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.