the BenchmarkEntry name with name. Otherwise it is inferred from the top-level method name.
* **extras**: (optional) Dict mapping string keys to additional benchmark info. Values may be either floats or values that are convertible to strings.
* **mbs**: (optional) The number of megabytes moved by this op, used to calculate the ops throughput.

#### Returns:

A dict containing the key-value pairs that were passed to report\_benchmark. If store\_traceoption is used, then full\_chrome\_trace\_format will be included in return dictionary even though it is not passed to report\_benchmark with extras.

# tf.test.benchmark\_config

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

Returns a tf.compat.v1.ConfigProto for disabling the dependency optimizer.

### Aliases:

* tf.compat.v1.test.benchmark\_config
* tf.compat.v2.test.benchmark\_config
* tf.test.benchmark\_config

tf.test.benchmark\_config()

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

#### Returns:

A TensorFlow ConfigProto object.

# tf.test.compute\_gradient

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

Computes the theoretical and numeric Jacobian of f.

### Aliases:

* tf.compat.v2.test.compute\_gradient
* tf.test.compute\_gradient

tf.test.compute\_gradient(  
    f,  
    x,  
    delta=0.001  
)

Defined in [python/ops/gradient\_checker\_v2.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/gradient_checker_v2.py).

With y = f(x), computes the theoretical and numeric Jacobian dy/dx.

#### Args:

* **f**: the function.
* **x**: a list arguments for the function
* **delta**: (optional) perturbation used to compute numeric Jacobian.

#### Returns:

A pair of lists, where the first is a list of 2-d numpy arrays representing the theoretical Jacobians for each argument, and the second list is the numerical ones. Each 2-d array has "x\_size" rows and "y\_size" columns where "x\_size" is the number of elements in the corresponding argument and "y\_size" is the number of elements in f(x).

#### Raises:

* **ValueError**: If result is empty but the gradient is nonzero.
* **ValueError**: If x is not list, but any other type.

#### Example:

@tf.function  
def test\_func(x):  
  return x\*x  
  
theoretical, numerical = tf.test.compute\_gradient(test\_func, [1.0])  
theoretical, numerical  
# ((array([[2.]], dtype=float32),), (array([[2.000004]], dtype=float32),))

# tf.test.create\_local\_cluster

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

Create and start local servers and return the associated Server objects.

### Aliases:

* tf.compat.v1.test.create\_local\_cluster
* tf.compat.v2.test.create\_local\_cluster
* tf.test.create\_local\_cluster

tf.test.create\_local\_cluster(  
    num\_workers,  
    num\_ps,  
    protocol='grpc',  
    worker\_config=None,  
    ps\_config=None  
)

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

"PS" stands for "parameter server": a task responsible for storing and updating the model's parameters. Other tasks send updates to these parameters as they work on optimizing the parameters. This particular division of labor between tasks is not required, but is common for distributed training.

Read more at https://www.tensorflow.org/guide/extend/architecture

Figure illustrates the interaction of these components. "/job:worker/task:0" and "/job:ps/task:0" are both tasks with worker services.

#### Example:

workers, \_ = tf.test.create\_local\_cluster(num\_workers=2, num\_ps=2)  
  
worker\_sessions = [tf.compat.v1.Session(w.target) for w in workers]  
  
with tf.device("/job:ps/task:0"):  
  ...  
with tf.device("/job:ps/task:1"):  
  ...  
with tf.device("/job:worker/task:0"):  
  ...  
with tf.device("/job:worker/task:1"):  
  ...  
  
worker\_sessions[0].run(...)

#### Args:

* **num\_workers**: Number of worker servers to start.
* **num\_ps**: Number of PS servers to start.
* **protocol**: Communication protocol. Allowed values are documented in the documentation of [tf.distribute.Server](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Server).
* **worker\_config**: (optional) tf.ConfigProto to initialize workers. Can be used to instantiate multiple devices etc.
* **ps\_config**: (optional) tf.ConfigProto to initialize PS servers.

#### Returns:

A tuple (worker\_servers, ps\_servers). worker\_servers is a list of num\_workers objects of type [tf.distribute.Server](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/distribute/Server) (all running locally); and ps\_servers is a list of num\_ps objects of similar type.

#### Raises:

* **ImportError**: if portpicker module was not found at load time

# tf.test.gpu\_device\_name

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

Returns the name of a GPU device if available or the empty string.

### Aliases:

* tf.compat.v1.test.gpu\_device\_name
* tf.compat.v2.test.gpu\_device\_name
* tf.test.gpu\_device\_name

tf.test.gpu\_device\_name()

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

# tf.test.is\_built\_with\_cuda

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

Returns whether TensorFlow was built with CUDA (GPU) support.

### Aliases:

* tf.compat.v1.test.is\_built\_with\_cuda
* tf.compat.v2.test.is\_built\_with\_cuda
* tf.test.is\_built\_with\_cuda

tf.test.is\_built\_with\_cuda()

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

# tf.test.is\_gpu\_available

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

Returns whether TensorFlow can access a GPU.

### Aliases:

* tf.compat.v1.test.is\_gpu\_available
* tf.compat.v2.test.is\_gpu\_available
* tf.test.is\_gpu\_available

tf.test.is\_gpu\_available(  
    cuda\_only=False,  
    min\_cuda\_compute\_capability=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)

### Used in the tutorials:

* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)
* [Text classification of movie reviews with Keras and TensorFlow Hub](https://www.tensorflow.org/beta/tutorials/keras/basic_text_classification_with_tfhub)

**Warning:** if a non-GPU version of the package is installed, the function would also return False. Use [**tf.test.is\_built\_with\_cuda**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/is_built_with_cuda) to validate if TensorFlow was build with CUDA support.

#### Args:

* **cuda\_only**: limit the search to CUDA GPUs.
* **min\_cuda\_compute\_capability**: a (major,minor) pair that indicates the minimum CUDA compute capability required, or None if no requirement.

#### Returns:

True if a GPU device of the requested kind is available.

# tf.test.main

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

Runs all unit tests.

### Aliases:

* tf.compat.v1.test.main
* tf.compat.v2.test.main
* tf.test.main

tf.test.main(argv=None)

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

# tf.test.TestCase

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase#top_of_page)
* [Class TestCase](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase#class_testcase)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase#__init__)
* [Child Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase#child_classes)

## Class TestCase

Base class for tests that need to test TensorFlow.

### Aliases:

* Class tf.compat.v1.test.TestCase
* Class tf.compat.v2.test.TestCase
* Class tf.test.TestCase

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

## \_\_init\_\_

\_\_init\_\_(methodName='runTest')

## Child Classes

[class failureException](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/test/TestCase/failureException)

## Methods

### \_\_call\_\_

\_\_call\_\_(  
    \*args,  
    \*\*kwds  
)

### \_\_eq\_\_

\_\_eq\_\_(other)

### addCleanup

addCleanup(  
    function,  
    \*args,  
    \*\*kwargs  
)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

### addTypeEqualityFunc

addTypeEqualityFunc(  
    typeobj,  
    function  
)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

#### Args:

* **typeobj**: The data type to call this function on when both values are of the same type in assertEqual().
* **function**: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

### assertAllClose

assertAllClose(  
    a,  
    b,  
    rtol=1e-06,  
    atol=1e-06,  
    msg=None  
)

Asserts that two structures of numpy arrays or Tensors, have near values.

a and b can be arbitrarily nested structures. A layer of a nested structure can be a dict, namedtuple, tuple or list.

#### Args:

* **a**: The expected numpy ndarray, or anything that can be converted into a numpy ndarray(including Tensor), or any arbitrarily nested of structure of these.
* **b**: The actual numpy ndarray, or anything that can be converted into a numpy ndarray(including Tensor), or any arbitrarily nested of structure of these.
* **rtol**: relative tolerance.
* **atol**: absolute tolerance.
* **msg**: Optional message to report on failure.

#### Raises:

* **ValueError**: if only one of a[p] and b[p] is a dict or a[p] and b[p] have different length, where [p] denotes a path to the nested structure, e.g. given a = [(1, 1), {'d': (6, 7)}]and [p] = [1]['d'], then a[p] = (6, 7).

### assertAllCloseAccordingToType

assertAllCloseAccordingToType(  
    a,  
    b,  
    rtol=1e-06,  
    atol=1e-06,  
    float\_rtol=1e-06,  
    float\_atol=1e-06,  
    half\_rtol=0.001,  
    half\_atol=0.001,  
    bfloat16\_rtol=0.01,  
    bfloat16\_atol=0.01,  
    msg=None  
)

Like assertAllClose, but also suitable for comparing fp16 arrays.

In particular, the tolerance is reduced to 1e-3 if at least one of the arguments is of type float16.

#### Args:

* **a**: the expected numpy ndarray or anything can be converted to one.
* **b**: the actual numpy ndarray or anything can be converted to one.
* **rtol**: relative tolerance.
* **atol**: absolute tolerance.
* **float\_rtol**: relative tolerance for float32.
* **float\_atol**: absolute tolerance for float32.
* **half\_rtol**: relative tolerance for float16.
* **half\_atol**: absolute tolerance for float16.
* **bfloat16\_rtol**: relative tolerance for bfloat16.
* **bfloat16\_atol**: absolute tolerance for bfloat16.
* **msg**: Optional message to report on failure.

### assertAllEqual

assertAllEqual(  
    a,  
    b,  
    msg=None  
)

Asserts that two numpy arrays or Tensors have the same values.

#### Args:

* **a**: the expected numpy ndarray or anything can be converted to one.
* **b**: the actual numpy ndarray or anything can be converted to one.
* **msg**: Optional message to report on failure.

### assertAllGreater

assertAllGreater(  
    a,  
    comparison\_target  
)

Assert element values are all greater than a target value.

#### Args:

* **a**: The numpy ndarray, or anything that can be converted into a numpy ndarray (including Tensor).
* **comparison\_target**: The target value of comparison.

### assertAllGreaterEqual

assertAllGreaterEqual(  
    a,  
    comparison\_target  
)

Assert element values are all greater than or equal to a target value.

#### Args:

* **a**: The numpy ndarray, or anything that can be converted into a numpy ndarray (including Tensor).
* **comparison\_target**: The target value of comparison.

### assertAllInRange

assertAllInRange(  
    target,  
    lower\_bound,  
    upper\_bound,  
    open\_lower\_bound=False,  
    open\_upper\_bound=False  
)

Assert that elements in a Tensor are all in a given range.

#### Args:

* **target**: The numpy ndarray, or anything that can be converted into a numpy ndarray(including Tensor).
* **lower\_bound**: lower bound of the range
* **upper\_bound**: upper bound of the range
* **open\_lower\_bound**: (bool) whether the lower bound is open (i.e., > rather than the default >=)
* **open\_upper\_bound**: (bool) whether the upper bound is open (i.e., < rather than the default <=)

#### Raises:

* **AssertionError**: if the value tensor does not have an ordered numeric type (float\* or int\*), or if there are nan values, or if any of the elements do not fall in the specified range.

### assertAllInSet

assertAllInSet(  
    target,  
    expected\_set  
)

Assert that elements of a Tensor are all in a given closed set.

#### Args:

* **target**: The numpy ndarray, or anything that can be converted into a numpy ndarray(including Tensor).
* **expected\_set**: (list, tuple or set) The closed set that the elements of the value of target are expected to fall into.

#### Raises:

* **AssertionError**: if any of the elements do not fall into expected\_set.

### assertAllLess

assertAllLess(  
    a,  
    comparison\_target  
)

Assert element values are all less than a target value.

#### Args:

* **a**: The numpy ndarray, or anything that can be converted into a numpy ndarray (including Tensor).
* **comparison\_target**: The target value of comparison.

### assertAllLessEqual

assertAllLessEqual(  
    a,  
    comparison\_target  
)

Assert element values are all less than or equal to a target value.

#### Args:

* **a**: The numpy ndarray, or anything that can be converted into a numpy ndarray (including Tensor).
* **comparison\_target**: The target value of comparison.

### assertAlmostEqual

assertAlmostEqual(  
    first,  
    second,  
    places=None,  
    msg=None,  
    delta=None  
)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most signficant digit).

If the two objects compare equal then they will automatically compare almost equal.

### assertAlmostEquals

assertAlmostEquals(  
    \*args,  
    \*\*kwargs  
)

### assertArrayNear

assertArrayNear(  
    farray1,  
    farray2,  
    err,  
    msg=None  
)

Asserts that two float arrays are near each other.

Checks that for all elements of farray1 and farray2 |f1 - f2| < err. Asserts a test failure if not.

#### Args:

* **farray1**: a list of float values.
* **farray2**: a list of float values.
* **err**: a float value.
* **msg**: Optional message to report on failure.

### assertBetween

assertBetween(  
    value,  
    minv,  
    maxv,  
    msg=None  
)

Asserts that value is between minv and maxv (inclusive).

### assertCommandFails

assertCommandFails(  
    command,  
    regexes,  
    env=None,  
    close\_fds=True,  
    msg=None  
)

Asserts a shell command fails and the error matches a regex in a list.

#### Args:

* **command**: List or string representing the command to run.
* **regexes**: the list of regular expression strings.
* **env**: Dictionary of environment variable settings. If None, no environment variables will be set for the child process. This is to make tests more hermetic. NOTE: this behavior is different than the standard subprocess module.
* **close\_fds**: Whether or not to close all open fd's in the child after forking.
* **msg**: Optional message to report on failure.

### assertCommandSucceeds

assertCommandSucceeds(  
    command,  
    regexes=(b'',),  
    env=None,  
    close\_fds=True,  
    msg=None  
)

Asserts that a shell command succeeds (i.e. exits with code 0).

#### Args:

* **command**: List or string representing the command to run.
* **regexes**: List of regular expression byte strings that match success.
* **env**: Dictionary of environment variable settings. If None, no environment variables will be set for the child process. This is to make tests more hermetic. NOTE: this behavior is different than the standard subprocess module.
* **close\_fds**: Whether or not to close all open fd's in the child after forking.
* **msg**: Optional message to report on failure.

### assertContainsExactSubsequence

assertContainsExactSubsequence(  
    container,  
    subsequence,  
    msg=None  
)

Asserts that "container" contains "subsequence" as an exact subsequence.

Asserts that "container" contains all the elements of "subsequence", in order, and without other elements interspersed. For example, [1, 2, 3] is an exact subsequence of [0, 0, 1, 2, 3, 0] but not of [0, 0, 1, 2, 0, 3, 0].

#### Args:

* **container**: the list we're testing for subsequence inclusion.
* **subsequence**: the list we hope will be an exact subsequence of container.
* **msg**: Optional message to report on failure.

### assertContainsInOrder

assertContainsInOrder(  
    strings,  
    target,  
    msg=None  
)

Asserts that the strings provided are found in the target in order.

This may be useful for checking HTML output.

#### Args:

* **strings**: A list of strings, such as [ 'fox', 'dog' ]
* **target**: A target string in which to look for the strings, such as 'The quick brown fox jumped over the lazy dog'.
* **msg**: Optional message to report on failure.

### assertContainsSubsequence

assertContainsSubsequence(  
    container,  
    subsequence,  
    msg=None  
)

Asserts that "container" contains "subsequence" as a subsequence.

Asserts that "container" contains all the elements of "subsequence", in order, but possibly with other elements interspersed. For example, [1, 2, 3] is a subsequence of [0, 0, 1, 2, 0, 3, 0] but not of [0, 0, 1, 3, 0, 2, 0].

#### Args:

* **container**: the list we're testing for subsequence inclusion.
* **subsequence**: the list we hope will be a subsequence of container.
* **msg**: Optional message to report on failure.

### assertContainsSubset

assertContainsSubset(  
    expected\_subset,  
    actual\_set,  
    msg=None  
)

Checks whether actual iterable is a superset of expected iterable.

### assertCountEqual

assertCountEqual(  
    first,  
    second,  
    msg=None  
)

An unordered sequence comparison asserting that the same elements, regardless of order. If the same element occurs more than once, it verifies that the elements occur the same number of times.

self.assertEqual(Counter(list(first)),  
                 Counter(list(second)))

Example: - [0, 1, 1] and [1, 0, 1] compare equal. - [0, 0, 1] and [0, 1] compare unequal.

### assertDTypeEqual

assertDTypeEqual(  
    target,  
    expected\_dtype  
)

Assert ndarray data type is equal to expected.

#### Args:

* **target**: The numpy ndarray, or anything that can be converted into a numpy ndarray(including Tensor).
* **expected\_dtype**: Expected data type.

### assertDeviceEqual

assertDeviceEqual(  
    device1,  
    device2,  
    msg=None  
)

Asserts that the two given devices are the same.

#### Args:

* **device1**: A string device name or TensorFlow DeviceSpec object.
* **device2**: A string device name or TensorFlow DeviceSpec object.
* **msg**: Optional message to report on failure.

### assertDictContainsSubset

assertDictContainsSubset(  
    subset,  
    dictionary,  
    msg=None  
)

Checks whether dictionary is a superset of subset.

### assertDictEqual

assertDictEqual(  
    a,  
    b,  
    msg=None  
)

Raises AssertionError if a and b are not equal dictionaries.

#### Args:

* **a**: A dict, the expected value.
* **b**: A dict, the actual value.
* **msg**: An optional str, the associated message.

#### Raises:

* **AssertionError**: if the dictionaries are not equal.

### assertEmpty

assertEmpty(  
    container,  
    msg=None  
)

Asserts that an object has zero length.

#### Args:

* **container**: Anything that implements the collections.Sized interface.
* **msg**: Optional message to report on failure.

### assertEndsWith

assertEndsWith(  
    actual,  
    expected\_end,  
    msg=None  
)

Asserts that actual.endswith(expected\_end) is True.

#### Args:

* **actual**: str
* **expected\_end**: str
* **msg**: Optional message to report on failure.

### assertEqual

assertEqual(  
    first,  
    second,  
    msg=None  
)

Fail if the two objects are unequal as determined by the '==' operator.

### assertEquals

assertEquals(  
    \*args,  
    \*\*kwargs  
)

### assertFalse

assertFalse(  
    expr,  
    msg=None  
)

Check that the expression is false.

### assertGreater

assertGreater(  
    a,  
    b,  
    msg=None  
)

Just like self.assertTrue(a > b), but with a nicer default message.

### assertGreaterEqual

assertGreaterEqual(  
    a,  
    b,  
    msg=None  
)

Just like self.assertTrue(a >= b), but with a nicer default message.

### assertIn

assertIn(  
    member,  
    container,  
    msg=None  
)

Just like self.assertTrue(a in b), but with a nicer default message.

### assertIs

assertIs(  
    expr1,  
    expr2,  
    msg=None  
)

Just like self.assertTrue(a is b), but with a nicer default message.

### assertIsInstance

assertIsInstance(  
    obj,  
    cls,  
    msg=None  
)

Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

### assertIsNone

assertIsNone(  
    obj,  
    msg=None  
)

Same as self.assertTrue(obj is None), with a nicer default message.

### assertIsNot

assertIsNot(  
    expr1,  
    expr2,  
    msg=None  
)

Just like self.assertTrue(a is not b), but with a nicer default message.

### assertIsNotNone

assertIsNotNone(  
    obj,  
    msg=None  
)

Included for symmetry with assertIsNone.

### assertItemsEqual

assertItemsEqual(  
    first,  
    second,  
    msg=None  
)

An unordered sequence comparison asserting that the same elements, regardless of order. If the same element occurs more than once, it verifies that the elements occur the same number of times.

self.assertEqual(Counter(list(first)),  
                 Counter(list(second)))

Example: - [0, 1, 1] and [1, 0, 1] compare equal. - [0, 0, 1] and [0, 1] compare unequal.

### assertJsonEqual

assertJsonEqual(  
    first,  
    second,  
    msg=None  
)

Asserts that the JSON objects defined in two strings are equal.

A summary of the differences will be included in the failure message using assertSameStructure.

#### Args:

* **first**: A string contining JSON to decode and compare to second.
* **second**: A string contining JSON to decode and compare to first.
* **msg**: Additional text to include in the failure message.

### assertLen

assertLen(  
    container,  
    expected\_len,  
    msg=None  
)

Asserts that an object has the expected length.

#### Args:

* **container**: Anything that implements the collections.Sized interface.
* **expected\_len**: The expected length of the container.
* **msg**: Optional message to report on failure.

### assertLess

assertLess(  
    a,  
    b,  
    msg=None  
)

Just like self.assertTrue(a < b), but with a nicer default message.

### assertLessEqual

assertLessEqual(  
    a,  
    b,  
    msg=None  
)

Just like self.assertTrue(a <= b), but with a nicer default message.

### assertListEqual

assertListEqual(  
    list1,  
    list2,  
    msg=None  
)

A list-specific equality assertion.

#### Args:

* **list1**: The first list to compare.
* **list2**: The second list to compare.
* **msg**: Optional message to use on failure instead of a list of differences.

### assertLogs

assertLogs(  
    logger=None,  
    level=None  
)

Fail unless a log message of level level or higher is emitted on logger\_name or its children. If omitted, level defaults to INFO and logger defaults to the root logger.

This method must be used as a context manager, and will yield a recording object with two attributes: output and records. At the end of the context manager, the output attribute will be a list of the matching formatted log messages and the records attribute will be a list of the corresponding LogRecord objects.

Example::

with self.assertLogs('foo', level='INFO') as cm:  
    logging.getLogger('foo').info('first message')  
    logging.getLogger('foo.bar').error('second message')  
self.assertEqual(cm.output, ['INFO:foo:first message',  
                             'ERROR:foo.bar:second message'])

### assertMultiLineEqual

assertMultiLineEqual(  
    first,  
    second,  
    msg=None,  
    \*\*kwargs  
)

Asserts that two multi-line strings are equal.

### assertNDArrayNear

assertNDArrayNear(  
    ndarray1,  
    ndarray2,  
    err,  
    msg=None  
)

Asserts that two numpy arrays have near values.

#### Args:

* **ndarray1**: a numpy ndarray.
* **ndarray2**: a numpy ndarray.
* **err**: a float. The maximum absolute difference allowed.
* **msg**: Optional message to report on failure.

### assertNear

assertNear(  
    f1,  
    f2,  
    err,  
    msg=None  
)

Asserts that two floats are near each other.

Checks that |f1 - f2| < err and asserts a test failure if not.

#### Args:

* **f1**: A float value.
* **f2**: A float value.
* **err**: A float value.
* **msg**: An optional string message to append to the failure message.

### assertNoCommonElements

assertNoCommonElements(  
    expected\_seq,  
    actual\_seq,  
    msg=None  
)

Checks whether actual iterable and expected iterable are disjoint.

### assertNotAllClose

assertNotAllClose(  
    a,  
    b,  
    \*\*kwargs  
)

Assert that two numpy arrays, or Tensors, do not have near values.

#### Args:

* **a**: the first value to compare.
* **b**: the second value to compare.
* **\*\*kwargs**: additional keyword arguments to be passed to the underlying assertAllClose call.

#### Raises:

* **AssertionError**: If a and b are unexpectedly close at all elements.

### assertNotAlmostEqual

assertNotAlmostEqual(  
    first,  
    second,  
    places=None,  
    msg=None,  
    delta=None  
)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most signficant digit).

Objects that are equal automatically fail.

### assertNotAlmostEquals

assertNotAlmostEquals(  
    \*args,  
    \*\*kwargs  
)

### assertNotEmpty

assertNotEmpty(  
    container,  
    msg=None  
)

Asserts that an object has non-zero length.

#### Args:

* **container**: Anything that implements the collections.Sized interface.
* **msg**: Optional message to report on failure.

### assertNotEndsWith

assertNotEndsWith(  
    actual,  
    unexpected\_end,  
    msg=None  
)

Asserts that actual.endswith(unexpected\_end) is False.

#### Args:

* **actual**: str
* **unexpected\_end**: str
* **msg**: Optional message to report on failure.

### assertNotEqual

assertNotEqual(  
    first,  
    second,  
    msg=None  
)

Fail if the two objects are equal as determined by the '!=' operator.

### assertNotEquals

assertNotEquals(  
    \*args,  
    \*\*kwargs  
)

### assertNotIn

assertNotIn(  
    member,  
    container,  
    msg=None  
)

Just like self.assertTrue(a not in b), but with a nicer default message.

### assertNotIsInstance

assertNotIsInstance(  
    obj,  
    cls,  
    msg=None  
)

Included for symmetry with assertIsInstance.

### assertNotRegex

assertNotRegex(  
    text,  
    unexpected\_regex,  
    msg=None  
)

Fail the test if the text matches the regular expression.

### assertNotStartsWith

assertNotStartsWith(  
    actual,  
    unexpected\_start,  
    msg=None  
)

Asserts that actual.startswith(unexpected\_start) is False.

#### Args:

* **actual**: str
* **unexpected\_start**: str
* **msg**: Optional message to report on failure.

### assertProtoEquals

assertProtoEquals(  
    expected\_message\_maybe\_ascii,  
    message,  
    msg=None  
)

Asserts that message is same as parsed expected\_message\_ascii.

Creates another prototype of message, reads the ascii message into it and then compares them using self.\_AssertProtoEqual().

#### Args:

* **expected\_message\_maybe\_ascii**: proto message in original or ascii form.
* **message**: the message to validate.
* **msg**: Optional message to report on failure.

### assertProtoEqualsVersion

assertProtoEqualsVersion(  
    expected,  
    actual,  
    producer=versions.GRAPH\_DEF\_VERSION,  
    min\_consumer=versions.GRAPH\_DEF\_VERSION\_MIN\_CONSUMER,  
    msg=None  
)

### assertRaises

assertRaises(  
    excClass,  
    callableObj=None,  
    \*args,  
    \*\*kwargs  
)

Fail unless an exception of class excClass is raised by callableObj when invoked with arguments args and keyword arguments kwargs. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with callableObj omitted or None, will return a context object used like this::

 with self.assertRaises(SomeException):  
     do\_something()

An optional keyword argument 'msg' can be provided when assertRaises is used as a context object.

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion::

with self.assertRaises(SomeException) as cm:  
    do\_something()  
the\_exception = cm.exception  
self.assertEqual(the\_exception.error\_code, 3)

### assertRaisesOpError

assertRaisesOpError(expected\_err\_re\_or\_predicate)

### assertRaisesRegex

assertRaisesRegex(  
    expected\_exception,  
    expected\_regex,  
    callable\_obj=None,  
    \*args,  
    \*\*kwargs  
)

Asserts that the message in a raised exception matches a regex.

#### Args:

* **expected\_exception**: Exception class expected to be raised.
* **expected\_regex**: Regex (re pattern object or string) expected to be found in error message.
* **callable\_obj**: Function to be called.
* **msg**: Optional message used in case of failure. Can only be used when assertRaisesRegex is used as a context manager.
* **args**: Extra args.
* **kwargs**: Extra kwargs.

### assertRaisesRegexp

assertRaisesRegexp(  
    expected\_exception,  
    expected\_regex,  
    callable\_obj=None,  
    \*args,  
    \*\*kwargs  
)

Asserts that the message in a raised exception matches a regex.

#### Args:

* **expected\_exception**: Exception class expected to be raised.
* **expected\_regex**: Regex (re pattern object or string) expected to be found in error message.
* **callable\_obj**: Function to be called.
* **msg**: Optional message used in case of failure. Can only be used when assertRaisesRegex is used as a context manager.
* **args**: Extra args.
* **kwargs**: Extra kwargs.

### assertRaisesWithLiteralMatch

assertRaisesWithLiteralMatch(  
    expected\_exception,  
    expected\_exception\_message,  
    callable\_obj=None,  
    \*args,  
    \*\*kwargs  
)

Asserts that the message in a raised exception equals the given string.

Unlike assertRaisesRegex, this method takes a literal string, not a regular expression.

with self.assertRaisesWithLiteralMatch(ExType, 'message'): DoSomething()

#### Args:

* **expected\_exception**: Exception class expected to be raised.
* **expected\_exception\_message**: String message expected in the raised exception. For a raise exception e, expected\_exception\_message must equal str(e).
* **callable\_obj**: Function to be called, or None to return a context.
* **\*args**: Extra args.
* **\*\*kwargs**: Extra kwargs.

#### Returns:

A context manager if callable\_obj is None. Otherwise, None.

#### Raises:

self.failureException if callable\_obj does not raise a matching exception.

### assertRaisesWithPredicateMatch

assertRaisesWithPredicateMatch(  
    \*args,  
    \*\*kwds  
)

Returns a context manager to enclose code expected to raise an exception.

If the exception is an OpError, the op stack is also included in the message predicate search.

#### Args:

* **exception\_type**: The expected type of exception that should be raised.
* **expected\_err\_re\_or\_predicate**: If this is callable, it should be a function of one argument that inspects the passed-in exception and returns True (success) or False (please fail the test). Otherwise, the error message is expected to match this regular expression partially.

#### Returns:

A context manager to surround code that is expected to raise an exception.

### assertRegex

assertRegex(  
    text,  
    expected\_regex,  
    msg=None  
)

Fail the test unless the text matches the regular expression.

### assertRegexMatch

assertRegexMatch(  
    actual\_str,  
    regexes,  
    message=None  
)

Asserts that at least one regex in regexes matches str.

If possible you should use assertRegex, which is a simpler version of this method. assertRegextakes a single regular expression (a string or re compiled object) instead of a list.

#### Notes:

1. This function uses substring matching, i.e. the matching succeeds if any substring of the error message matches any regex in the list. This is more convenient for the user than full-string matching.
2. If regexes is the empty list, the matching will always fail.
3. Use regexes=[''] for a regex that will always pass.
4. '.' matches any single character except the newline. To match any character, use '(.|\n)'.
5. '^' matches the beginning of each line, not just the beginning of the string. Similarly, '$' matches the end of each line.
6. An exception will be thrown if regexes contains an invalid regex.

#### Args:

* **actual\_str**: The string we try to match with the items in regexes.
* **regexes**: The regular expressions we want to match against str. See "Notes" above for detailed notes on how this is interpreted.
* **message**: The message to be printed if the test fails.

### assertRegexpMatches

assertRegexpMatches(  
    \*args,  
    \*\*kwargs  
)

### assertSameElements

assertSameElements(  
    expected\_seq,  
    actual\_seq,  
    msg=None  
)

Asserts that two sequences have the same elements (in any order).

This method, unlike assertCountEqual, doesn't care about any duplicates in the expected and actual sequences.

assertSameElements([1, 1, 1, 0, 0, 0], [0, 1]) # Doesn't raise an AssertionError

If possible, you should use assertCountEqual instead of assertSameElements.

#### Args:

* **expected\_seq**: A sequence containing elements we are expecting.
* **actual\_seq**: The sequence that we are testing.
* **msg**: The message to be printed if the test fails.

### assertSameStructure

assertSameStructure(  
    a,  
    b,  
    aname='a',  
    bname='b',  
    msg=None  
)

Asserts that two values contain the same structural content.

The two arguments should be data trees consisting of trees of dicts and lists. They will be deeply compared by walking into the contents of dicts and lists; other items will be compared using the == operator. If the two structures differ in content, the failure message will indicate the location within the structures where the first difference is found. This may be helpful when comparing large structures.

Mixed Sequence and Set types are supported. Mixed Mapping types are supported, but the order of the keys will not be considered in the comparison.

#### Args:

* **a**: The first structure to compare.
* **b**: The second structure to compare.
* **aname**: Variable name to use for the first structure in assertion messages.
* **bname**: Variable name to use for the second structure.
* **msg**: Additional text to include in the failure message.

### assertSequenceAlmostEqual

assertSequenceAlmostEqual(  
    expected\_seq,  
    actual\_seq,  
    places=None,  
    msg=None,  
    delta=None  
)

An approximate equality assertion for ordered sequences.

Fail if the two sequences are unequal as determined by their value differences rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the difference between each value in the two sequences is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most signficant digit).

If the two sequences compare equal then they will automatically compare almost equal.

#### Args:

* **expected\_seq**: A sequence containing elements we are expecting.
* **actual\_seq**: The sequence that we are testing.
* **places**: The number of decimal places to compare.
* **msg**: The message to be printed if the test fails.
* **delta**: The OK difference between compared values.

### assertSequenceEqual

assertSequenceEqual(  
    seq1,  
    seq2,  
    msg=None,  
    seq\_type=None  
)

An equality assertion for ordered sequences (like lists and tuples).

For the purposes of this function, a valid ordered sequence type is one which can be indexed, has a length, and has an equality operator.

#### Args:

* **seq1**: The first sequence to compare.
* **seq2**: The second sequence to compare.
* **seq\_type**: The expected datatype of the sequences, or None if no datatype should be enforced.
* **msg**: Optional message to use on failure instead of a list of differences.

### assertSequenceStartsWith

assertSequenceStartsWith(  
    prefix,  
    whole,  
    msg=None  
)

An equality assertion for the beginning of ordered sequences.

If prefix is an empty sequence, it will raise an error unless whole is also an empty sequence.

If prefix is not a sequence, it will raise an error if the first element of whole does not match.

#### Args:

* **prefix**: A sequence expected at the beginning of the whole parameter.
* **whole**: The sequence in which to look for prefix.
* **msg**: Optional message to report on failure.

### assertSetEqual

assertSetEqual(  
    set1,  
    set2,  
    msg=None  
)

A set-specific equality assertion.

#### Args:

* **set1**: The first set to compare.
* **set2**: The second set to compare.
* **msg**: Optional message to use on failure instead of a list of differences.

assertSetEqual uses ducktyping to support different types of sets, and is optimized for sets specifically (parameters must support a difference method).

### assertShapeEqual

assertShapeEqual(  
    np\_array,  
    tf\_tensor,  
    msg=None  
)

Asserts that a Numpy ndarray and a TensorFlow tensor have the same shape.

#### Args:

* **np\_array**: A Numpy ndarray or Numpy scalar.
* **tf\_tensor**: A Tensor.
* **msg**: Optional message to report on failure.

#### Raises:

* **TypeError**: If the arguments have the wrong type.

### assertStartsWith

assertStartsWith(  
    actual,  
    expected\_start,  
    msg=None  
)

Assert that actual.startswith(expected\_start) is True.

#### Args:

* **actual**: str
* **expected\_start**: str
* **msg**: Optional message to report on failure.

### assertTotallyOrdered

assertTotallyOrdered(  
    \*groups,  
    \*\*kwargs  
)

Asserts that total ordering has been implemented correctly.

For example, say you have a class A that compares only on its attribute x. Comparators other than **lt**are omitted for brevity.

class A(object): def **init**(self, x, y): self.x = x self.y = y

def **hash**(self): return hash(self.x)

def **lt**(self, other): try: return self.x < other.x except AttributeError: return NotImplemented

assertTotallyOrdered will check that instances can be ordered correctly. For example,

self.assertTotallyOrdered( [None], # None should come before everything else. [1], # Integers sort earlier. [A(1, 'a')], [A(2, 'b')], # 2 is after 1. [A(3, 'c'), A(3, 'd')], # The second argument is irrelevant. [A(4, 'z')], ['foo']) # Strings sort last.

#### Args:

* **\*groups**: A list of groups of elements. Each group of elements is a list of objects that are equal. The elements in each group must be less than the elements in the group after it. For example, these groups are totally ordered: [None], [1], [2, 2], [3]. \*\*kwargs: optional msg keyword argument can be passed.

### assertTrue

assertTrue(  
    expr,  
    msg=None  
)

Check that the expression is true.

### assertTupleEqual

assertTupleEqual(  
    tuple1,  
    tuple2,  
    msg=None  
)

A tuple-specific equality assertion.

#### Args:

* **tuple1**: The first tuple to compare.
* **tuple2**: The second tuple to compare.
* **msg**: Optional message to use on failure instead of a list of differences.

### assertUrlEqual

assertUrlEqual(  
    a,  
    b,  
    msg=None  
)

Asserts that urls are equal, ignoring ordering of query params.

### assertWarns

assertWarns(  
    expected\_warning,  
    callable\_obj=None,  
    \*args,  
    \*\*kwargs  
)

Fail unless a warning of class warnClass is triggered by callable\_obj when invoked with arguments args and keyword arguments kwargs. If a different type of warning is triggered, it will not be handled: depending on the other warning filtering rules in effect, it might be silenced, printed out, or raised as an exception.

If called with callable\_obj omitted or None, will return a context object used like this::

 with self.assertWarns(SomeWarning):  
     do\_something()

An optional keyword argument 'msg' can be provided when assertWarns is used as a context object.

The context manager keeps a reference to the first matching warning as the 'warning' attribute; similarly, the 'filename' and 'lineno' attributes give you information about the line of Python code from which the warning was triggered. This allows you to inspect the warning after the assertion::

with self.assertWarns(SomeWarning) as cm:  
    do\_something()  
the\_warning = cm.warning  
self.assertEqual(the\_warning.some\_attribute, 147)

### assertWarnsRegex

assertWarnsRegex(  
    expected\_warning,  
    expected\_regex,  
    callable\_obj=None,  
    \*args,  
    \*\*kwargs  
)

Asserts that the message in a triggered warning matches a regexp. Basic functioning is similar to assertWarns() with the addition that only warnings whose messages also match the regular expression are considered successful matches.

#### Args:

* **expected\_warning**: Warning class expected to be triggered.
* **expected\_regex**: Regex (re pattern object or string) expected to be found in error message.
* **callable\_obj**: Function to be called.
* **msg**: Optional message used in case of failure. Can only be used when assertWarnsRegex is used as a context manager.
* **args**: Extra args.
* **kwargs**: Extra kwargs.

### assert\_

assert\_(  
    \*args,  
    \*\*kwargs  
)

### cached\_session

cached\_session(  
    \*args,  
    \*\*kwds  
)

Returns a TensorFlow Session for use in executing tests.

This method behaves differently than self.session(): for performance reasons cached\_session will by default reuse the same session within the same test. The session returned by this function will only be closed at the end of the test (in the TearDown function).

Use the use\_gpu and force\_gpu options to control where ops are run. If force\_gpu is True, all ops are pinned to /device:GPU:0. Otherwise, if use\_gpu is True, TensorFlow tries to run as many ops on the GPU as possible. If both force\_gpu anduse\_gpu` are False, all ops are pinned to the CPU.

#### Example:

class MyOperatorTest(test\_util.TensorFlowTestCase):  
  def testMyOperator(self):  
    with self.cached\_session(use\_gpu=True) as sess:  
      valid\_input = [1.0, 2.0, 3.0, 4.0, 5.0]  
      result = MyOperator(valid\_input).eval()  
      self.assertEqual(result, [1.0, 2.0, 3.0, 5.0, 8.0]  
      invalid\_input = [-1.0, 2.0, 7.0]  
      with self.assertRaisesOpError("negative input not supported"):  
        MyOperator(invalid\_input).eval()

#### Args:

* **graph**: Optional graph to use during the returned session.
* **config**: An optional config\_pb2.ConfigProto to use to configure the session.
* **use\_gpu**: If True, attempt to run as many ops as possible on GPU.
* **force\_gpu**: If True, pin all ops to /device:GPU:0.

#### Yields:

A Session object that should be used as a context manager to surround the graph building and execution code in a test case.

### captureWritesToStream

captureWritesToStream(  
    \*args,  
    \*\*kwds  
)

A context manager that captures the writes to a given stream.

This context manager captures all writes to a given stream inside of a CapturedWrites object. When this context manager is created, it yields the CapturedWrites object. The captured contents can be accessed by calling .contents() on the CapturedWrites.

For this function to work, the stream must have a file descriptor that can be modified using os.dupand os.dup2, and the stream must support a .flush() method. The default python sys.stdout and sys.stderr are examples of this. Note that this does not work in Colab or Jupyter notebooks, because those use alternate stdout streams.

#### Example:

class MyOperatorTest(test\_util.TensorFlowTestCase):  
  def testMyOperator(self):  
    input = [1.0, 2.0, 3.0, 4.0, 5.0]  
    with self.captureWritesToStream(sys.stdout) as captured:  
      result = MyOperator(input).eval()  
    self.assertStartsWith(captured.contents(), "This was printed.")

#### Args:

* **stream**: The stream whose writes should be captured. This stream must have a file descriptor, support writing via using that file descriptor, and must have a .flush() method.

#### Yields:

A CapturedWrites object that contains all writes to the specified stream made during this context.

### checkedThread

checkedThread(  
    target,  
    args=None,  
    kwargs=None  
)

Returns a Thread wrapper that asserts 'target' completes successfully.

This method should be used to create all threads in test cases, as otherwise there is a risk that a thread will silently fail, and/or assertions made in the thread will not be respected.

#### Args:

* **target**: A callable object to be executed in the thread.
* **args**: The argument tuple for the target invocation. Defaults to ().
* **kwargs**: A dictionary of keyword arguments for the target invocation. Defaults to {}.

#### Returns:

A wrapper for threading.Thread that supports start() and join() methods.

### countTestCases

countTestCases()

### create\_tempdir

create\_tempdir(  
    name=None,  
    cleanup=None  
)

Create a temporary directory specific to the test.

NOTE: The directory and its contents will be recursively cleared before creation. This ensures that there is no pre-existing state.

This creates a named directory on disk that is isolated to this test, and will be properly cleaned up by the test. This avoids several pitfalls of creating temporary directories for test purposes, as well as makes it easier to setup directories and verify their contents.

See also: create\_tempfile() for creating temporary files.

#### Args:

* **name**: Optional name of the directory. If not given, a unique name will be generated and used.
* **cleanup**: Optional cleanup policy on when/if to remove the directory (and all its contents) at the end of the test. If None, then uses self.tempfile\_cleanup.

#### Returns:

A \_TempDir representing the created directory.

### create\_tempfile

create\_tempfile(  
    file\_path=None,  
    content=None,  
    mode='w',  
    encoding='utf8',  
    errors='strict',  
    cleanup=None  
)

Create a temporary file specific to the test.

This creates a named file on disk that is isolated to this test, and will be properly cleaned up by the test. This avoids several pitfalls of creating temporary files for test purposes, as well as makes it easier to setup files, their data, read them back, and inspect them when a test fails.

NOTE: This will zero-out the file. This ensures there is no pre-existing state.

See also: create\_tempdir() for creating temporary directories.

#### Args:

* **file\_path**: Optional file path for the temp file. If not given, a unique file name will be generated and used. Slashes are allowed in the name; any missing intermediate directories will be created. NOTE: This path is the path that will be cleaned up, including any directories in the path, e.g., 'foo/bar/baz.txt' will rm -r foo.
* **content**: Optional string or bytes to initially write to the file. If not specified, then an empty file is created.
* **mode**: Mode string to use when writing content. Only used if content is non-empty.
* **encoding**: Encoding to use when writing string content. Only used if content is text.
* **errors**: How to handle text to bytes encoding errors. Only used if content is text.
* **cleanup**: Optional cleanup policy on when/if to remove the directory (and all its contents) at the end of the test. If None, then uses self.tempfile\_cleanup.

#### Returns:

A \_TempFile representing the created file.

### debug

debug()

Run the test without collecting errors in a TestResult

### defaultTestResult

defaultTestResult()

### doCleanups

doCleanups()

Execute all cleanup functions. Normally called for you after tearDown.

### evaluate

evaluate(tensors)

Evaluates tensors and returns numpy values.

#### Args:

* **tensors**: A Tensor or a nested list/tuple of Tensors.

#### Returns:

tensors numpy values.

### fail

fail(  
    msg=None,  
    prefix=None  
)

Fail immediately with the given message, optionally prefixed.

### failIf

failIf(  
    \*args,  
    \*\*kwargs  
)

### failIfAlmostEqual

failIfAlmostEqual(  
    \*args,  
    \*\*kwargs  
)

### failIfEqual

failIfEqual(  
    \*args,  
    \*\*kwargs  
)

### failUnless

failUnless(  
    \*args,  
    \*\*kwargs  
)

### failUnlessAlmostEqual

failUnlessAlmostEqual(  
    \*args,  
    \*\*kwargs  
)

### failUnlessEqual

failUnlessEqual(  
    \*args,  
    \*\*kwargs  
)

### failUnlessRaises

failUnlessRaises(  
    \*args,  
    \*\*kwargs  
)

### get\_temp\_dir

get\_temp\_dir()

Returns a unique temporary directory for the test to use.

If you call this method multiple times during in a test, it will return the same folder. However, across different runs the directories will be different. This will ensure that across different runs tests will not be able to pollute each others environment. If you need multiple unique directories within a single test, you should use tempfile.mkdtemp as follows: tempfile.mkdtemp(dir=self.get\_temp\_dir()):

#### Returns:

string, the path to the unique temporary directory created for this test.

### id

id()

### run

run(result=None)

### session

session(  
    \*args,  
    \*\*kwds  
)

Returns a TensorFlow Session for use in executing tests.

Note that this will set this session and the graph as global defaults.

Use the use\_gpu and force\_gpu options to control where ops are run. If force\_gpu is True, all ops are pinned to /device:GPU:0. Otherwise, if use\_gpu is True, TensorFlow tries to run as many ops on the GPU as possible. If both force\_gpu anduse\_gpu` are False, all ops are pinned to the CPU.

#### Example:

class MyOperatorTest(test\_util.TensorFlowTestCase):  
  def testMyOperator(self):  
    with self.session(use\_gpu=True):  
      valid\_input = [1.0, 2.0, 3.0, 4.0, 5.0]  
      result = MyOperator(valid\_input).eval()  
      self.assertEqual(result, [1.0, 2.0, 3.0, 5.0, 8.0]  
      invalid\_input = [-1.0, 2.0, 7.0]  
      with self.assertRaisesOpError("negative input not supported"):  
        MyOperator(invalid\_input).eval()

#### Args:

* **graph**: Optional graph to use during the returned session.
* **config**: An optional config\_pb2.ConfigProto to use to configure the session.
* **use\_gpu**: If True, attempt to run as many ops as possible on GPU.
* **force\_gpu**: If True, pin all ops to /device:GPU:0.

#### Yields:

A Session object that should be used as a context manager to surround the graph building and execution code in a test case.

### setUp

setUp()

### setUpClass

setUpClass(cls)

Hook method for setting up class fixture before running tests in the class.

### shortDescription

shortDescription()

Formats both the test method name and the first line of its docstring.

If no docstring is given, only returns the method name.

This method overrides unittest.TestCase.shortDescription(), which only returns the first line of the docstring, obscuring the name of the test upon failure.

#### Returns:

* **desc**: A short description of a test method.

### skipTest

skipTest(reason)

Skip this test.

### subTest

subTest(  
    \*args,  
    \*\*kwds  
)

Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

### tearDown

tearDown()

### tearDownClass

tearDownClass(cls)

Hook method for deconstructing the class fixture after running all tests in the class.

### test\_session

test\_session(  
    graph=None,  
    config=None,  
    use\_gpu=False,  
    force\_gpu=False  
)

Use cached\_session instead. (deprecated)

**Warning:** THIS FUNCTION IS DEPRECATED. It will be removed in a future version. Instructions for updating: Use **self.session()** or **self.cached\_session()** instead.

## Class Members

* longMessage = True
* maxDiff = 1600
* tempfile\_cleanup

# tf.test.TestCase.failureException

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

## Class failureException

Assertion failed.

### Aliases:

* Class tf.compat.v1.test.TestCase.failureException
* Class tf.compat.v2.test.TestCase.failureException
* Class tf.test.TestCase.failureException

## \_\_init\_\_

\_\_init\_\_(  
    \*args,  
    \*\*kwargs  
)

Module: tf.compat.v1.tpu / tf.tpu

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

Ops related to Tensor Processing Units.

Modules

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

Classes

[class CrossShardOptimizer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/CrossShardOptimizer): An optimizer that averages gradients across TPU shards.

Functions

[batch\_parallel(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/batch_parallel): Shards computation along the batch dimension for parallel execution.

[bfloat16\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/bfloat16_scope): Scope class for bfloat16 variables so that the model uses custom getter.

[core(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/core): Returns the device name for a core in a replicated TPU computation.

[cross\_replica\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/cross_replica_sum): Sum the input tensor across replicas according to group\_assignment.

[initialize\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/initialize_system): Initializes a distributed TPU system for use with TensorFlow.

[outside\_compilation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/outside_compilation): Builds part of a computation outside any current TPU replicate scope.

[replicate(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/replicate): Builds a graph operator that runs a replicated TPU computation.

[rewrite(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/rewrite): Rewrites computation for execution on a TPU system.

[shard(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/shard): Shards computation for parallel execution.

[shutdown\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/tpu/shutdown_system): Shuts down a running a distributed TPU system.

# tf.compat.v1.tpu.batch\_parallel

Shards computation along the batch dimension for parallel execution.

tf.compat.v1.tpu.batch\_parallel(  
    computation,  
    inputs=None,  
    num\_shards=1,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

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

Convenience wrapper around shard().

inputs must be a list of Tensors or None (equivalent to an empty list). Each input is split into num\_shards pieces along the 0-th dimension, and computation is applied to each shard in parallel.

Tensors are broadcast to all shards if they are lexically captured by computation. e.g.,

x = tf.constant(7) def computation(): return x + 3 ... = shard(computation, ...)

The outputs from all shards are concatenated back together along their 0-th dimension.

Inputs and outputs of the computation must be at least rank-1 Tensors.

#### Args:

* **computation**: A Python function that builds a computation to apply to each shard of the input.
* **inputs**: A list of input tensors or None (equivalent to an empty list). The 0-th dimension of each Tensor must have size divisible by num\_shards.
* **num\_shards**: The number of shards.
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each shard of the computation uses only one core, and there is either only one shard, or the number of shards is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.

#### Returns:

A list of output tensors.

#### Raises:

* **ValueError**: If num\_shards <= 0

# tf.compat.v1.tpu.bfloat16\_scope

Scope class for bfloat16 variables so that the model uses custom getter.

tf.compat.v1.tpu.bfloat16\_scope()

Defined in [python/tpu/bfloat16.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/bfloat16.py).

This enables variables to be read as bfloat16 type when using get\_variable.

# tf.compat.v1.tpu.core

Returns the device name for a core in a replicated TPU computation.

tf.compat.v1.tpu.core(num)

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

#### Args:

* **num**: the virtual core number within each replica to which operators should be assigned.

#### Returns:

A device name, suitable for passing to tf.device().

# tf.compat.v1.tpu.CrossShardOptimizer

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

## Class CrossShardOptimizer

An optimizer that averages gradients across TPU shards.

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

Defined in [python/tpu/tpu\_optimizer.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/tpu_optimizer.py).

## \_\_init\_\_

\_\_init\_\_(  
    opt,  
    reduction=losses.Reduction.MEAN,  
    name='CrossShardOptimizer',  
    group\_assignment=None  
)

Construct a new cross-shard optimizer.

#### Args:

* **opt**: An existing Optimizer to encapsulate.
* **reduction**: The reduction to apply to the shard losses.
* **name**: Optional name prefix for the operations created when applying gradients. Defaults to "CrossShardOptimizer".
* **group\_assignment**: Optional 2d int32 lists with shape [num\_groups, num\_replicas\_per\_group] which describles how to apply optimizer to subgroups.

#### Raises:

* **ValueError**: If reduction is not a valid cross-shard reduction.

## Methods

### apply\_gradients

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

Apply gradients to variables.

Calls tpu\_ops.cross\_replica\_sum() to sum gradient contributions across replicas, and then applies the real optimizer.

#### Args:

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

#### Returns:

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

#### Raises:

* **ValueError**: If the grads\_and\_vars is malformed.

### compute\_gradients

compute\_gradients(  
    loss,  
    var\_list=None,  
    \*\*kwargs  
)

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

This simply wraps the compute\_gradients() from the real optimizer. The gradients will be aggregated in the apply\_gradients() so that user can modify the gradients like clipping with per replica global norm if needed. The global norm with aggregated gradients can be bad as one replica's huge gradients can hurt the gradients from other replicas.

#### Args:

* **loss**: A Tensor containing the value to minimize.
* **var\_list**: Optional list or tuple of [tf.Variable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Variable) to update to minimize loss. Defaults to the list of variables collected in the graph under the key GraphKey.TRAINABLE\_VARIABLES.
* **\*\*kwargs**: Keyword arguments for compute\_gradients().

#### Returns:

A list of (gradient, variable) pairs.

#### Raises:

* **ValueError**: If not within a tpu\_shard\_context or group\_assignment is invalid.

### get\_name

get\_name()

### get\_slot

get\_slot(  
    \*args,  
    \*\*kwargs  
)

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

This simply wraps the get\_slot() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

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

### get\_slot\_names

get\_slot\_names(  
    \*args,  
    \*\*kwargs  
)

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

This simply wraps the get\_slot\_names() from the actual optimizer.

#### Args:

* **\*args**: Arguments for get\_slot().
* **\*\*kwargs**: Keyword arguments for get\_slot().

#### Returns:

A list of strings.

### minimize

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

Add operations to minimize loss by updating var\_list.

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

#### Args:

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

#### Returns:

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

#### Raises:

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

#### Eager Compatibility

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

### variables

variables()

Forwarding the variables from the underlying optimizer.

## Class Members

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

# tf.compat.v1.tpu.cross\_replica\_sum

Sum the input tensor across replicas according to group\_assignment.

tf.compat.v1.tpu.cross\_replica\_sum(  
    x,  
    group\_assignment=None,  
    name=None  
)

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

#### Args:

* **x**: The local tensor to the sum.
* **group\_assignment**: Optional 2d int32 lists with shape [num\_groups, num\_replicas\_per\_group]. group\_assignment[i] represents the replica ids in the ith subgroup.
* **name**: Optional op name.

#### Returns:

A Tensor which is summed across replicas.

# tf.compat.v1.tpu.initialize\_system

Initializes a distributed TPU system for use with TensorFlow.

tf.compat.v1.tpu.initialize\_system(  
    embedding\_config=None,  
    job=None  
)

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

#### Args:

* **embedding\_config**: If not None, a TPUEmbeddingConfiguration proto describing the desired configuration of the hardware embedding lookup tables. If embedding\_config is None, no hardware embeddings can be used.
* **job**: The job (the XXX in TensorFlow device specification /job:XXX) that contains the TPU devices that will be initialized. If job=None it is assumed there is only one job in the TensorFlow flock, and an error will be returned if this assumption does not hold.

#### Returns:

A serialized TopologyProto that describes the TPU system. Note: the topology must be evaluated using Session.run before it can be used.

# tf.compat.v1.tpu.outside\_compilation

Builds part of a computation outside any current TPU replicate scope.

tf.compat.v1.tpu.outside\_compilation(  
    computation,  
    \*args,  
    \*\*kwargs  
)

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

#### Args:

* **computation**: A Python function that builds the computation to place on the host.
* **\*args**: the positional arguments for the computation.
* **\*\*kwargs**: the keyword arguments for the computation.

#### Returns:

The Tensors returned by computation.

# tf.compat.v1.tpu.replicate

Builds a graph operator that runs a replicated TPU computation.

tf.compat.v1.tpu.replicate(  
    computation,  
    inputs=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None,  
    maximum\_shapes=None  
)

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

#### Args:

* **computation**: A Python function that builds the computation to replicate.
* **inputs**: A list of lists of input tensors or None (equivalent to [[]]), indexed by [replica\_num][input\_num]. All replicas must have the same number of inputs. Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimention list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each replica of the computation uses only one core, and there is either only one replica, or the number of replicas is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.
* **maximum\_shapes**: A nested structure of tf.TensorShape representing the shape to which the respective component of each input element in each replica should be padded. Any unknown dimensions (e.g. tf.compat.v1.Dimension(None) in a tf.TensorShape or -1 in a tensor-like object) will be padded to the maximum size of that dimension over all replicas. Note that if the input dimension is already static, we won't do padding on it and we require the maximum\_shapes to have the same value or None on that dimension. The structure of maximum\_shapes needs to be the same as inputs[0].

#### Returns:

A list of outputs, indexed by [replica\_num] each output can be a nested structure same as what computation() returns with a few exceptions.

Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

#### Raises:

* **ValueError**: If all replicas do not have equal numbers of input tensors.
* **ValueError**: If the number of inputs per replica does not match the number of formal parameters to computation.
* **ValueError**: If the static inputs dimensions don't match with the values given in maximum\_shapes.
* **ValueError**: If the structure of inputs per replica does not match the structure of maximum\_shapes.

# tf.compat.v1.tpu.rewrite

Rewrites computation for execution on a TPU system.

tf.compat.v1.tpu.rewrite(  
    computation,  
    inputs=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

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

#### Args:

* **computation**: A Python function that builds a computation to apply to the input. If the function takes n inputs, 'inputs' should be a list of n tensors.

computation may return a list of operations and tensors. Tensors must come before operations in the returned list. The return value of rewrite is a list of tensors corresponding to the tensors from the output of computation.

All Operations constructed during computation will be executed when evaluating any of the returned output tensors, not just the ones returned.

* **inputs**: A list of input tensors or None (equivalent to an empty list). Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimention list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).
* **infeed\_queue**: If not None, the InfeedQueue from which to append a tuple of arguments as inputs to computation.
* **device\_assignment**: if not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. May be omitted for a single-core computation, in which case the core attached to task 0, TPU device 0 is used.
* **name**: (Deprecated) Does nothing.

#### Returns:

Same data structure as if computation(\*inputs) is called directly with some exceptions for correctness. Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

# tf.compat.v1.tpu.shard

Shards computation for parallel execution.

tf.compat.v1.tpu.shard(  
    computation,  
    inputs=None,  
    num\_shards=1,  
    input\_shard\_axes=None,  
    outputs\_from\_all\_shards=True,  
    output\_shard\_axes=None,  
    infeed\_queue=None,  
    device\_assignment=None,  
    name=None  
)

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

inputs must be a list of Tensors or None (equivalent to an empty list), each of which has a corresponding split axis (from input\_shard\_axes). Each input is split into num\_shards pieces along the corresponding axis, and computation is applied to each shard in parallel.

Tensors are broadcast to all shards if they are lexically captured by computation. e.g.,

x = tf.constant(7) def computation(): return x + 3 ... = shard(computation, ...)

TODO(phawkins): consider adding support for broadcasting Tensors passed as inputs.

If outputs\_from\_all\_shards is true, the outputs from all shards of computation are concatenated back together along their output\_shards\_axes. Otherwise, each output is taken from an arbitrary shard.

Inputs and outputs of the computation must be at least rank-1 Tensors.

#### Args:

* **computation**: A Python function that builds a computation to apply to each shard of the input.
* **inputs**: A list of input tensors or None (equivalent to an empty list). Each input tensor has a corresponding shard axes, given by input\_shard\_axes, which must have size divisible by num\_shards.
* **num\_shards**: The number of shards.
* **input\_shard\_axes**: A list of dimensions along which to shard inputs, or None. None means "shard all inputs along dimension 0". If not None, there must be one dimension per input.
* **outputs\_from\_all\_shards**: Boolean or list of boolean. For each output, if True, outputs from all shards are concatenated along the corresponding output\_shard\_axes entry. Otherwise, each output is taken from an arbitrary shard. If the argument is a boolean, the argument's value is used for each output.
* **output\_shard\_axes**: A list of dimensions along which to concatenate the outputs of computation, or None. None means "concatenate all outputs along dimension 0". If not None, there must be one dimension per output. Ignored if outputs\_from\_all\_shards is False.
* **infeed\_queue**: If not None, the InfeedQueue to use to augment the inputs of computation.
* **device\_assignment**: If not None, a DeviceAssignment describing the mapping between logical cores in the computation with physical cores in the TPU topology. Uses a default device assignment if None. The DeviceAssignment may be omitted if each shard of the computation uses only one core, and there is either only one shard, or the number of shards is equal to the number of cores in the TPU system.
* **name**: (Deprecated) Does nothing.

#### Returns:

A list of output tensors.

#### Raises:

* **ValueError**: If num\_shards <= 0
* **ValueError**: If len(input\_shard\_axes) != len(inputs)
* **ValueError**: If len(output\_shard\_axes) != len(outputs from computation)

# tf.compat.v1.tpu.shutdown\_system

Shuts down a running a distributed TPU system.

tf.compat.v1.tpu.shutdown\_system(job=None)

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

#### Args:

* **job**: The job (the XXX in TensorFlow device specification /job:XXX) that contains the TPU devices that will be shutdown. If job=None it is assumed there is only one job in the TensorFlow flock, and an error will be returned if this assumption does not hold.

Module: tf.tpu.experimental

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

Public API for tf.tpu.experimental namespace.

Classes

[class DeviceAssignment](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment): Mapping from logical cores in a computation to the physical TPU topology.

Functions

[initialize\_tpu\_system(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/initialize_tpu_system): Initialize the TPU devices.

# tf.tpu.experimental.DeviceAssignment

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#top_of_page)
* [Class DeviceAssignment](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#class_deviceassignment)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/tpu/experimental/DeviceAssignment#properties)

## Class DeviceAssignment

Mapping from logical cores in a computation to the physical TPU topology.

### Aliases:

* Class tf.compat.v1.tpu.experimental.DeviceAssignment
* Class tf.compat.v2.tpu.experimental.DeviceAssignment
* Class tf.tpu.experimental.DeviceAssignment

Defined in [python/tpu/device\_assignment.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/tpu/device_assignment.py).

Prefer to use the DeviceAssignment.build() helper to construct a DeviceAssignment; it is easier if less flexible than constructing a DeviceAssignment directly.

## \_\_init\_\_

\_\_init\_\_(  
    topology,  
    core\_assignment  
)

Constructs a DeviceAssignment object.

#### Args:

* **topology**: A Topology object that describes the physical TPU topology.
* **core\_assignment**: A logical to physical core mapping, represented as a rank 3 numpy array. See the description of the core\_assignment property for more details.

#### Raises:

* **ValueError**: If topology is not Topology object.
* **ValueError**: If core\_assignment is not a rank 3 numpy array.

## Properties

### core\_assignment

The logical to physical core mapping.

#### Returns:

An integer numpy array of rank 3, with shape [num\_replicas, num\_cores\_per\_replica, topology\_rank]. Maps (replica, logical core) pairs to physical topology coordinates.

### num\_cores\_per\_replica

The number of cores per replica.

### num\_replicas

The number of replicas of the computation.

### topology

A Topology that describes the TPU topology.

## Methods

### build

@staticmethod  
build(  
    topology,  
    computation\_shape=None,  
    computation\_stride=None,  
    num\_replicas=1  
)

### coordinates

coordinates(  
    replica,  
    logical\_core  
)

Returns the physical topology coordinates of a logical core.

### host\_device

host\_device(  
    replica=0,  
    logical\_core=0,  
    job=None  
)

Returns the CPU device attached to a logical core.

### lookup\_replicas

lookup\_replicas(  
    task\_id,  
    logical\_core  
)

Lookup replica ids by task number and logical core.

#### Args:

* **task\_id**: TensorFlow task number.
* **logical\_core**: An integer, identifying a logical core.

#### Returns:

A sorted list of the replicas that are attached to that task and logical\_core.

#### Raises:

* **ValueError**: If no replica exists in the task which contains the logical core.

### tpu\_device

tpu\_device(  
    replica=0,  
    logical\_core=0,  
    job=None  
)

Returns the name of the TPU device assigned to a logical core.

### tpu\_ordinal

tpu\_ordinal(  
    replica=0,  
    logical\_core=0  
)

Returns the ordinal of the TPU device assigned to a logical core.

# tf.tpu.experimental.initialize\_tpu\_system

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

Initialize the TPU devices.

### Aliases:

* tf.compat.v1.tpu.experimental.initialize\_tpu\_system
* tf.compat.v2.tpu.experimental.initialize\_tpu\_system
* tf.tpu.experimental.initialize\_tpu\_system

tf.tpu.experimental.initialize\_tpu\_system(cluster\_resolver=None)

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

#### Args:

* **cluster\_resolver**: A tf.distribute.cluster\_resolver.TPUClusterResolver, which provides information about the TPU cluster.

#### Returns:

The tf.tpu.Topology object for the topology of the TPU cluster.

#### Raises:

* **RuntimeError**: If no TPU devices found for eager execution.

Module: tf.xla.experimental

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

Public API for tf.xla.experimental namespace.

Functions

[compile(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/compile): Builds an operator that compiles and runs computation with XLA.

[jit\_scope(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/xla/experimental/jit_scope): Enable or disable JIT compilation of operators within the scope.

# tf.xla.experimental.compile

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

Builds an operator that compiles and runs computation with XLA.

### Aliases:

* tf.compat.v1.xla.experimental.compile
* tf.compat.v2.xla.experimental.compile
* tf.xla.experimental.compile

tf.xla.experimental.compile(  
    computation,  
    inputs=None  
)

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

NOTE: In eager mode, computation will have @tf.function semantics.

#### Args:

* **computation**: A Python function that builds a computation to apply to the input. If the function takes n inputs, 'inputs' should be a list of n tensors.

computation may return a list of operations and tensors. Tensors must come before operations in the returned list. The return value of compile is a list of tensors corresponding to the tensors from the output of computation.

All Operations returned from computation will be executed when evaluating any of the returned output tensors.

* **inputs**: A list of inputs or None (equivalent to an empty list). Each input can be a nested structure containing values that are convertible to tensors. Note that passing an N-dimension list of compatible values will result in a N-dimension list of scalar tensors rather than a single Rank-N tensors. If you need different behavior, convert part of inputs to tensors with [tf.convert\_to\_tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).

#### Returns:

Same data structure as if computation(\*inputs) is called directly with some exceptions for correctness. Exceptions include: 1) None output: a NoOp would be returned which control-depends on computation. 2) Single value output: A tuple containing the value would be returned. 3) Operation-only outputs: a NoOp would be returned which control-depends on computation. TODO(b/121383831): Investigate into removing these special cases.

#### Raises:

* **RuntimeError**: if called when eager execution is enabled.

# tf.xla.experimental.jit\_scope

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

Enable or disable JIT compilation of operators within the scope.

### Aliases:

* tf.compat.v1.xla.experimental.jit\_scope
* tf.compat.v2.xla.experimental.jit\_scope
* tf.xla.experimental.jit\_scope

tf.xla.experimental.jit\_scope(  
    \*args,  
    \*\*kwds  
)

NOTE: This is an experimental feature.

The compilation is a hint and only supported on a best-effort basis.

#### Example usage:

with tf.xla.experimental.jit\_scope(): c = tf.matmul(a, b) # compiled with tf.xla.experimental.jit\_scope(compile\_ops=False): d = tf.matmul(a, c) # not compiled with tf.xla.experimental.jit\_scope( compile\_ops=lambda node\_def: 'matmul' in node\_def.op.lower()): e = tf.matmul(a, b) + d # matmul is compiled, the addition is not.

Example of separate\_compiled\_gradients: # In the example below, the computations for f, g and h will all be compiled # in separate scopes. with tf.xla.experimental.jit\_scope( separate\_compiled\_gradients=True): f = tf.matmul(a, b) g = tf.gradients([f], [a, b], name='mygrads1') h = tf.gradients([f], [a, b], name='mygrads2')

#### Args:

* **compile\_ops**: Whether to enable or disable compilation in the scope. Either a Python bool, or a callable that accepts the parameter node\_def and returns a python bool.
* **separate\_compiled\_gradients**: If true put each gradient subgraph into a separate compilation scope. This gives fine-grained control over which portions of the graph will be compiled as a single unit. Compiling gradients separately may yield better performance for some graphs. The scope is named based on the scope of the forward computation as well as the name of the gradients. As a result, the gradients will be compiled in a scope that is separate from both the forward computation, and from other gradients.

#### Raises:

* **RuntimeError**: if called when eager execution is enabled.

#### Yields:

The current scope, enabling or disabling compilation.

# tf.compat.v1.user\_ops.my\_fact

Example of overriding the generated code for an Op.

tf.compat.v1.user\_ops.my\_fact()

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

# Module: tf.math

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#top_of_page)
* [About Segmentation](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#about_segmentation)
* [Functions](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math#functions)

Math Operations.

**Note:** Functions taking **Tensor** arguments can also take anything accepted by [**tf.convert\_to\_tensor**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/convert_to_tensor).**Note:** Elementwise binary operations in TensorFlow follow [numpy-style broadcasting](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html).

TensorFlow provides a variety of math functions including:

* Basic arithmetic operators and trigonometric functions.
* Special math functions (like: [tf.math.igamma](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma) and [tf.math.zeta](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta))
* Complex number functions (like: [tf.math.imag](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag) and [tf.math.angle](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle))
* Reductions and scans (like: [tf.math.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean) and [tf.math.cumsum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum))
* Segment functions (like: [tf.math.segment\_sum](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum))

See: [tf.linalg](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/linalg) for matrix and tensor functions.

## About Segmentation

TensorFlow provides several operations that you can use to perform common math computations on tensor segments. Here a segmentation is a partitioning of a tensor along the first dimension, i.e. it defines a mapping from the first dimension onto segment\_ids. The segment\_ids tensor should be the size of the first dimension, d0, with consecutive IDs in the range 0 to k, where k<d0. In particular, a segmentation of a matrix tensor is a mapping of rows to segments.

#### For example:

c = tf.constant([[1,2,3,4], [-1,-2,-3,-4], [5,6,7,8]])  
tf.math.segment\_sum(c, tf.constant([0, 0, 1]))  
#  ==>  [[0 0 0 0]  
#        [5 6 7 8]]

The standard segment\_\* functions assert that the segment indices are sorted. If you have unsorted indices use the equivalent unsorted\_segment\_ function. Thses functions take an additional argument num\_segments so that the output tensor can be efficiently allocated.

c = tf.constant([[1,2,3,4], [-1,-2,-3,-4], [5,6,7,8]])  
tf.math.unsorted\_segment\_sum(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 6,  8, 10, 12],  
#       [-1, -2, -3, -4]]

## Functions

[abs(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/abs): Computes the absolute value of a tensor.

[accumulate\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n): Returns the element-wise sum of a list of tensors.

[acos(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acos): Computes acos of x element-wise.

[acosh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/acosh): Computes inverse hyperbolic cosine of x element-wise.

[add(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add): Returns x + y element-wise.

[add\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n): Adds all input tensors element-wise.

[angle(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/angle): Returns the element-wise argument of a complex (or real) tensor.

[argmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmax): Returns the index with the largest value across axes of a tensor.

[argmin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/argmin): Returns the index with the smallest value across axes of a tensor.

[asin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin): Computes the trignometric inverse sine of x element-wise.

[asinh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asinh): Computes inverse hyperbolic sine of x element-wise.

[atan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan): Computes the trignometric inverse tangent of x element-wise.

[atan2(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2): Computes arctangent of y/x element-wise, respecting signs of the arguments.

[atanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atanh): Computes inverse hyperbolic tangent of x element-wise.

[bessel\_i0(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0): Computes the Bessel i0 function of x element-wise.

[bessel\_i0e(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e): Computes the Bessel i0e function of x element-wise.

[bessel\_i1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1): Computes the Bessel i1 function of x element-wise.

[bessel\_i1e(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e): Computes the Bessel i1e function of x element-wise.

[betainc(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/betainc): Compute the regularized incomplete beta integral Ix(a,b).

[bincount(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bincount): Counts the number of occurrences of each value in an integer array.

[ceil(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/ceil): Returns element-wise smallest integer not less than x.

[confusion\_matrix(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/confusion_matrix): Computes the confusion matrix from predictions and labels.

[conj(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/conj): Returns the complex conjugate of a complex number.

[cos(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cos): Computes cos of x element-wise.

[cosh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cosh): Computes hyperbolic cosine of x element-wise.

[count\_nonzero(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/count_nonzero): Computes number of nonzero elements across dimensions of a tensor.

[cumprod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumprod): Compute the cumulative product of the tensor x along axis.

[cumsum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/cumsum): Compute the cumulative sum of the tensor x along axis.

[digamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/digamma): Computes Psi, the derivative of Lgamma (the log of the absolute value of

[divide(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide): Computes Python style division of x by y.

[divide\_no\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/divide_no_nan): Computes an unsafe divide which returns 0 if the y is zero.

[equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/equal): Returns the truth value of (x == y) element-wise.

[erf(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erf): Computes the Gauss error function of x element-wise.

[erfc(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/erfc): Computes the complementary error function of x element-wise.

[exp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/exp): Computes exponential of x element-wise. y=ex.

[expm1(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1): Computes exponential of x - 1 element-wise.

[floor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floor): Returns element-wise largest integer not greater than x.

[floordiv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv): Divides x / y elementwise, rounding toward the most negative integer.

[floormod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod): Returns element-wise remainder of division. When x < 0 xor y < 0 is

[greater(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater): Returns the truth value of (x > y) element-wise.

[greater\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/greater_equal): Returns the truth value of (x >= y) element-wise.

[igamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igamma): Compute the lower regularized incomplete Gamma function P(a, x).

[igammac(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/igammac): Compute the upper regularized incomplete Gamma function Q(a, x).

[imag(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/imag): Returns the imaginary part of a complex (or real) tensor.

[in\_top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/in_top_k): Says whether the targets are in the top K predictions.

[invert\_permutation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/invert_permutation): Computes the inverse permutation of a tensor.

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

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

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

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

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

[l2\_normalize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize): Normalizes along dimension axis using an L2 norm.

[lbeta(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lbeta): Computes ln(|Beta(x)|), reducing along the last dimension.

[less(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less): Returns the truth value of (x < y) element-wise.

[less\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/less_equal): Returns the truth value of (x <= y) element-wise.

[lgamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/lgamma): Computes the log of the absolute value of Gamma(x) element-wise.

[log(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log): Computes natural logarithm of x element-wise.

[log1p(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p): Computes natural logarithm of (1 + x) element-wise.

[log\_sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log_sigmoid): Computes log sigmoid of x element-wise.

[log\_softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/log_softmax): Computes log softmax activations.

[logical\_and(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_and): Returns the truth value of x AND y element-wise.

[logical\_not(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_not): Returns the truth value of NOT x element-wise.

[logical\_or(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_or): Returns the truth value of x OR y element-wise.

[logical\_xor(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/logical_xor): Logical XOR function.

[maximum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/maximum): Returns the max of x and y (i.e. x > y ? x : y) element-wise.

[minimum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/minimum): Returns the min of x and y (i.e. x < y ? x : y) element-wise.

[mod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floormod): Returns element-wise remainder of division. When x < 0 xor y < 0 is

[multiply(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply): Returns x \* y element-wise.

[multiply\_no\_nan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/multiply_no_nan): Computes the product of x and y and returns 0 if the y is zero, even if x is NaN or infinite.

[negative(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/negative): Computes numerical negative value element-wise.

[nextafter(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/nextafter): Returns the next representable value of x1 in the direction of x2, element-wise.

[not\_equal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/not_equal): Returns the truth value of (x != y) element-wise.

[polygamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polygamma): Compute the polygamma function ψ(n)(x).

[polyval(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/polyval): Computes the elementwise value of a polynomial.

[pow(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/pow): Computes the power of one value to another.

[real(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/real): Returns the real part of a complex (or real) tensor.

[reciprocal(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reciprocal): Computes the reciprocal of x element-wise.

[reduce\_all(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reduce_all): Computes the "logical and" of elements across dimensions of a tensor.

[reduce\_any(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_any): Computes the "logical or" of elements across dimensions of a tensor.

[reduce\_euclidean\_norm(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_euclidean_norm): Computes the Euclidean norm of elements across dimensions of a tensor.

[reduce\_logsumexp(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_logsumexp): Computes log(sum(exp(elements across dimensions of a tensor))).

[reduce\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_max): Computes the maximum of elements across dimensions of a tensor.

[reduce\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean): Computes the mean of elements across dimensions of a tensor.

[reduce\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_min): Computes the minimum of elements across dimensions of a tensor.

[reduce\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_prod): Computes the product of elements across dimensions of a tensor.

[reduce\_std(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_std): Computes the standard deviation of elements across dimensions of a tensor.

[reduce\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_sum): Computes the sum of elements across dimensions of a tensor.

[reduce\_variance(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_variance): Computes the variance of elements across dimensions of a tensor.

[rint(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rint): Returns element-wise integer closest to x.

[round(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/round): Rounds the values of a tensor to the nearest integer, element-wise.

[rsqrt(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/rsqrt): Computes reciprocal of square root of x element-wise.

[scalar\_mul(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/scalar_mul): Multiplies a scalar times a Tensor or IndexedSlices object.

[segment\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_max): Computes the maximum along segments of a tensor.

[segment\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_mean): Computes the mean along segments of a tensor.

[segment\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_min): Computes the minimum along segments of a tensor.

[segment\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_prod): Computes the product along segments of a tensor.

[segment\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/segment_sum): Computes the sum along segments of a tensor.

[sigmoid(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sigmoid): Computes sigmoid of x element-wise.

[sign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sign): Returns an element-wise indication of the sign of a number.

[sin(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin): Computes sin of x element-wise.

[sinh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sinh): Computes hyperbolic sine of x element-wise.

[softmax(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softmax): Computes softmax activations.

[softplus(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/softplus): Computes softplus: log(exp(features) + 1).

[softsign(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/nn/softsign): Computes softsign: features / (abs(features) + 1).

[sqrt(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sqrt): Computes square root of x element-wise.

[square(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/square): Computes square of x element-wise.

[squared\_difference(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/squared_difference): Returns (x - y)(x - y) element-wise.

[subtract(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/subtract): Returns x - y element-wise.

[tan(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan): Computes tan of x element-wise.

[tanh(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tanh): Computes hyperbolic tangent of x element-wise.

[top\_k(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/top_k): Finds values and indices of the k largest entries for the last dimension.

[truediv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/truediv): Divides x / y elementwise (using Python 3 division operator semantics).

[unsorted\_segment\_max(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_max): Computes the maximum along segments of a tensor.

[unsorted\_segment\_mean(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_mean): Computes the mean along segments of a tensor.

[unsorted\_segment\_min(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_min): Computes the minimum along segments of a tensor.

[unsorted\_segment\_prod(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_prod): Computes the product along segments of a tensor.

[unsorted\_segment\_sqrt\_n(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sqrt_n): Computes the sum along segments of a tensor divided by the sqrt(N).

[unsorted\_segment\_sum(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/unsorted_segment_sum): Computes the sum along segments of a tensor.

[xdivy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xdivy): Returns 0 if x == 0, and x / y otherwise, elementwise.

[xlogy(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/xlogy): Returns 0 if x == 0, and x \* log(y) otherwise, elementwise.

[zero\_fraction(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zero_fraction): Returns the fraction of zeros in value.

[zeta(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/zeta): Compute the Hurwitz zeta function ζ(x,q).

# tf.compat.v1.math.log\_softmax

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

Computes log softmax activations. (deprecated arguments)

### Aliases:

* tf.compat.v1.math.log\_softmax
* tf.compat.v1.nn.log\_softmax

tf.compat.v1.math.log\_softmax(  
    logits,  
    axis=None,  
    name=None,  
    dim=None  
)

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

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dim)**. They will be removed in a future version. Instructions for updating: dim is deprecated, use axis instead

For each batch i and class j we have

logsoftmax = logits - log(reduce\_sum(exp(logits), axis))

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).
* **dim**: Deprecated alias for axis.

#### Returns:

A Tensor. Has the same type as logits. Same shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.compat.v1.math.softmax

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

Computes softmax activations. (deprecated arguments)

### Aliases:

* tf.compat.v1.math.softmax
* tf.compat.v1.nn.softmax

tf.compat.v1.math.softmax(  
    logits,  
    axis=None,  
    name=None,  
    dim=None  
)

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

**Warning:** SOME ARGUMENTS ARE DEPRECATED: **(dim)**. They will be removed in a future version. Instructions for updating: dim is deprecated, use axis instead

This function performs the equivalent of

softmax = tf.exp(logits) / tf.reduce\_sum(tf.exp(logits), axis)

#### Args:

* **logits**: A non-empty Tensor. Must be one of the following types: half, float32, float64.
* **axis**: The dimension softmax would be performed on. The default is -1 which indicates the last dimension.
* **name**: A name for the operation (optional).
* **dim**: Deprecated alias for axis.

#### Returns:

A Tensor. Has the same type and shape as logits.

#### Raises:

* **InvalidArgumentError**: if logits is empty or axis is beyond the last dimension of logits.

# tf.math.abs

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

Computes the absolute value of a tensor.

### Aliases:

* tf.RaggedTensor.\_\_abs\_\_
* tf.Tensor.\_\_abs\_\_
* tf.abs
* tf.compat.v1.RaggedTensor.\_\_abs\_\_
* tf.compat.v1.Tensor.\_\_abs\_\_
* tf.compat.v1.abs
* tf.compat.v1.math.abs
* tf.compat.v2.RaggedTensor.\_\_abs\_\_
* tf.compat.v2.Tensor.\_\_abs\_\_
* tf.compat.v2.abs
* tf.compat.v2.math.abs
* tf.math.abs

tf.math.abs(  
    x,  
    name=None  
)

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

### Used in the guide:

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

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

Given a tensor of integer or floating-point values, this operation returns a tensor of the same type, where each element contains the absolute value of the corresponding element in the input.

Given a tensor x of complex numbers, this operation returns a tensor of type float32 or float64that is the absolute value of each element in x. All elements in x must be complex numbers of the form a+bj. The absolute value is computed as a2+b2. For example:

x = tf.constant([[-2.25 + 4.75j], [-3.25 + 5.75j]])  
tf.abs(x)  # [5.25594902, 6.60492229]

#### Args:

* **x**: A Tensor or SparseTensor of type float16, float32, float64, int32, int64, complex64 or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor the same size, type, and sparsity as x with absolute values. Note, for complex64 or complex128 input, the returned Tensor will be of type float32 or float64, respectively.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.abs(x.values, ...), x.dense\_shape)

# tf.math.accumulate\_n

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

Returns the element-wise sum of a list of tensors.

### Aliases:

* tf.compat.v1.accumulate\_n
* tf.compat.v1.math.accumulate\_n
* tf.compat.v2.math.accumulate\_n
* tf.math.accumulate\_n

tf.math.accumulate\_n(  
    inputs,  
    shape=None,  
    tensor\_dtype=None,  
    name=None  
)

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

Optionally, pass shape and tensor\_dtype for shape and type checking, otherwise, these are inferred.

accumulate\_n performs the same operation as [tf.math.add\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n), but does not wait for all of its inputs to be ready before beginning to sum. This approach can save memory if inputs are ready at different times, since minimum temporary storage is proportional to the output size rather than the inputs' size.

accumulate\_n is differentiable (but wasn't previous to TensorFlow 1.7).

#### For example:

a = tf.constant([[1, 2], [3, 4]])  
b = tf.constant([[5, 0], [0, 6]])  
tf.math.accumulate\_n([a, b, a])  # [[7, 4], [6, 14]]  
  
# Explicitly pass shape and type  
tf.math.accumulate\_n([a, b, a], shape=[2, 2], tensor\_dtype=tf.int32)  
                                                               # [[7,  4],  
                                                               #  [6, 14]]

#### Args:

* **inputs**: A list of Tensor objects, each with same shape and type.
* **shape**: Expected shape of elements of inputs (optional). Also controls the output shape of this op, which may affect type inference in other ops. A value of None means "infer the input shape from the shapes in inputs".
* **tensor\_dtype**: Expected data type of inputs (optional). A value of None means "infer the input dtype from inputs[0]".
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as the elements of inputs.

#### Raises:

* **ValueError**: If inputs don't all have same shape and dtype or the shape cannot be inferred.

# tf.math.acos

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

Computes acos of x element-wise.

### Aliases:

* tf.acos
* tf.compat.v1.acos
* tf.compat.v1.math.acos
* tf.compat.v2.acos
* tf.compat.v2.math.acos
* tf.math.acos

tf.math.acos(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.acosh

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

Computes inverse hyperbolic cosine of x element-wise.

### Aliases:

* tf.acosh
* tf.compat.v1.acosh
* tf.compat.v1.math.acosh
* tf.compat.v2.acosh
* tf.compat.v2.math.acosh
* tf.math.acosh

tf.math.acosh(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.add

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

Returns x + y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_add\_\_
* tf.add
* tf.compat.v1.RaggedTensor.\_\_add\_\_
* tf.compat.v1.add
* tf.compat.v1.math.add
* tf.compat.v2.RaggedTensor.\_\_add\_\_
* tf.compat.v2.add
* tf.compat.v2.math.add
* tf.math.add

tf.math.add(  
    x,  
    y,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)

### Used in the tutorials:

* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

NOTE: math.add supports broadcasting. AddN does not. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, complex128, string.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.add\_n

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

Adds all input tensors element-wise.

### Aliases:

* tf.add\_n
* tf.compat.v1.add\_n
* tf.compat.v1.math.add\_n
* tf.compat.v2.add\_n
* tf.compat.v2.math.add\_n
* tf.math.add\_n

tf.math.add\_n(  
    inputs,  
    name=None  
)

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

### Used in the guide:

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

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Converts IndexedSlices objects into dense tensors prior to adding.

[tf.math.add\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add_n) performs the same operation as [tf.math.accumulate\_n](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/accumulate_n), but it waits for all of its inputs to be ready before beginning to sum. This buffering can result in higher memory consumption when inputs are ready at different times, since the minimum temporary storage required is proportional to the input size rather than the output size.

This op does not [broadcast](https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html) its inputs. If you need broadcasting, use [tf.math.add](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/add) (or the + operator) instead.

#### For example:

a = tf.constant([[3, 5], [4, 8]])  
b = tf.constant([[1, 6], [2, 9]])  
tf.math.add\_n([a, b, a])  # [[7, 16], [10, 25]]

#### Args:

* **inputs**: A list of [tf.Tensor](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/Tensor) or [tf.IndexedSlices](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/IndexedSlices) objects, each with same shape and type.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as the elements of inputs.

#### Raises:

* **ValueError**: If inputs don't all have same shape and dtype or the shape cannot be inferred.

# tf.math.angle

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

Returns the element-wise argument of a complex (or real) tensor.

### Aliases:

* tf.compat.v1.angle
* tf.compat.v1.math.angle
* tf.compat.v2.math.angle
* tf.math.angle

tf.math.angle(  
    input,  
    name=None  
)

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

Given a tensor input, this operation returns a tensor of type float that is the argument of each element in input considered as a complex number.

The elements in input are considered to be complex numbers of the form a+bj, where a is the real part and b is the imaginary part. If input is real then b is zero by definition.

The argument returned by this function is of the form atan2(b,a). If input is real, a tensor of all zeros is returned.

#### For example:

input = tf.constant([-2.25 + 4.75j, 3.25 + 5.75j], dtype=tf.complex64)  
tf.math.angle(input).numpy()  
# ==> array([2.0131705, 1.056345 ], dtype=float32)

#### Args:

* **input**: A Tensor. Must be one of the following types: float, double, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32 or float64.

# tf.math.argmax

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

Returns the index with the largest value across axes of a tensor.

### Aliases:

* tf.argmax
* tf.compat.v2.argmax
* tf.compat.v2.math.argmax
* tf.math.argmax

tf.math.argmax(  
    input,  
    axis=None,  
    output\_type=tf.dtypes.int64,  
    name=None  
)

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

### Used in the guide:

* [Convert Your Existing Code to TensorFlow 2.0](https://www.tensorflow.org/beta/guide/migration_guide)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Custom training: walkthrough](https://www.tensorflow.org/beta/tutorials/eager/custom_training_walkthrough)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Note that in case of ties the identity of the return value is not guaranteed.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **axis**: A Tensor. Must be one of the following types: int32, int64. int32 or int64, must be in the range -rank(input), rank(input)). Describes which axis of the input Tensor to reduce across. For vectors, use axis = 0.
* **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.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

#### Usage:

import tensorflow as tf  
a = [1, 10, 26.9, 2.8, 166.32, 62.3]  
b = tf.math.argmax(input = a)  
c = tf.keras.backend.eval(b)  
# c = 4  
# here a[4] = 166.32 which is the largest element of a across axis 0

# tf.math.argmin

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

Returns the index with the smallest value across axes of a tensor.

### Aliases:

* tf.argmin
* tf.compat.v2.argmin
* tf.compat.v2.math.argmin
* tf.math.argmin

tf.math.argmin(  
    input,  
    axis=None,  
    output\_type=tf.dtypes.int64,  
    name=None  
)

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

Note that in case of ties the identity of the return value is not guaranteed.

#### Args:

* **input**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **axis**: A Tensor. Must be one of the following types: int32, int64. int32 or int64, must be in the range -rank(input), rank(input)). Describes which axis of the input Tensor to reduce across. For vectors, use axis = 0.
* **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.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type output\_type.

#### Usage:

import tensorflow as tf  
a = [1, 10, 26.9, 2.8, 166.32, 62.3]  
b = tf.math.argmin(input = a)  
c = tf.keras.backend.eval(b)  
# c = 0  
# here a[0] = 1 which is the smallest element of a across axis 0

# tf.math.asin

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

Computes the trignometric inverse sine of x element-wise.

### Aliases:

* tf.asin
* tf.compat.v1.asin
* tf.compat.v1.math.asin
* tf.compat.v2.asin
* tf.compat.v2.math.asin
* tf.math.asin

tf.math.asin(  
    x,  
    name=None  
)

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

The [tf.math.asin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin) operation returns the inverse of [tf.math.sin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/sin), such that if y = tf.math.sin(x) then, x = tf.math.asin(y).

**Note**: The output of [tf.math.asin](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/asin) will lie within the invertible range of sine, i.e [-pi/2, pi/2].

#### For example:

# Note: [1.047, 0.785] ~= [(pi/3), (pi/4)]  
x = tf.constant([1.047, 0.785])  
y = tf.math.sin(x) # [0.8659266, 0.7068252]  
  
tf.math.asin(y) # [1.047, 0.785] = x

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.asinh

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

Computes inverse hyperbolic sine of x element-wise.

### Aliases:

* tf.asinh
* tf.compat.v1.asinh
* tf.compat.v1.math.asinh
* tf.compat.v2.asinh
* tf.compat.v2.math.asinh
* tf.math.asinh

tf.math.asinh(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.atan

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

Computes the trignometric inverse tangent of x element-wise.

### Aliases:

* tf.atan
* tf.compat.v1.atan
* tf.compat.v1.math.atan
* tf.compat.v2.atan
* tf.compat.v2.math.atan
* tf.math.atan

tf.math.atan(  
    x,  
    name=None  
)

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

The [tf.math.atan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan) operation returns the inverse of [tf.math.tan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/tan), such that if y = tf.math.tan(x) then, x = tf.math.atan(y).

**Note**: The output of [tf.math.atan](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan) will lie within the invertible range of tan, i.e (-pi/2, pi/2).

#### For example:

# Note: [1.047, 0.785] ~= [(pi/3), (pi/4)]  
x = tf.constant([1.047, 0.785])  
y = tf.math.tan(x) # [1.731261, 0.99920404]  
  
tf.math.atan(y) # [1.047, 0.785] = x

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.atan2

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/atan2#aliases)

Computes arctangent of y/x element-wise, respecting signs of the arguments.

### Aliases:

* tf.atan2
* tf.compat.v1.atan2
* tf.compat.v1.math.atan2
* tf.compat.v2.atan2
* tf.compat.v2.math.atan2
* tf.math.atan2

tf.math.atan2(  
    y,  
    x,  
    name=None  
)

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

This is the angle ( \theta \in [-\pi, \pi] ) such that [ x = r \cos(\theta) ] and [ y = r \sin(\theta) ] where (r = \sqrt(x^2 + y^2) ).

#### Args:

* **y**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **x**: A Tensor. Must have the same type as y.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as y.

# tf.math.atanh

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

Computes inverse hyperbolic tangent of x element-wise.

### Aliases:

* tf.atanh
* tf.compat.v1.atanh
* tf.compat.v1.math.atanh
* tf.compat.v2.atanh
* tf.compat.v2.math.atanh
* tf.math.atanh

tf.math.atanh(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.bessel\_i0

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0#aliases)

Computes the Bessel i0 function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i0
* tf.compat.v2.math.bessel\_i0
* tf.math.bessel\_i0

tf.math.bessel\_i0(  
    x,  
    name=None  
)

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

Modified Bessel function of order 0.

It is preferable to use the numerically stabler function i0e(x) instead.

#### Args:

* **x**: A Tensor or SparseTensor. Must be one of the following types: half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor, respectively. Has the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.i0

# tf.math.bessel\_i0e

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i0e#aliases)

Computes the Bessel i0e function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i0e
* tf.compat.v2.math.bessel\_i0e
* tf.math.bessel\_i0e

tf.math.bessel\_i0e(  
    x,  
    name=None  
)

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

Exponentially scaled modified Bessel function of order 0 defined as bessel\_i0e(x) = exp(-abs(x)) bessel\_i0(x).

This function is faster and numerically stabler than bessel\_i0(x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.bessel\_i0e(x.values, ...), x.dense\_shape)

# tf.math.bessel\_i1

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1#aliases)

Computes the Bessel i1 function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i1
* tf.compat.v2.math.bessel\_i1
* tf.math.bessel\_i1

tf.math.bessel\_i1(  
    x,  
    name=None  
)

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

Modified Bessel function of order 1.

It is preferable to use the numerically stabler function i1e(x) instead.

#### Args:

* **x**: A Tensor or SparseTensor. Must be one of the following types: half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor or SparseTensor, respectively. Has the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.i1

# tf.math.bessel\_i1e

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/bessel_i1e#aliases)

Computes the Bessel i1e function of x element-wise.

### Aliases:

* tf.compat.v1.math.bessel\_i1e
* tf.compat.v2.math.bessel\_i1e
* tf.math.bessel\_i1e

tf.math.bessel\_i1e(  
    x,  
    name=None  
)

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

Exponentially scaled modified Bessel function of order 0 defined as bessel\_i1e(x) = exp(-abs(x)) bessel\_i1(x).

This function is faster and numerically stabler than bessel\_i1(x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.bessel\_i1e(x.values, ...), x.dense\_shape)

# tf.math.betainc

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

Compute the regularized incomplete beta integral Ix(a,b).

### Aliases:

* tf.compat.v1.betainc
* tf.compat.v1.math.betainc
* tf.compat.v2.math.betainc
* tf.math.betainc

tf.math.betainc(  
    a,  
    b,  
    x,  
    name=None  
)

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

The regularized incomplete beta integral is defined as:

Ix(a,b)=B(x;a,b)B(a,b)

where

B(x;a,b)=∫0xta−1(1−t)b−1dt

is the incomplete beta function and B(a,b) is the complete beta function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **b**: A Tensor. Must have the same type as a.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.bincount

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

Counts the number of occurrences of each value in an integer array.

### Aliases:

* tf.compat.v2.math.bincount
* tf.math.bincount

tf.math.bincount(  
    arr,  
    weights=None,  
    minlength=None,  
    maxlength=None,  
    dtype=tf.dtypes.int32,  
    name=None  
)

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

If minlength and maxlength are not given, returns a vector with length tf.reduce\_max(arr) + 1if arr is non-empty, and length 0 otherwise. If weights are non-None, then index i of the output stores the sum of the value in weights at each index where the corresponding value in arr is i.

#### Args:

* **arr**: An int32 tensor of non-negative values.
* **weights**: If non-None, must be the same shape as arr. For each value in arr, the bin will be incremented by the corresponding weight instead of 1.
* **minlength**: If given, ensures the output has length at least minlength, padding with zeros at the end if necessary.
* **maxlength**: If given, skips values in arr that are equal or greater than maxlength, ensuring that the output has length at most maxlength.
* **dtype**: If weights is None, determines the type of the output bins.
* **name**: A name scope for the associated operations (optional).

#### Returns:

A vector with the same dtype as weights or the given dtype. The bin values.

# tf.math.ceil

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

Returns element-wise smallest integer not less than x.

### Aliases:

* tf.compat.v1.ceil
* tf.compat.v1.math.ceil
* tf.compat.v2.math.ceil
* tf.math.ceil

tf.math.ceil(  
    x,  
    name=None  
)

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

### Used in the tutorials:

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.confusion\_matrix

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

Computes the confusion matrix from predictions and labels.

### Aliases:

* tf.compat.v2.math.confusion\_matrix
* tf.math.confusion\_matrix

tf.math.confusion\_matrix(  
    labels,  
    predictions,  
    num\_classes=None,  
    weights=None,  
    dtype=tf.dtypes.int32,  
    name=None  
)

Defined in [python/ops/confusion\_matrix.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/confusion_matrix.py).

The matrix columns represent the prediction labels and the rows represent the real labels. The confusion matrix is always a 2-D array of shape [n, n], where n is the number of valid labels for a given classification task. Both prediction and labels must be 1-D arrays of the same shape in order for this function to work.

If num\_classes is None, then num\_classes will be set to one plus the maximum value in either predictions or labels. Class labels are expected to start at 0. For example, if num\_classes is 3, then the possible labels would be [0, 1, 2].

If weights is not None, then each prediction contributes its corresponding weight to the total value of the confusion matrix cell.

#### For example:

  tf.math.confusion\_matrix([1, 2, 4], [2, 2, 4]) ==>  
      [[0 0 0 0 0]  
       [0 0 1 0 0]  
       [0 0 1 0 0]  
       [0 0 0 0 0]  
       [0 0 0 0 1]]

Note that the possible labels are assumed to be [0, 1, 2, 3, 4], resulting in a 5x5 confusion matrix.

#### Args:

* **labels**: 1-D Tensor of real labels for the classification task.
* **predictions**: 1-D Tensor of predictions for a given classification.
* **num\_classes**: The possible number of labels the classification task can have. If this value is not provided, it will be calculated using both predictions and labels array.
* **weights**: An optional Tensor whose shape matches predictions.
* **dtype**: Data type of the confusion matrix.
* **name**: Scope name.

#### Returns:

A Tensor of type dtype with shape [n, n] representing the confusion matrix, where n is the number of possible labels in the classification task.

#### Raises:

* **ValueError**: If both predictions and labels are not 1-D vectors and have mismatched shapes, or if weights is not None and its shape doesn't match predictions.

# tf.math.conj

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

Returns the complex conjugate of a complex number.

### Aliases:

* tf.compat.v1.conj
* tf.compat.v1.math.conj
* tf.compat.v2.math.conj
* tf.math.conj

tf.math.conj(  
    x,  
    name=None  
)

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

Given a tensor input of complex numbers, this operation returns a tensor of complex numbers that are the complex conjugate of each element in input. The complex numbers in input must be of the form a+bj, where a is the real part and b is the imaginary part.

The complex conjugate returned by this operation is of the form a−bj.

#### For example:

# tensor 'input' is [-2.25 + 4.75j, 3.25 + 5.75j]

tf.math.conj(input) ==> [-2.25 - 4.75j, 3.25 - 5.75j]

If x is real, it is returned unchanged.

#### Args:

* **x**: Tensor to conjugate. Must have numeric or variant type.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor that is the conjugate of x (with the same type).

#### Raises:

* **TypeError**: If x is not a numeric tensor.

# tf.math.cos

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

Computes cos of x element-wise.

### Aliases:

* tf.compat.v1.cos
* tf.compat.v1.math.cos
* tf.compat.v2.cos
* tf.compat.v2.math.cos
* tf.cos
* tf.math.cos

tf.math.cos(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.cosh

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

Computes hyperbolic cosine of x element-wise.

### Aliases:

* tf.compat.v1.cosh
* tf.compat.v1.math.cosh
* tf.compat.v2.cosh
* tf.compat.v2.math.cosh
* tf.cosh
* tf.math.cosh

tf.math.cosh(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.count\_nonzero

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

Computes number of nonzero elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.count\_nonzero
* tf.math.count\_nonzero

tf.math.count\_nonzero(  
    input,  
    axis=None,  
    keepdims=None,  
    dtype=tf.dtypes.int64,  
    name=None  
)

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

Reduces input along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned.

**NOTE** Floating point comparison to zero is done by exact floating point equality check. Small values are **not** rounded to zero for purposes of the nonzero check.

#### For example:

x = tf.constant([[0, 1, 0], [1, 1, 0]])  
tf.math.count\_nonzero(x)  # 3  
tf.math.count\_nonzero(x, 0)  # [1, 2, 0]  
tf.math.count\_nonzero(x, 1)  # [1, 2]  
tf.math.count\_nonzero(x, 1, keepdims=True)  # [[1], [2]]  
tf.math.count\_nonzero(x, [0, 1])  # 3

**NOTE** Strings are compared against zero-length empty string "". Any string with a size greater than zero is already considered as nonzero.

#### For example:

x = tf.constant(["", "a", "  ", "b", ""])  
tf.math.count\_nonzero(x) # 3, with "a", "  ", and "b" as nonzero strings.

#### Args:

* **input**: The tensor to reduce. Should be of numeric type, bool, or string.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input), rank(input)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **dtype**: The output dtype; defaults to [tf.int64](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#int64).
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor (number of nonzero values).

# tf.math.cumprod

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

Compute the cumulative product of the tensor x along axis.

### Aliases:

* tf.compat.v1.cumprod
* tf.compat.v1.math.cumprod
* tf.compat.v2.math.cumprod
* tf.math.cumprod

tf.math.cumprod(  
    x,  
    axis=0,  
    exclusive=False,  
    reverse=False,  
    name=None  
)

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

By default, this op performs an inclusive cumprod, which means that the first element of the input is identical to the first element of the output:

tf.math.cumprod([a, b, c])  # [a, a \* b, a \* b \* c]

By setting the exclusive kwarg to True, an exclusive cumprod is performed instead:

tf.math.cumprod([a, b, c], exclusive=True)  # [1, a, a \* b]

By setting the reverse kwarg to True, the cumprod is performed in the opposite direction:

tf.math.cumprod([a, b, c], reverse=True)  # [a \* b \* c, b \* c, c]

This is more efficient than using separate [tf.reverse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reverse) ops. The reverse and exclusive kwargs can also be combined:

tf.math.cumprod([a, b, c], exclusive=True, reverse=True)  # [b \* c, c, 1]

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int64, int32, uint8, uint16, int16, int8, complex64, complex128, qint8, quint8, qint32, half.
* **axis**: A Tensor of type int32 (default: 0). Must be in the range [-rank(x), rank(x)).
* **exclusive**: If True, perform exclusive cumprod.
* **reverse**: A bool (default: False).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.cumsum

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

Compute the cumulative sum of the tensor x along axis.

### Aliases:

* tf.compat.v1.cumsum
* tf.compat.v1.math.cumsum
* tf.compat.v2.cumsum
* tf.compat.v2.math.cumsum
* tf.cumsum
* tf.math.cumsum

tf.math.cumsum(  
    x,  
    axis=0,  
    exclusive=False,  
    reverse=False,  
    name=None  
)

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

By default, this op performs an inclusive cumsum, which means that the first element of the input is identical to the first element of the output:

tf.cumsum([a, b, c])  # [a, a + b, a + b + c]

By setting the exclusive kwarg to True, an exclusive cumsum is performed instead:

tf.cumsum([a, b, c], exclusive=True)  # [0, a, a + b]

By setting the reverse kwarg to True, the cumsum is performed in the opposite direction:

tf.cumsum([a, b, c], reverse=True)  # [a + b + c, b + c, c]

This is more efficient than using separate [tf.reverse](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/reverse) ops.

The reverse and exclusive kwargs can also be combined:

tf.cumsum([a, b, c], exclusive=True, reverse=True)  # [b + c, c, 0]

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int64, int32, uint8, uint16, int16, int8, complex64, complex128, qint8, quint8, qint32, half.
* **axis**: A Tensor of type int32 (default: 0). Must be in the range [-rank(x), rank(x)).
* **exclusive**: If True, perform exclusive cumsum.
* **reverse**: A bool (default: False).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.digamma

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

Computes Psi, the derivative of Lgamma (the log of the absolute value of

### Aliases:

* tf.compat.v1.digamma
* tf.compat.v1.math.digamma
* tf.compat.v2.math.digamma
* tf.math.digamma

tf.math.digamma(  
    x,  
    name=None  
)

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

Gamma(x)), element-wise.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.divide

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

Computes Python style division of x by y.

### Aliases:

* tf.compat.v1.divide
* tf.compat.v1.math.divide
* tf.compat.v2.divide
* tf.compat.v2.math.divide
* tf.divide
* tf.math.divide

tf.math.divide(  
    x,  
    y,  
    name=None  
)

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

# tf.math.divide\_no\_nan

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

Computes an unsafe divide which returns 0 if the y is zero.

### Aliases:

* tf.compat.v1.div\_no\_nan
* tf.compat.v1.math.divide\_no\_nan
* tf.compat.v2.math.divide\_no\_nan
* tf.math.divide\_no\_nan

tf.math.divide\_no\_nan(  
    x,  
    y,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **y**: A Tensor whose dtype is compatible with x.
* **name**: A name for the operation (optional).

#### Returns:

The element-wise value of the x divided by y.

# tf.math.equal

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

Returns the truth value of (x == y) element-wise.

### Aliases:

* tf.compat.v1.equal
* tf.compat.v1.math.equal
* tf.compat.v2.equal
* tf.compat.v2.math.equal
* tf.equal
* tf.math.equal

tf.math.equal(  
    x,  
    y,  
    name=None  
)

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

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Load CSV with tf.data](https://www.tensorflow.org/beta/tutorials/load_data/csv)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

NOTE: math.equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, quint8, qint8, qint32, string, bool, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.erf

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

Computes the Gauss error function of x element-wise.

### Aliases:

* tf.compat.v1.erf
* tf.compat.v1.math.erf
* tf.compat.v2.math.erf
* tf.math.erf

tf.math.erf(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.erf(x.values, ...), x.dense\_shape)

# tf.math.erfc

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

Computes the complementary error function of x element-wise.

### Aliases:

* tf.compat.v1.erfc
* tf.compat.v1.math.erfc
* tf.compat.v2.math.erfc
* tf.math.erfc

tf.math.erfc(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.exp

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

Computes exponential of x element-wise. y=ex.

### Aliases:

* tf.compat.v1.exp
* tf.compat.v1.math.exp
* tf.compat.v2.exp
* tf.compat.v2.math.exp
* tf.exp
* tf.math.exp

tf.math.exp(  
    x,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.expm1

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/expm1#aliases)

Computes exponential of x - 1 element-wise.

### Aliases:

* tf.compat.v1.expm1
* tf.compat.v1.math.expm1
* tf.compat.v2.math.expm1
* tf.math.expm1

tf.math.expm1(  
    x,  
    name=None  
)

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

I.e., y=(exp⁡x)−1.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.floor

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

Returns element-wise largest integer not greater than x.

### Aliases:

* tf.compat.v1.floor
* tf.compat.v1.math.floor
* tf.compat.v2.floor
* tf.compat.v2.math.floor
* tf.floor
* tf.math.floor

tf.math.floor(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.floordiv

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

Divides x / y elementwise, rounding toward the most negative integer.

### Aliases:

* tf.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v1.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v1.floordiv
* tf.compat.v1.math.floordiv
* tf.compat.v2.RaggedTensor.\_\_floordiv\_\_
* tf.compat.v2.math.floordiv
* tf.math.floordiv

tf.math.floordiv(  
    x,  
    y,  
    name=None  
)

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

The same as tf.compat.v1.div(x,y) for integers, but uses tf.floor(tf.compat.v1.div(x,y))for floating point arguments so that the result is always an integer (though possibly an integer represented as floating point). This op is generated by x // y floor division in Python 3 and in Python 2.7 with from \_\_future\_\_ import division.

x and y must have the same type, and the result will have the same type as well.

#### Args:

* **x**: Tensor numerator of real numeric type.
* **y**: Tensor denominator of real numeric type.
* **name**: A name for the operation (optional).

#### Returns:

x / y rounded down.

#### Raises:

* **TypeError**: If the inputs are complex.

# tf.math.floormod

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

Returns element-wise remainder of division. When x < 0 xor y < 0 is

### Aliases:

* tf.RaggedTensor.\_\_mod\_\_
* tf.compat.v1.RaggedTensor.\_\_mod\_\_
* tf.compat.v1.floormod
* tf.compat.v1.math.floormod
* tf.compat.v1.math.mod
* tf.compat.v1.mod
* tf.compat.v2.RaggedTensor.\_\_mod\_\_
* tf.compat.v2.math.floormod
* tf.compat.v2.math.mod
* tf.math.floormod
* tf.math.mod

tf.math.floormod(  
    x,  
    y,  
    name=None  
)

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

true, this follows Python semantics in that the result here is consistent with a flooring divide. E.g. floor(x / y) \* y + mod(x, y) = x.

NOTE: math.floormod supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: int32, int64, bfloat16, half, float32, float64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.greater

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

Returns the truth value of (x > y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_gt\_\_
* tf.Tensor.\_\_gt\_\_
* tf.compat.v1.RaggedTensor.\_\_gt\_\_
* tf.compat.v1.Tensor.\_\_gt\_\_
* tf.compat.v1.greater
* tf.compat.v1.math.greater
* tf.compat.v2.RaggedTensor.\_\_gt\_\_
* tf.compat.v2.Tensor.\_\_gt\_\_
* tf.compat.v2.greater
* tf.compat.v2.math.greater
* tf.greater
* tf.math.greater

tf.math.greater(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.greater supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.greater\_equal

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

Returns the truth value of (x >= y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_ge\_\_
* tf.Tensor.\_\_ge\_\_
* tf.compat.v1.RaggedTensor.\_\_ge\_\_
* tf.compat.v1.Tensor.\_\_ge\_\_
* tf.compat.v1.greater\_equal
* tf.compat.v1.math.greater\_equal
* tf.compat.v2.RaggedTensor.\_\_ge\_\_
* tf.compat.v2.Tensor.\_\_ge\_\_
* tf.compat.v2.greater\_equal
* tf.compat.v2.math.greater\_equal
* tf.greater\_equal
* tf.math.greater\_equal

tf.math.greater\_equal(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.greater\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.igamma

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

Compute the lower regularized incomplete Gamma function P(a, x).

### Aliases:

* tf.compat.v1.igamma
* tf.compat.v1.math.igamma
* tf.compat.v2.math.igamma
* tf.math.igamma

tf.math.igamma(  
    a,  
    x,  
    name=None  
)

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

The lower regularized incomplete Gamma function is defined as:

P(a,x)=gamma(a,x)/Gamma(a)=1−Q(a,x)

where

gamma(a,x)=int0xta−1exp(−t)dt

is the lower incomplete Gamma function.

Note, above Q(a, x) (Igammac) is the upper regularized complete Gamma function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.igammac

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

Compute the upper regularized incomplete Gamma function Q(a, x).

### Aliases:

* tf.compat.v1.igammac
* tf.compat.v1.math.igammac
* tf.compat.v2.math.igammac
* tf.math.igammac

tf.math.igammac(  
    a,  
    x,  
    name=None  
)

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

The upper regularized incomplete Gamma function is defined as:

Q(a,x)=Gamma(a,x)/Gamma(a)=1−P(a,x)

where

Gamma(a,x)=intx∞ta−1exp(−t)dt

is the upper incomplete Gama function.

Note, above P(a, x) (Igamma) is the lower regularized complete Gamma function.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.imag

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

Returns the imaginary part of a complex (or real) tensor.

### Aliases:

* tf.compat.v1.imag
* tf.compat.v1.math.imag
* tf.compat.v2.math.imag
* tf.math.imag

tf.math.imag(  
    input,  
    name=None  
)

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

Given a tensor input, this operation returns a tensor of type float that is the imaginary part of each element in input considered as a complex number. If input is real, a tensor of all zeros is returned.

#### For example:

x = tf.constant([-2.25 + 4.75j, 3.25 + 5.75j])  
tf.math.imag(x)  # [4.75, 5.75]

#### Args:

* **input**: A Tensor. Must be one of the following types: float, double, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32 or float64.

# tf.math.invert\_permutation

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

Computes the inverse permutation of a tensor.

### Aliases:

* tf.compat.v1.invert\_permutation
* tf.compat.v1.math.invert\_permutation
* tf.compat.v2.math.invert\_permutation
* tf.math.invert\_permutation

tf.math.invert\_permutation(  
    x,  
    name=None  
)

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

This operation computes the inverse of an index permutation. It takes a 1-D integer tensor x, which represents the indices of a zero-based array, and swaps each value with its index position. In other words, for an output tensor y and an input tensor x, this operation computes the following:

y[x[i]] = i for i in [0, 1, ..., len(x) - 1]

The values must include 0. There can be no duplicate values or negative values.

#### For example:

# tensor `x` is [3, 4, 0, 2, 1]  
invert\_permutation(x) ==> [2, 4, 3, 0, 1]

#### Args:

* **x**: A Tensor. Must be one of the following types: int32, int64. 1-D.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.in\_top\_k

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

Says whether the targets are in the top K predictions.

### Aliases:

* tf.compat.v2.math.in\_top\_k
* tf.compat.v2.nn.in\_top\_k
* tf.math.in\_top\_k
* tf.nn.in\_top\_k

tf.math.in\_top\_k(  
    targets,  
    predictions,  
    k,  
    name=None  
)

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

This outputs a batch\_size bool array, an entry out[i] is true if the prediction for the target class is finite (not inf, -inf, or nan) and among the top k predictions among all predictions for example i. Note that the behavior of InTopK differs from the TopK op in its handling of ties; if multiple classes have the same prediction value and straddle the top-k boundary, all of those classes are considered to be in the top k.

More formally, let

predictionsi be the predictions for all classes for example i, targetsi be the target class for example i, outi be the output for example i,

outi=predictionsi,targetsi∈TopKIncludingTies(predictionsi)

#### Args:

* **predictions**: A Tensor of type float32. A batch\_size x classes tensor.
* **targets**: A Tensor. Must be one of the following types: int32, int64. A batch\_size vector of class ids.
* **k**: An int. Number of top elements to look at for computing precision.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool. Computed Precision at k as a bool Tensor.

# tf.math.is\_finite

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

Returns which elements of x are finite.

### Aliases:

* tf.compat.v1.debugging.is\_finite
* tf.compat.v1.is\_finite
* tf.compat.v1.math.is\_finite
* tf.compat.v2.math.is\_finite
* tf.math.is\_finite

tf.math.is\_finite(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isfinite

# tf.math.is\_inf

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

Returns which elements of x are Inf.

### Aliases:

* tf.compat.v1.debugging.is\_inf
* tf.compat.v1.is\_inf
* tf.compat.v1.math.is\_inf
* tf.compat.v2.math.is\_inf
* tf.math.is\_inf

tf.math.is\_inf(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isinf

# tf.math.is\_nan

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

Returns which elements of x are NaN.

### Aliases:

* tf.compat.v1.debugging.is\_nan
* tf.compat.v1.is\_nan
* tf.compat.v1.math.is\_nan
* tf.compat.v2.math.is\_nan
* tf.math.is\_nan

tf.math.is\_nan(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

#### Numpy Compatibility

Equivalent to np.isnan

# tf.math.is\_non\_decreasing

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

Returns True if x is non-decreasing.

### Aliases:

* tf.compat.v1.debugging.is\_non\_decreasing
* tf.compat.v1.is\_non\_decreasing
* tf.compat.v1.math.is\_non\_decreasing
* tf.compat.v2.math.is\_non\_decreasing
* tf.math.is\_non\_decreasing

tf.math.is\_non\_decreasing(  
    x,  
    name=None  
)

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

Elements of x are compared in row-major order. The tensor [x[0],...] is non-decreasing if for every adjacent pair we have x[i] <= x[i+1]. If x has less than two elements, it is trivially non-decreasing.

See also: is\_strictly\_increasing

#### Args:

* **x**: Numeric Tensor.
* **name**: A name for this operation (optional). Defaults to "is\_non\_decreasing"

#### Returns:

Boolean Tensor, equal to True iff x is non-decreasing.

#### Raises:

* **TypeError**: if x is not a numeric tensor.

# tf.math.is\_strictly\_increasing

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

Returns True if x is strictly increasing.

### Aliases:

* tf.compat.v1.debugging.is\_strictly\_increasing
* tf.compat.v1.is\_strictly\_increasing
* tf.compat.v1.math.is\_strictly\_increasing
* tf.compat.v2.math.is\_strictly\_increasing
* tf.math.is\_strictly\_increasing

tf.math.is\_strictly\_increasing(  
    x,  
    name=None  
)

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

Elements of x are compared in row-major order. The tensor [x[0],...] is strictly increasing if for every adjacent pair we have x[i] < x[i+1]. If x has less than two elements, it is trivially strictly increasing.

See also: is\_non\_decreasing

#### Args:

* **x**: Numeric Tensor.
* **name**: A name for this operation (optional). Defaults to "is\_strictly\_increasing"

#### Returns:

Boolean Tensor, equal to True iff x is strictly increasing.

#### Raises:

* **TypeError**: if x is not a numeric tensor.

# tf.math.l2\_normalize

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/l2_normalize#aliases)

Normalizes along dimension axis using an L2 norm.

### Aliases:

* tf.compat.v2.linalg.l2\_normalize
* tf.compat.v2.math.l2\_normalize
* tf.compat.v2.nn.l2\_normalize
* tf.linalg.l2\_normalize
* tf.math.l2\_normalize
* tf.nn.l2\_normalize

tf.math.l2\_normalize(  
    x,  
    axis=None,  
    epsilon=1e-12,  
    name=None  
)

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

For a 1-D tensor with axis = 0, computes

output = x / sqrt(max(sum(x\*\*2), epsilon))

For x with more dimensions, independently normalizes each 1-D slice along dimension axis.

#### Args:

* **x**: A Tensor.
* **axis**: Dimension along which to normalize. A scalar or a vector of integers.
* **epsilon**: A lower bound value for the norm. Will use sqrt(epsilon) as the divisor if norm < sqrt(epsilon).
* **name**: A name for this operation (optional).

#### Returns:

A Tensor with the same shape as x.

# tf.math.lbeta

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

Computes ln(|Beta(x)|), reducing along the last dimension.

### Aliases:

* tf.compat.v1.lbeta
* tf.compat.v1.math.lbeta
* tf.compat.v2.math.lbeta
* tf.math.lbeta

tf.math.lbeta(  
    x,  
    name=None  
)

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

Given one-dimensional z = [z\_0,...,z\_{K-1}], we define

Beta(z)=∏jGamma(zj)/Gamma(∑jzj)

And for n + 1 dimensional x with shape [N1, ..., Nn, K], we define

lbeta(x)[i1,...,in]=Log(|Beta(x[i1,...,in,:])|).

In other words, the last dimension is treated as the z vector.

Note that if z = [u, v], then Beta(z)=int01tu−1(1−t)v−1dt, which defines the traditional bivariate beta function.

If the last dimension is empty, we follow the convention that the sum over the empty set is zero, and the product is one.

#### Args:

* **x**: A rank n + 1 Tensor, n >= 0 with type float, or double.
* **name**: A name for the operation (optional).

#### Returns:

The logarithm of |Beta(x)| reducing along the last dimension.

# tf.math.less

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

Returns the truth value of (x < y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_lt\_\_
* tf.Tensor.\_\_lt\_\_
* tf.compat.v1.RaggedTensor.\_\_lt\_\_
* tf.compat.v1.Tensor.\_\_lt\_\_
* tf.compat.v1.less
* tf.compat.v1.math.less
* tf.compat.v2.RaggedTensor.\_\_lt\_\_
* tf.compat.v2.Tensor.\_\_lt\_\_
* tf.compat.v2.less
* tf.compat.v2.math.less
* tf.less
* tf.math.less

tf.math.less(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.less supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.less\_equal

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

Returns the truth value of (x <= y) element-wise.

### Aliases:

* tf.RaggedTensor.\_\_le\_\_
* tf.Tensor.\_\_le\_\_
* tf.compat.v1.RaggedTensor.\_\_le\_\_
* tf.compat.v1.Tensor.\_\_le\_\_
* tf.compat.v1.less\_equal
* tf.compat.v1.math.less\_equal
* tf.compat.v2.RaggedTensor.\_\_le\_\_
* tf.compat.v2.Tensor.\_\_le\_\_
* tf.compat.v2.less\_equal
* tf.compat.v2.math.less\_equal
* tf.less\_equal
* tf.math.less\_equal

tf.math.less\_equal(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.less\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.lgamma

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

Computes the log of the absolute value of Gamma(x) element-wise.

### Aliases:

* tf.compat.v1.lgamma
* tf.compat.v1.math.lgamma
* tf.compat.v2.math.lgamma
* tf.math.lgamma

tf.math.lgamma(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.log

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

Computes natural logarithm of x element-wise.

### Aliases:

* tf.compat.v1.log
* tf.compat.v1.math.log
* tf.compat.v2.math.log
* tf.math.log

tf.math.log(  
    x,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

I.e., y=loge⁡x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.log1p

* [Contents](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/log1p#aliases)

Computes natural logarithm of (1 + x) element-wise.

### Aliases:

* tf.compat.v1.log1p
* tf.compat.v1.math.log1p
* tf.compat.v2.math.log1p
* tf.math.log1p

tf.math.log1p(  
    x,  
    name=None  
)

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

I.e., y=loge⁡(1+x).

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.logical\_and

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

Returns the truth value of x AND y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_and\_\_
* tf.compat.v1.RaggedTensor.\_\_and\_\_
* tf.compat.v1.logical\_and
* tf.compat.v1.math.logical\_and
* tf.compat.v2.RaggedTensor.\_\_and\_\_
* tf.compat.v2.logical\_and
* tf.compat.v2.math.logical\_and
* tf.logical\_and
* tf.math.logical\_and

tf.math.logical\_and(  
    x,  
    y,  
    name=None  
)

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

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.logical\_and supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor of type bool.
* **y**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_not

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

Returns the truth value of NOT x element-wise.

### Aliases:

* tf.RaggedTensor.\_\_invert\_\_
* tf.Tensor.\_\_invert\_\_
* tf.compat.v1.RaggedTensor.\_\_invert\_\_
* tf.compat.v1.Tensor.\_\_invert\_\_
* tf.compat.v1.logical\_not
* tf.compat.v1.math.logical\_not
* tf.compat.v2.RaggedTensor.\_\_invert\_\_
* tf.compat.v2.Tensor.\_\_invert\_\_
* tf.compat.v2.logical\_not
* tf.compat.v2.math.logical\_not
* tf.logical\_not
* tf.math.logical\_not

tf.math.logical\_not(  
    x,  
    name=None  
)

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

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

#### Args:

* **x**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_or

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

Returns the truth value of x OR y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_or\_\_
* tf.compat.v1.RaggedTensor.\_\_or\_\_
* tf.compat.v1.logical\_or
* tf.compat.v1.math.logical\_or
* tf.compat.v2.RaggedTensor.\_\_or\_\_
* tf.compat.v2.logical\_or
* tf.compat.v2.math.logical\_or
* tf.logical\_or
* tf.math.logical\_or

tf.math.logical\_or(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.logical\_or supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor of type bool.
* **y**: A Tensor of type bool.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.logical\_xor

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

Logical XOR function.

### Aliases:

* tf.RaggedTensor.\_\_xor\_\_
* tf.compat.v1.RaggedTensor.\_\_xor\_\_
* tf.compat.v1.logical\_xor
* tf.compat.v1.math.logical\_xor
* tf.compat.v2.RaggedTensor.\_\_xor\_\_
* tf.compat.v2.math.logical\_xor
* tf.math.logical\_xor

tf.math.logical\_xor(  
    x,  
    y,  
    name='LogicalXor'  
)

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

x ^ y = (x | y) & ~(x & y)

Inputs are tensor and if the tensors contains more than one element, an element-wise logical XOR is computed.

#### Usage:

x = tf.constant([False, False, True, True], dtype = tf.bool)  
y = tf.constant([False, True, False, True], dtype = tf.bool)  
z = tf.logical\_xor(x, y, name="LogicalXor")  
#  here z = [False  True  True False]

#### Args:

* **x**: A Tensor type bool.
* **y**: A Tensor of type bool.

#### Returns:

A Tensor of type bool with the same size as that of x or y.

# tf.math.log\_sigmoid

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

Computes log sigmoid of x element-wise.

### Aliases:

* tf.compat.v1.log\_sigmoid
* tf.compat.v1.math.log\_sigmoid
* tf.compat.v2.math.log\_sigmoid
* tf.math.log\_sigmoid

tf.math.log\_sigmoid(  
    x,  
    name=None  
)

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

Specifically, y = log(1 / (1 + exp(-x))). For numerical stability, we use y = -tf.nn.softplus(-x).

#### Args:

* **x**: A Tensor with type float32 or float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as x.

# tf.math.maximum

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

Returns the max of x and y (i.e. x > y ? x : y) element-wise.

### Aliases:

* tf.compat.v1.math.maximum
* tf.compat.v1.maximum
* tf.compat.v2.math.maximum
* tf.compat.v2.maximum
* tf.math.maximum
* tf.maximum

tf.math.maximum(  
    x,  
    y,  
    name=None  
)

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

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.maximum supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.minimum

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

Returns the min of x and y (i.e. x < y ? x : y) element-wise.

### Aliases:

* tf.compat.v1.math.minimum
* tf.compat.v1.minimum
* tf.compat.v2.math.minimum
* tf.compat.v2.minimum
* tf.math.minimum
* tf.minimum

tf.math.minimum(  
    x,  
    y,  
    name=None  
)

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

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

NOTE: math.minimum supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.multiply

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

Returns x \* y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_mul\_\_
* tf.compat.v1.RaggedTensor.\_\_mul\_\_
* tf.compat.v1.math.multiply
* tf.compat.v1.multiply
* tf.compat.v2.RaggedTensor.\_\_mul\_\_
* tf.compat.v2.math.multiply
* tf.compat.v2.multiply
* tf.math.multiply
* tf.multiply

tf.math.multiply(  
    x,  
    y,  
    name=None  
)

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

### Used in the guide:

* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)

### Used in the tutorials:

* [Automatic differentiation and gradient tape](https://www.tensorflow.org/beta/tutorials/eager/automatic_differentiation)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

NOTE: <a href="../../tf/math/multiply"><code>tf.multiply</code></a> supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, uint16, int16, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.multiply\_no\_nan

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

Computes the product of x and y and returns 0 if the y is zero, even if x is NaN or infinite.

### Aliases:

* tf.compat.v1.math.multiply\_no\_nan
* tf.compat.v2.math.multiply\_no\_nan
* tf.math.multiply\_no\_nan

tf.math.multiply\_no\_nan(  
    x,  
    y,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **y**: A Tensor whose dtype is compatible with x.
* **name**: A name for the operation (optional).

#### Returns:

The element-wise value of the x times y.

# tf.math.negative

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

Computes numerical negative value element-wise.

### Aliases:

* tf.RaggedTensor.\_\_neg\_\_
* tf.Tensor.\_\_neg\_\_
* tf.compat.v1.RaggedTensor.\_\_neg\_\_
* tf.compat.v1.Tensor.\_\_neg\_\_
* tf.compat.v1.math.negative
* tf.compat.v1.negative
* tf.compat.v2.RaggedTensor.\_\_neg\_\_
* tf.compat.v2.Tensor.\_\_neg\_\_
* tf.compat.v2.math.negative
* tf.compat.v2.negative
* tf.math.negative
* tf.negative

tf.math.negative(  
    x,  
    name=None  
)

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

I.e., y=−x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.negative(x.values, ...), x.dense\_shape)

# tf.math.nextafter

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

Returns the next representable value of x1 in the direction of x2, element-wise.

### Aliases:

* tf.compat.v1.math.nextafter
* tf.compat.v2.math.nextafter
* tf.math.nextafter

tf.math.nextafter(  
    x1,  
    x2,  
    name=None  
)

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

This operation returns the same result as the C++ std::nextafter function.

It can also return a subnormal number.

#### Args:

* **x1**: A Tensor. Must be one of the following types: float64, float32.
* **x2**: A Tensor. Must have the same type as x1.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x1.

#### Cpp Compatibility

Equivalent to C++ std::nextafter function.

# tf.math.not\_equal

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

Returns the truth value of (x != y) element-wise.

### Aliases:

* tf.compat.v1.math.not\_equal
* tf.compat.v1.not\_equal
* tf.compat.v2.math.not\_equal
* tf.compat.v2.not\_equal
* tf.math.not\_equal
* tf.not\_equal

tf.math.not\_equal(  
    x,  
    y,  
    name=None  
)

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

### Used in the tutorials:

* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)

NOTE: math.not\_equal supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, int16, int32, int64, complex64, quint8, qint8, qint32, string, bool, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type bool.

# tf.math.polygamma

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

Compute the polygamma function ψ(n)(x).

### Aliases:

* tf.compat.v1.math.polygamma
* tf.compat.v1.polygamma
* tf.compat.v2.math.polygamma
* tf.math.polygamma

tf.math.polygamma(  
    a,  
    x,  
    name=None  
)

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

The polygamma function is defined as:

ψ(a)(x)=dadxaψ(x)

where ψ(x) is the digamma function. The polygamma function is defined only for non-negative integer orders \a\.

#### Args:

* **a**: A Tensor. Must be one of the following types: float32, float64.
* **x**: A Tensor. Must have the same type as a.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as a.

# tf.math.polyval

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

Computes the elementwise value of a polynomial.

### Aliases:

* tf.compat.v1.math.polyval
* tf.compat.v2.math.polyval
* tf.math.polyval

tf.math.polyval(  
    coeffs,  
    x,  
    name=None  
)

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

If x is a tensor and coeffs is a list n + 1 tensors, this function returns the value of the n-th order polynomial

p(x) = coeffs[n-1] + coeffs[n-2] \* x + ... + coeffs[0] \* x\*\*(n-1)

evaluated using Horner's method, i.e.

p(x) = coeffs[n-1] + x \* (coeffs[n-2] + ... + x \* (coeffs[1] + x \* coeffs[0]))

#### Args:

* **coeffs**: A list of Tensor representing the coefficients of the polynomial.
* **x**: A Tensor representing the variable of the polynomial.
* **name**: A name for the operation (optional).

#### Returns:

A tensor of the shape as the expression p(x) with usual broadcasting rules for element-wise addition and multiplication applied.

#### Numpy Compatibility

Equivalent to numpy.polyval.

# tf.math.pow

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

Computes the power of one value to another.

### Aliases:

* tf.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.math.pow
* tf.compat.v1.pow
* tf.compat.v2.RaggedTensor.\_\_pow\_\_
* tf.compat.v2.math.pow
* tf.compat.v2.pow
* tf.math.pow
* tf.pow

tf.math.pow(  
    x,  
    y,  
    name=None  
)

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

Given a tensor x and a tensor y, this operation computes xy for corresponding elements in x and y. For example:

x = tf.constant([[2, 2], [3, 3]])  
y = tf.constant([[8, 16], [2, 3]])  
tf.pow(x, y)  # [[256, 65536], [9, 27]]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **y**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor.

# tf.math.pow

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

Computes the power of one value to another.

### Aliases:

* tf.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.RaggedTensor.\_\_pow\_\_
* tf.compat.v1.math.pow
* tf.compat.v1.pow
* tf.compat.v2.RaggedTensor.\_\_pow\_\_
* tf.compat.v2.math.pow
* tf.compat.v2.pow
* tf.math.pow
* tf.pow

tf.math.pow(  
    x,  
    y,  
    name=None  
)

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

Given a tensor x and a tensor y, this operation computes xy for corresponding elements in x and y. For example:

x = tf.constant([[2, 2], [3, 3]])  
y = tf.constant([[8, 16], [2, 3]])  
tf.pow(x, y)  # [[256, 65536], [9, 27]]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **y**: A Tensor of type float16, float32, float64, int32, int64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor.

# tf.math.reciprocal

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

Computes the reciprocal of x element-wise.

### Aliases:

* tf.compat.v1.math.reciprocal
* tf.compat.v1.reciprocal
* tf.compat.v2.math.reciprocal
* tf.math.reciprocal

tf.math.reciprocal(  
    x,  
    name=None  
)

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

I.e., y=1/x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.reduce\_any

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

Computes the "logical or" of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_any
* tf.compat.v2.reduce\_any
* tf.math.reduce\_any
* tf.reduce\_any

tf.math.reduce\_any(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[True,  True], [False, False]])  
tf.reduce\_any(x)  # True  
tf.reduce\_any(x, 0)  # [True, True]  
tf.reduce\_any(x, 1)  # [True, False]

#### Args:

* **input\_tensor**: The boolean tensor to reduce.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.any

# tf.math.reduce\_euclidean\_norm

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

Computes the Euclidean norm of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_euclidean\_norm
* tf.compat.v2.math.reduce\_euclidean\_norm
* tf.math.reduce\_euclidean\_norm

tf.math.reduce\_euclidean\_norm(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1, 2, 3], [1, 1, 1]])  
tf.reduce\_euclidean\_norm(x)  # sqrt(17)  
tf.reduce\_euclidean\_norm(x, 0)  # [sqrt(2), sqrt(5), sqrt(10)]  
tf.reduce\_euclidean\_norm(x, 1)  # [sqrt(14), sqrt(3)]  
tf.reduce\_euclidean\_norm(x, 1, keepdims=True)  # [[sqrt(14)], [sqrt(3)]]  
tf.reduce\_euclidean\_norm(x, [0, 1])  # sqrt(17)

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

# tf.math.reduce\_logsumexp

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

Computes log(sum(exp(elements across dimensions of a tensor))).

### Aliases:

* tf.compat.v2.math.reduce\_logsumexp
* tf.compat.v2.reduce\_logsumexp
* tf.math.reduce\_logsumexp
* tf.reduce\_logsumexp

tf.math.reduce\_logsumexp(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis has no entries, all dimensions are reduced, and a tensor with a single element is returned.

This function is more numerically stable than log(sum(exp(input))). It avoids overflows caused by taking the exp of large inputs and underflows caused by taking the log of small inputs.

#### For example:

x = tf.constant([[0., 0., 0.], [0., 0., 0.]])  
tf.reduce\_logsumexp(x)  # log(6)  
tf.reduce\_logsumexp(x, 0)  # [log(2), log(2), log(2)]  
tf.reduce\_logsumexp(x, 1)  # [log(3), log(3)]  
tf.reduce\_logsumexp(x, 1, keepdims=True)  # [[log(3)], [log(3)]]  
tf.reduce\_logsumexp(x, [0, 1])  # log(6)

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

# tf.math.reduce\_max

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

Computes the maximum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_max
* tf.compat.v2.reduce\_max
* tf.math.reduce\_max
* tf.reduce\_max

tf.math.reduce\_max(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have real numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.max

# tf.math.reduce\_mean

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

Computes the mean of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_mean
* tf.compat.v2.reduce\_mean
* tf.math.reduce\_mean
* tf.reduce\_mean

tf.math.reduce\_mean(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Ragged Tensors](https://www.tensorflow.org/beta/guide/ragged_tensors)
* [Training checkpoints](https://www.tensorflow.org/beta/guide/checkpoints)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)
* [tf.function and AutoGraph in TensorFlow 2.0](https://www.tensorflow.org/beta/guide/autograph)

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Custom training: basics](https://www.tensorflow.org/beta/tutorials/eager/custom_training)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)
* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)
* [Text generation with an RNN](https://www.tensorflow.org/beta/tutorials/text/text_generation)
* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 1.], [2., 2.]])  
tf.reduce\_mean(x)  # 1.5  
tf.reduce\_mean(x, 0)  # [1.5, 1.5]  
tf.reduce\_mean(x, 1)  # [1.,  2.]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.mean

Please note that np.mean has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, [tf.reduce\_mean](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/reduce_mean) has an aggressive type inference from input\_tensor, for example:

x = tf.constant([1, 0, 1, 0])  
tf.reduce\_mean(x)  # 0  
y = tf.constant([1., 0., 1., 0.])  
tf.reduce\_mean(y)  # 0.5

# tf.math.reduce\_min

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

Computes the minimum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_min
* tf.compat.v2.reduce\_min
* tf.math.reduce\_min
* tf.reduce\_min

tf.math.reduce\_min(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have real numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.min

# tf.math.reduce\_prod

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

Computes the product of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_prod
* tf.compat.v2.reduce\_prod
* tf.math.reduce\_prod
* tf.reduce\_prod

tf.math.reduce\_prod(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor.

#### Numpy Compatibility

Equivalent to np.prod

# tf.math.reduce\_std

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

Computes the standard deviation of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_std
* tf.compat.v2.math.reduce\_std
* tf.math.reduce\_std

tf.math.reduce\_std(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 2.], [3., 4.]])  
tf.reduce\_std(x)  # 1.1180339887498949  
tf.reduce\_std(x, 0)  # [1., 1.]  
tf.reduce\_std(x, 1)  # [0.5,  0.5]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name scope for the associated operations (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.std

Please note that np.std has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, tf.reduce\_std has an aggressive type inference from input\_tensor,

# tf.math.reduce\_sum

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

Computes the sum of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v2.math.reduce\_sum
* tf.compat.v2.reduce\_sum
* tf.math.reduce\_sum
* tf.reduce\_sum

tf.math.reduce\_sum(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

### Used in the guide:

* [Distributed training in TensorFlow](https://www.tensorflow.org/beta/guide/distribute_strategy)
* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Training and Evaluation with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/training_and_evaluation)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Automatic differentiation and gradient tape](https://www.tensorflow.org/beta/tutorials/eager/automatic_differentiation)
* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)
* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Multi-worker Training with Estimator](https://www.tensorflow.org/beta/tutorials/distribute/multi_worker_with_estimator)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)
* [Unicode strings](https://www.tensorflow.org/beta/tutorials/text/unicode)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1, 1, 1], [1, 1, 1]])  
tf.reduce\_sum(x)  # 6  
tf.reduce\_sum(x, 0)  # [2, 2, 2]  
tf.reduce\_sum(x, 1)  # [3, 3]  
tf.reduce\_sum(x, 1, keepdims=True)  # [[3], [3]]  
tf.reduce\_sum(x, [0, 1])  # 6

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name for the operation (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.sum apart the fact that numpy upcast uint8 and int32 to int64 while tensorflow returns the same dtype as the input.

# tf.math.reduce\_variance

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

Computes the variance of elements across dimensions of a tensor.

### Aliases:

* tf.compat.v1.math.reduce\_variance
* tf.compat.v2.math.reduce\_variance
* tf.math.reduce\_variance

tf.math.reduce\_variance(  
    input\_tensor,  
    axis=None,  
    keepdims=False,  
    name=None  
)

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

Reduces input\_tensor along the dimensions given in axis. Unless keepdims is true, the rank of the tensor is reduced by 1 for each entry in axis. If keepdims is true, the reduced dimensions are retained with length 1.

If axis is None, all dimensions are reduced, and a tensor with a single element is returned.

#### For example:

x = tf.constant([[1., 2.], [3., 4.]])  
tf.reduce\_variance(x)  # 1.25  
tf.reduce\_variance(x, 0)  # [1., 1.]  
tf.reduce\_variance(x, 1)  # [0.25,  0.25]

#### Args:

* **input\_tensor**: The tensor to reduce. Should have numeric type.
* **axis**: The dimensions to reduce. If None (the default), reduces all dimensions. Must be in the range [-rank(input\_tensor), rank(input\_tensor)).
* **keepdims**: If true, retains reduced dimensions with length 1.
* **name**: A name scope for the associated operations (optional).

#### Returns:

The reduced tensor, of the same dtype as the input\_tensor.

#### Numpy Compatibility

Equivalent to np.var

Please note that np.var has a dtype parameter that could be used to specify the output type. By default this is dtype=float64. On the other hand, tf.reduce\_variance has an aggressive type inference from input\_tensor,

# tf.math.rint

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

Returns element-wise integer closest to x.

### Aliases:

* tf.compat.v1.math.rint
* tf.compat.v1.rint
* tf.compat.v2.math.rint
* tf.math.rint

tf.math.rint(  
    x,  
    name=None  
)

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

If the result is midway between two representable values, the even representable is chosen. For example:

rint(-1.5) ==> -2.0  
rint(0.5000001) ==> 1.0  
rint([-1.7, -1.5, -0.2, 0.2, 1.5, 1.7, 2.0]) ==> [-2., -2., -0., 0., 2., 2., 2.]

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.round

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

Rounds the values of a tensor to the nearest integer, element-wise.

### Aliases:

* tf.compat.v1.math.round
* tf.compat.v1.round
* tf.compat.v2.math.round
* tf.compat.v2.round
* tf.math.round
* tf.round

tf.math.round(  
    x,  
    name=None  
)

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

Rounds half to even. Also known as bankers rounding. If you want to round according to the current system rounding mode use tf::cint. For example:

x = tf.constant([0.9, 2.5, 2.3, 1.5, -4.5])  
tf.round(x)  # [ 1.0, 2.0, 2.0, 2.0, -4.0 ]

#### Args:

* **x**: A Tensor of type float16, float32, float64, int32, or int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of same shape and type as x.

# tf.math.rsqrt

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

Computes reciprocal of square root of x element-wise.

### Aliases:

* tf.compat.v1.math.rsqrt
* tf.compat.v1.rsqrt
* tf.compat.v2.math.rsqrt
* tf.math.rsqrt

tf.math.rsqrt(  
    x,  
    name=None  
)

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

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

I.e., y=1/x.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.scalar\_mul

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

Multiplies a scalar times a Tensor or IndexedSlices object.

### Aliases:

* tf.compat.v2.math.scalar\_mul
* tf.compat.v2.scalar\_mul
* tf.math.scalar\_mul
* tf.scalar\_mul

tf.math.scalar\_mul(  
    scalar,  
    x,  
    name=None  
)

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

Intended for use in gradient code which might deal with IndexedSlices objects, which are easy to multiply by a scalar but more expensive to multiply with arbitrary tensors.

#### Args:

* **scalar**: A 0-D scalar Tensor. Must have known shape.
* **x**: A Tensor or IndexedSlices to be scaled.
* **name**: A name for the operation (optional).

#### Returns:

scalar \* x of the same type (Tensor or IndexedSlices) as x.

#### Raises:

* **ValueError**: if scalar is not a 0-D scalar.

# tf.math.segment\_max

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

Computes the maximum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_max
* tf.compat.v1.segment\_max
* tf.compat.v2.math.segment\_max
* tf.math.segment\_max

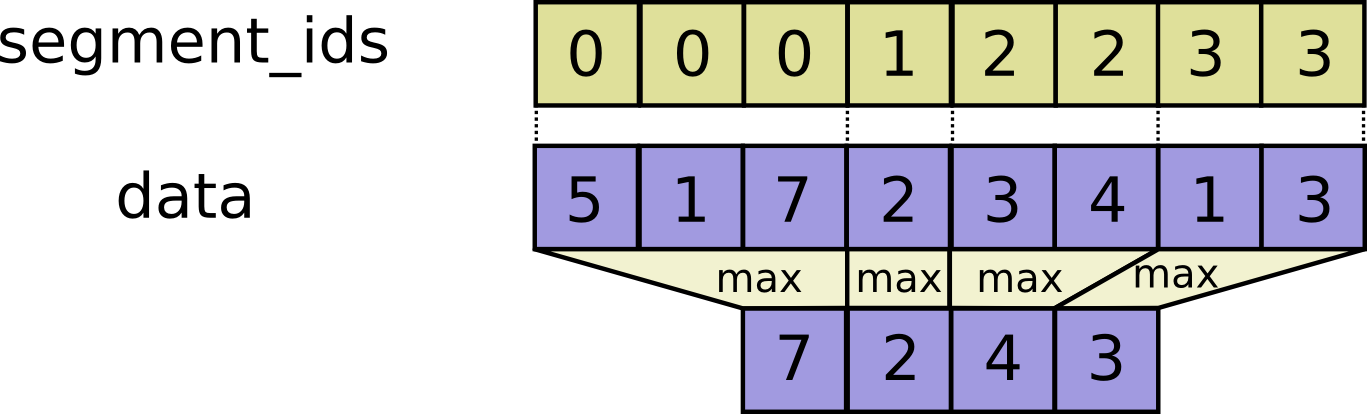
tf.math.segment\_max(  
    data,  
    segment\_ids,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=maxj(dataj) where max is over j such that segment\_ids[j] == i.

If the max is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_max(c, tf.constant([0, 0, 1]))  
# ==> [[4, 3, 3, 4],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_mean

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

Computes the mean along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_mean
* tf.compat.v1.segment\_mean
* tf.compat.v2.math.segment\_mean
* tf.math.segment\_mean

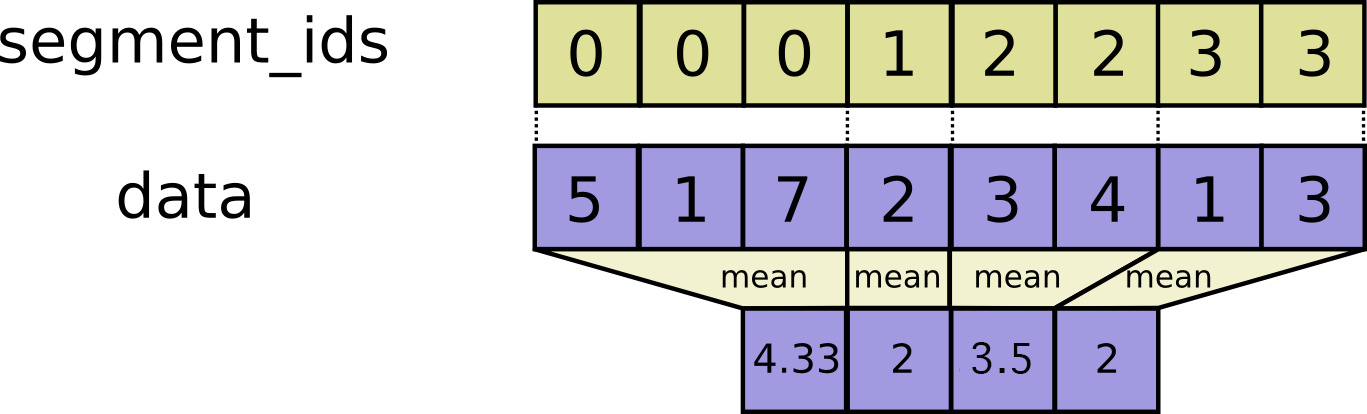
tf.math.segment\_mean(  
    data,  
    segment\_ids,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∑jdatajN where mean is over j such that segment\_ids[j] == i and N is the total number of values summed.

If the mean is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1.0,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_mean(c, tf.constant([0, 0, 1]))  
# ==> [[2.5, 2.5, 2.5, 2.5],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_min

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

Computes the minimum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_min
* tf.compat.v1.segment\_min
* tf.compat.v2.math.segment\_min
* tf.math.segment\_min

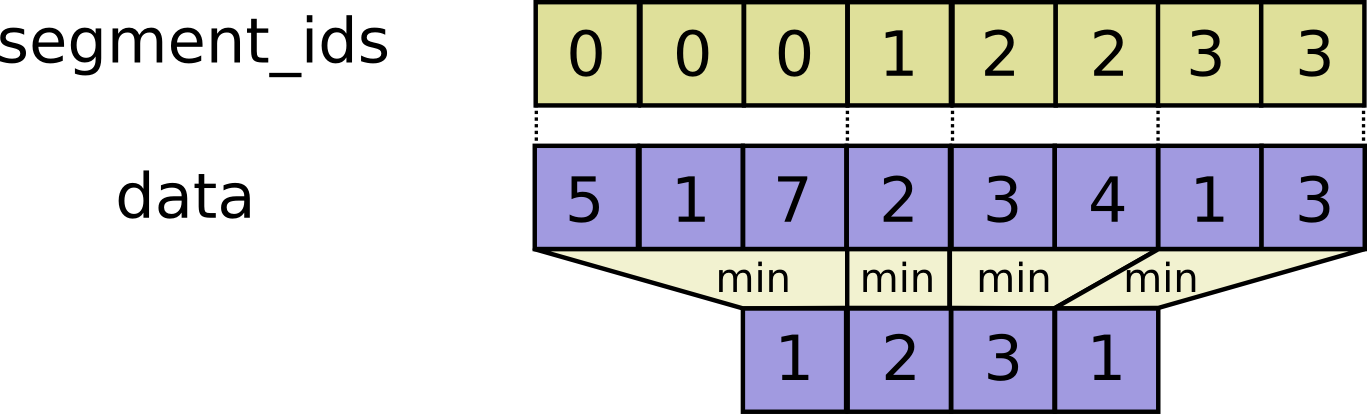
tf.math.segment\_min(  
    data,  
    segment\_ids,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=minj(dataj) where min is over j such that segment\_ids[j] == i.

If the min is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_min(c, tf.constant([0, 0, 1]))  
# ==> [[1, 2, 2, 1],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_prod

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

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_prod
* tf.compat.v1.segment\_prod
* tf.compat.v2.math.segment\_prod
* tf.math.segment\_prod

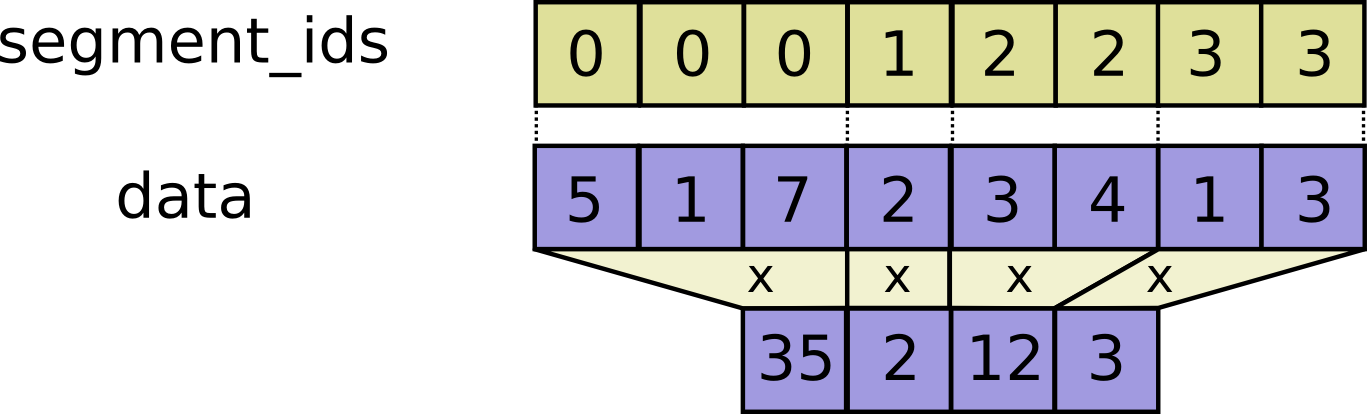
tf.math.segment\_prod(  
    data,  
    segment\_ids,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∏jdataj where the product is over j such that segment\_ids[j] == i.

If the product is empty for a given segment ID i, output[i] = 1.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_prod(c, tf.constant([0, 0, 1]))  
# ==> [[4, 6, 6, 4],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.segment\_sum

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

Computes the sum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.segment\_sum
* tf.compat.v1.segment\_sum
* tf.compat.v2.math.segment\_sum
* tf.math.segment\_sum

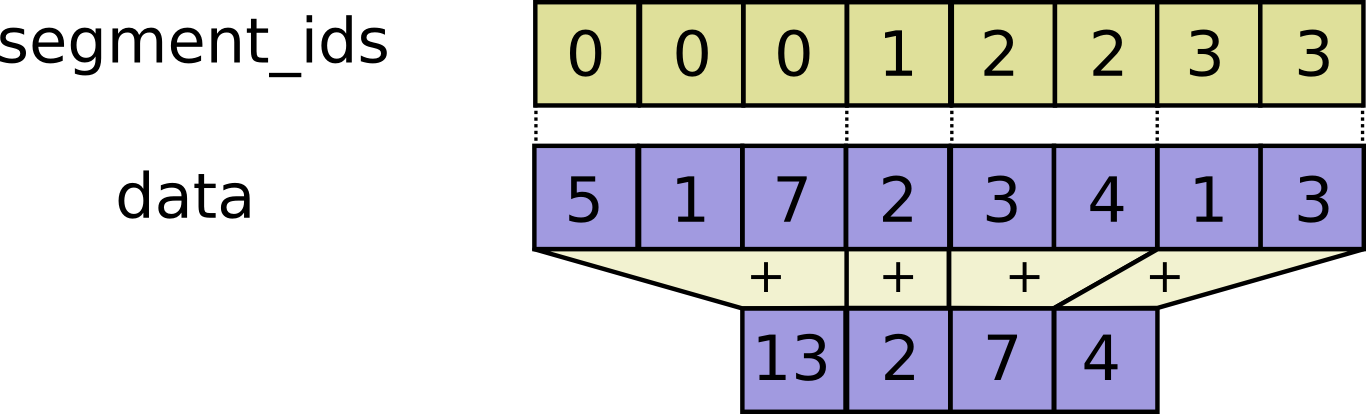
tf.math.segment\_sum(  
    data,  
    segment\_ids,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that outputi=∑jdataj where sum is over j such that segment\_ids[j] == i.

If the sum is empty for a given segment ID i, output[i] = 0.



#### For example:

c = tf.constant([[1,2,3,4], [4, 3, 2, 1], [5,6,7,8]])  
tf.segment\_sum(c, tf.constant([0, 0, 1]))  
# ==> [[5, 5, 5, 5],  
#      [5, 6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A 1-D tensor whose size is equal to the size of data's first dimension. Values should be sorted and can be repeated.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.sigmoid

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

Computes sigmoid of x element-wise.

### Aliases:

* tf.compat.v1.math.sigmoid
* tf.compat.v1.nn.sigmoid
* tf.compat.v1.sigmoid
* tf.compat.v2.math.sigmoid
* tf.compat.v2.nn.sigmoid
* tf.compat.v2.sigmoid
* tf.math.sigmoid
* tf.nn.sigmoid
* tf.sigmoid

tf.math.sigmoid(  
    x,  
    name=None  
)

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

### Used in the tutorials:

* [Convolutional Variational Autoencoder](https://www.tensorflow.org/beta/tutorials/generative/cvae)

Specifically, y = 1 / (1 + exp(-x)).

#### Args:

* **x**: A Tensor with type float16, float32, float64, complex64, or complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor with the same type as x.

#### Scipy Compatibility

Equivalent to scipy.special.expit

# tf.math.sign

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

Returns an element-wise indication of the sign of a number.

### Aliases:

* tf.compat.v1.math.sign
* tf.compat.v1.sign
* tf.compat.v2.math.sign
* tf.compat.v2.sign
* tf.math.sign
* tf.sign

tf.math.sign(  
    x,  
    name=None  
)

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

y = sign(x) = -1 if x < 0; 0 if x == 0; 1 if x > 0.

For complex numbers, y = sign(x) = x / |x| if x != 0, otherwise y = 0.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.sign(x.values, ...), x.dense\_shape)

# tf.math.sin

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

Computes sin of x element-wise.

### Aliases:

* tf.compat.v1.math.sin
* tf.compat.v1.sin
* tf.compat.v2.math.sin
* tf.compat.v2.sin
* tf.math.sin
* tf.sin

tf.math.sin(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.sinh

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

Computes hyperbolic sine of x element-wise.

### Aliases:

* tf.compat.v1.math.sinh
* tf.compat.v1.sinh
* tf.compat.v2.math.sinh
* tf.compat.v2.sinh
* tf.math.sinh
* tf.sinh

tf.math.sinh(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.softplus

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

Computes softplus: log(exp(features) + 1).

### Aliases:

* tf.compat.v1.math.softplus
* tf.compat.v1.nn.softplus
* tf.compat.v2.math.softplus
* tf.compat.v2.nn.softplus
* tf.math.softplus
* tf.nn.softplus

tf.math.softplus(  
    features,  
    name=None  
)

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

#### Args:

* **features**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as features.

# tf.math.sqrt

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

Computes square root of x element-wise.

### Aliases:

* tf.compat.v1.math.sqrt
* tf.compat.v1.sqrt
* tf.compat.v2.math.sqrt
* tf.compat.v2.sqrt
* tf.math.sqrt
* tf.sqrt

tf.math.sqrt(  
    x,  
    name=None  
)

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

### Used in the tutorials:

* [Transformer model for language understanding](https://www.tensorflow.org/beta/tutorials/text/transformer)

I.e., y=x=x1/2.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.sqrt(x.values, ...), x.dense\_shape)

# tf.math.square

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

Computes square of x element-wise.

### Aliases:

* tf.compat.v1.math.square
* tf.compat.v1.square
* tf.compat.v2.math.square
* tf.compat.v2.square
* tf.math.square
* tf.square

tf.math.square(  
    x,  
    name=None  
)

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

### Used in the guide:

* [Eager essentials](https://www.tensorflow.org/beta/guide/eager)
* [Writing layers and models with TensorFlow Keras](https://www.tensorflow.org/beta/guide/keras/custom_layers_and_models)

### Used in the tutorials:

* [Custom training: basics](https://www.tensorflow.org/beta/tutorials/eager/custom_training)
* [Tensors and Operations](https://www.tensorflow.org/beta/tutorials/eager/basics)

I.e., y=x∗x=x2.

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.square(x.values, ...), x.dense\_shape)

# tf.math.squared\_difference

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

Returns (x - y)(x - y) element-wise.

### Aliases:

* tf.compat.v1.math.squared\_difference
* tf.compat.v1.squared\_difference
* tf.compat.v2.math.squared\_difference
* tf.math.squared\_difference

tf.math.squared\_difference(  
    x,  
    y,  
    name=None  
)

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

NOTE: math.squared\_difference supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.subtract

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

Returns x - y element-wise.

### Aliases:

* tf.RaggedTensor.\_\_sub\_\_
* tf.compat.v1.RaggedTensor.\_\_sub\_\_
* tf.compat.v1.math.subtract
* tf.compat.v1.subtract
* tf.compat.v2.RaggedTensor.\_\_sub\_\_
* tf.compat.v2.math.subtract
* tf.compat.v2.subtract
* tf.math.subtract
* tf.subtract

tf.math.subtract(  
    x,  
    y,  
    name=None  
)

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

NOTE: Subtract supports broadcasting. More about broadcasting [here](http://docs.scipy.org/doc/numpy/user/basics.broadcasting.html)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, uint8, int8, uint16, int16, int32, int64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.tan

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

Computes tan of x element-wise.

### Aliases:

* tf.compat.v1.math.tan
* tf.compat.v1.tan
* tf.compat.v2.math.tan
* tf.compat.v2.tan
* tf.math.tan
* tf.tan

tf.math.tan(  
    x,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, int32, int64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.tanh

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

Computes hyperbolic tangent of x element-wise.

### Aliases:

* tf.compat.v1.math.tanh
* tf.compat.v1.nn.tanh
* tf.compat.v1.tanh
* tf.compat.v2.math.tanh
* tf.compat.v2.nn.tanh
* tf.compat.v2.tanh
* tf.math.tanh
* tf.nn.tanh
* tf.tanh

tf.math.tanh(  
    x,  
    name=None  
)

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

### Used in the tutorials:

* [Image Captioning with Attention](https://www.tensorflow.org/beta/tutorials/text/image_captioning)
* [Neural Machine Translation with Attention](https://www.tensorflow.org/beta/tutorials/text/nmt_with_attention)
* [tf.function](https://www.tensorflow.org/beta/tutorials/eager/tf_function)

#### Args:

* **x**: A Tensor. Must be one of the following types: bfloat16, half, float32, float64, complex64, complex128.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

If x is a SparseTensor, returns SparseTensor(x.indices, tf.math.tanh(x.values, ...), x.dense\_shape)

# tf.math.top\_k

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

Finds values and indices of the k largest entries for the last dimension.

### Aliases:

* tf.compat.v1.math.top\_k
* tf.compat.v1.nn.top\_k
* tf.compat.v2.math.top\_k
* tf.compat.v2.nn.top\_k
* tf.math.top\_k
* tf.nn.top\_k

tf.math.top\_k(  
    input,  
    k=1,  
    sorted=True,  
    name=None  
)

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

If the input is a vector (rank=1), finds the k largest entries in the vector and outputs their values and indices as vectors. Thus values[j] is the j-th largest entry in input, and its index is indices[j].

For matrices (resp. higher rank input), computes the top k entries in each row (resp. vector along the last dimension). Thus,

values.shape = indices.shape = input.shape[:-1] + [k]

If two elements are equal, the lower-index element appears first.

#### Args:

* **input**: 1-D or higher Tensor with last dimension at least k.
* **k**: 0-D int32 Tensor. Number of top elements to look for along the last dimension (along each row for matrices).
* **sorted**: If true the resulting k elements will be sorted by the values in descending order.
* **name**: Optional name for the operation.

#### Returns:

* **values**: The k largest elements along each last dimensional slice.
* **indices**: The indices of values within the last dimension of input.

# tf.math.truediv

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

Divides x / y elementwise (using Python 3 division operator semantics).

### Aliases:

* tf.RaggedTensor.\_\_truediv\_\_
* tf.compat.v1.RaggedTensor.\_\_truediv\_\_
* tf.compat.v1.math.truediv
* tf.compat.v1.truediv
* tf.compat.v2.RaggedTensor.\_\_truediv\_\_
* tf.compat.v2.math.truediv
* tf.compat.v2.truediv
* tf.math.truediv
* tf.truediv

tf.math.truediv(  
    x,  
    y,  
    name=None  
)

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

NOTE: Prefer using the Tensor operator or tf.divide which obey Python division operator semantics.

This function forces Python 3 division operator semantics where all integer arguments are cast to floating types first. This op is generated by normal x / y division in Python 3 and in Python 2.7 withfrom \_\_future\_\_ import division. If you want integer division that rounds down, use x // y or [tf.math.floordiv](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/math/floordiv).

x and y must have the same numeric type. If the inputs are floating point, the output will have the same type. If the inputs are integral, the inputs are cast to float32 for int8 and int16 and float64 for int32 and int64 (matching the behavior of Numpy).

#### Args:

* **x**: Tensor numerator of numeric type.
* **y**: Tensor denominator of numeric type.
* **name**: A name for the operation (optional).

#### Returns:

x / y evaluated in floating point.

#### Raises:

* **TypeError**: If x and y have different dtypes.

# tf.math.unsorted\_segment\_max

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

Computes the maximum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_max
* tf.compat.v1.unsorted\_segment\_max
* tf.compat.v2.math.unsorted\_segment\_max
* tf.math.unsorted\_segment\_max

tf.math.unsorted\_segment\_max(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

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

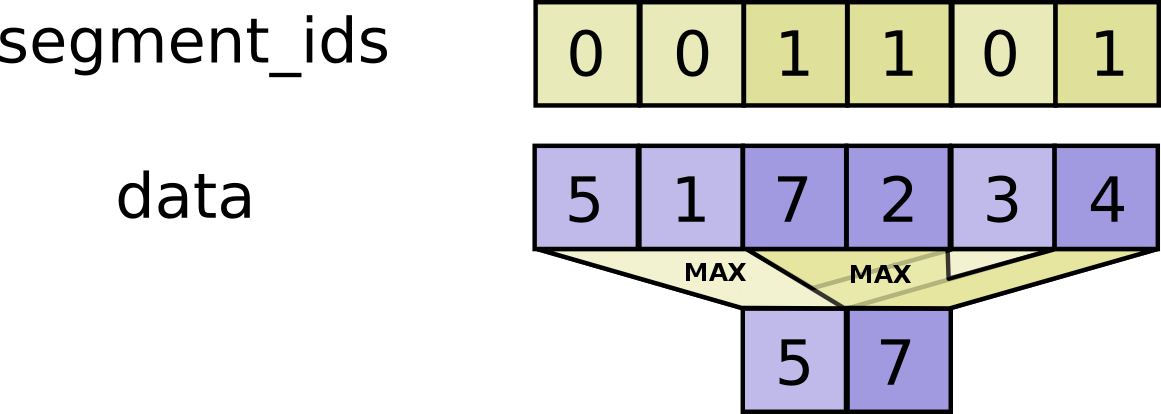
Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the maximum such that:

outputi=maxj...data[j...] where max is over tuples j... such that segment\_ids[j...] == i.

If the maximum is empty for a given segment ID i, it outputs the smallest possible value for the specific numeric type, output[i] = numeric\_limits<T>::lowest().

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.



#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_max(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  3, 3, 4],  
#       [5,  6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_mean

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

Computes the mean along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_mean
* tf.compat.v1.unsorted\_segment\_mean
* tf.compat.v2.math.unsorted\_segment\_mean
* tf.math.unsorted\_segment\_mean

tf.math.unsorted\_segment\_mean(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [here](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the mean of all entries belonging to a segment such that:

outputi=1/Ni∑j...data[j...] where the sum is over tuples j... such that segment\_ids[j...] == i with \N\_i\ being the number of occurrences of id \i\.

If there is no entry for a given segment ID i, it outputs 0.

If the given segment ID i is negative, the value is dropped and will not be added to the sum of the segment.

#### Args:

* **data**: A Tensor with floating point or complex dtype.
* **segment\_ids**: An integer tensor whose shape is a prefix of data.shape.
* **num\_segments**: An integer scalar Tensor. The number of distinct segment IDs.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has same shape as data, except for the first segment\_ids.rank dimensions, which are replaced with a single dimension which has size num\_segments.

# tf.math.unsorted\_segment\_min

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

Computes the minimum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_min
* tf.compat.v1.unsorted\_segment\_min
* tf.compat.v2.math.unsorted\_segment\_min
* tf.math.unsorted\_segment\_min

tf.math.unsorted\_segment\_min(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the minimum such that:

outputi=minj...data[j...] where min is over tuples j... such that segment\_ids[j...] == i.

If the minimum is empty for a given segment ID i, it outputs the largest possible value for the specific numeric type, output[i] = numeric\_limits<T>::max().

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_min(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 1,  2, 2, 1],  
#       [5,  6, 7, 8]]

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, int64, bfloat16, uint16, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_prod

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

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_prod
* tf.compat.v1.unsorted\_segment\_prod
* tf.compat.v2.math.unsorted\_segment\_prod
* tf.math.unsorted\_segment\_prod

tf.math.unsorted\_segment\_prod(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the product of all entries belonging to a segment such that:

outputi=∏j...data[j...] where the product is over tuples j... such that segment\_ids[j...] == i.

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_prod(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  6, 6, 4],  
#       [5,  6, 7, 8]]

If there is no entry for a given segment ID i, it outputs 1.

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_prod

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

Computes the product along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_prod
* tf.compat.v1.unsorted\_segment\_prod
* tf.compat.v2.math.unsorted\_segment\_prod
* tf.math.unsorted\_segment\_prod

tf.math.unsorted\_segment\_prod(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

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

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

This operator is similar to the unsorted segment sum operator found [(here)](https://www.tensorflow.org/versions/r2.0/api_docs/api_docs/python/math_ops#UnsortedSegmentSum). Instead of computing the sum over segments, it computes the product of all entries belonging to a segment such that:

outputi=∏j...data[j...] where the product is over tuples j... such that segment\_ids[j...] == i.

#### For example:

c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_prod(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 4,  6, 6, 4],  
#       [5,  6, 7, 8]]

If there is no entry for a given segment ID i, it outputs 1.

If the given segment ID i is negative, then the corresponding value is dropped, and will not be included in the result.

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.unsorted\_segment\_sum

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

Computes the sum along segments of a tensor.

### Aliases:

* tf.compat.v1.math.unsorted\_segment\_sum
* tf.compat.v1.unsorted\_segment\_sum
* tf.compat.v2.math.unsorted\_segment\_sum
* tf.math.unsorted\_segment\_sum

tf.math.unsorted\_segment\_sum(  
    data,  
    segment\_ids,  
    num\_segments,  
    name=None  
)

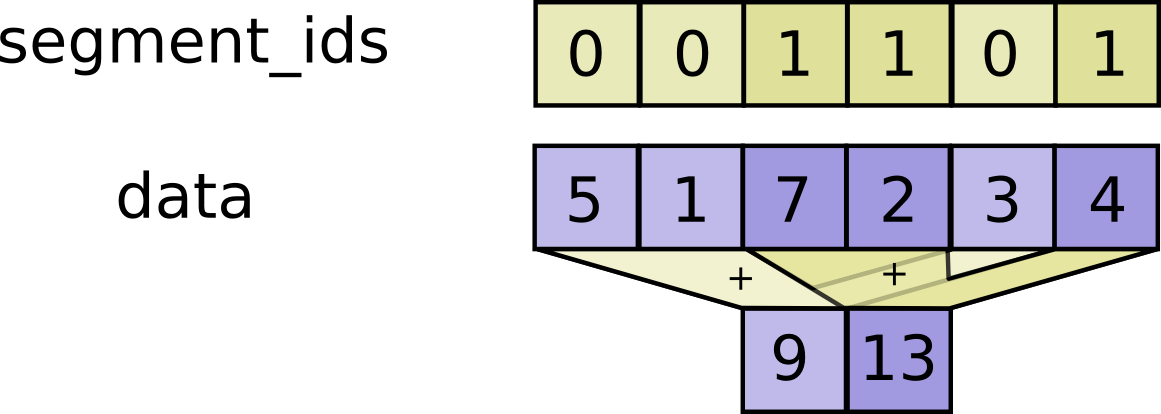
Defined in generated file: python/ops/gen\_math\_ops.py.

Read [the section on segmentation](https://tensorflow.org/api_docs/python/tf/math#Segmentation) for an explanation of segments.

Computes a tensor such that output[i]=∑j...data[j...] where the sum is over tuples j... such that segment\_ids[j...] == i. Unlike SegmentSum, segment\_ids need not be sorted and need not cover all values in the full range of valid values.

If the sum is empty for a given segment ID i, output[i] = 0. If the given segment ID i is negative, the value is dropped and will not be added to the sum of the segment.

num\_segments should equal the number of distinct segment IDs.



c = tf.constant([[1,2,3,4], [5,6,7,8], [4,3,2,1]])  
tf.unsorted\_segment\_sum(c, tf.constant([0, 1, 0]), num\_segments=2)  
# ==> [[ 5,  5, 5, 5],  
#       [5,  6, 7, 8]]

#### Args:

* **data**: A Tensor. Must be one of the following types: float32, float64, int32, uint8, int16, int8, complex64, int64, qint8, quint8, qint32, bfloat16, uint16, complex128, half, uint32, uint64.
* **segment\_ids**: A Tensor. Must be one of the following types: int32, int64. A tensor whose shape is a prefix of data.shape.
* **num\_segments**: A Tensor. Must be one of the following types: int32, int64.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as data.

# tf.math.xdivy

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

Returns 0 if x == 0, and x / y otherwise, elementwise.

### Aliases:

* tf.compat.v1.math.xdivy
* tf.compat.v2.math.xdivy
* tf.math.xdivy

tf.math.xdivy(  
    x,  
    y,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: half, float32, float64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.xlogy

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

Returns 0 if x == 0, and x \* log(y) otherwise, elementwise.

### Aliases:

* tf.compat.v1.math.xlogy
* tf.compat.v2.math.xlogy
* tf.math.xlogy

tf.math.xlogy(  
    x,  
    y,  
    name=None  
)

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

#### Args:

* **x**: A Tensor. Must be one of the following types: half, float32, float64, complex64, complex128.
* **y**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.

# tf.math.zero\_fraction

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

Returns the fraction of zeros in value.

### Aliases:

* tf.compat.v1.math.zero\_fraction
* tf.compat.v1.nn.zero\_fraction
* tf.compat.v2.math.zero\_fraction
* tf.compat.v2.nn.zero\_fraction
* tf.math.zero\_fraction
* tf.nn.zero\_fraction

tf.math.zero\_fraction(  
    value,  
    name=None  
)

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

If value is empty, the result is nan.

This is useful in summaries to measure and report sparsity. For example,

    z = tf.nn.relu(...)  
    summ = tf.compat.v1.summary.scalar('sparsity', tf.nn.zero\_fraction(z))

#### Args:

* **value**: A tensor of numeric type.
* **name**: A name for the operation (optional).

#### Returns:

The fraction of zeros in value, with type float32.

# tf.math.zeta

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

Compute the Hurwitz zeta function ζ(x,q).

### Aliases:

* tf.compat.v1.math.zeta
* tf.compat.v1.zeta
* tf.compat.v2.math.zeta
* tf.math.zeta

tf.math.zeta(  
    x,  
    q,  
    name=None  
)

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

The Hurwitz zeta function is defined as:

ζ(x,q)=∑n=0∞(q+n)−x

#### Args:

* **x**: A Tensor. Must be one of the following types: float32, float64.
* **q**: A Tensor. Must have the same type as x.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as x.