Module: tf.image

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

Image processing and decoding ops.

See the [Images](https://tensorflow.org/api_guides/python/image) guide.

Classes

[class ResizeMethod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod)

Functions

[adjust\_brightness(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_brightness): Adjust the brightness of RGB or Grayscale images.

[adjust\_contrast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_contrast): Adjust contrast of RGB or grayscale images.

[adjust\_gamma(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_gamma): Performs Gamma Correction on the input image.

[adjust\_hue(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_hue): Adjust hue of RGB images.

[adjust\_jpeg\_quality(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_jpeg_quality): Adjust jpeg encoding quality of an RGB image.

[adjust\_saturation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/adjust_saturation): Adjust saturation of RGB images.

[central\_crop(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/central_crop): Crop the central region of the image(s).

[combined\_non\_max\_suppression(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/combined_non_max_suppression): Greedily selects a subset of bounding boxes in descending order of score.

[convert\_image\_dtype(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/convert_image_dtype): Convert image to dtype, scaling its values if needed.

[crop\_and\_resize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/crop_and_resize): Extracts crops from the input image tensor and resizes them.

[crop\_to\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/crop_to_bounding_box): Crops an image to a specified bounding box.

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

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

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

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

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

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

[draw\_bounding\_boxes(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes): Draw bounding boxes on a batch of images.

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

[encode\_png(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/encode_png): PNG-encode an image.

[extract\_glimpse(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/extract_glimpse): Extracts a glimpse from the input tensor.

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

[extract\_patches(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/extract_patches): Extract patches from images and put them in the "depth" output dimension.

[flip\_left\_right(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_left_right): Flip an image horizontally (left to right).

[flip\_up\_down(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/flip_up_down): Flip an image vertically (upside down).

[grayscale\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/grayscale_to_rgb): Converts one or more images from Grayscale to RGB.

[hsv\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/hsv_to_rgb): Convert one or more images from HSV to RGB.

[image\_gradients(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/image_gradients): Returns image gradients (dy, dx) for each color channel.

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

[non\_max\_suppression(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_overlaps(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_overlaps): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_padded(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_padded): Greedily selects a subset of bounding boxes in descending order of score.

[non\_max\_suppression\_with\_scores(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression_with_scores): Greedily selects a subset of bounding boxes in descending order of score.

[pad\_to\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/pad_to_bounding_box): Pad image with zeros to the specified height and width.

[per\_image\_standardization(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/per_image_standardization): Linearly scales each image in image to have mean 0 and variance 1.

[psnr(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/psnr): Returns the Peak Signal-to-Noise Ratio between a and b.

[random\_brightness(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_brightness): Adjust the brightness of images by a random factor.

[random\_contrast(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_contrast): Adjust the contrast of an image or images by a random factor.

[random\_crop(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_crop): Randomly crops a tensor to a given size.

[random\_flip\_left\_right(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_flip_left_right): Randomly flip an image horizontally (left to right).

[random\_flip\_up\_down(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_flip_up_down): Randomly flips an image vertically (upside down).

[random\_hue(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_hue): Adjust the hue of RGB images by a random factor.

[random\_jpeg\_quality(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_jpeg_quality): Randomly changes jpeg encoding quality for inducing jpeg noise.

[random\_saturation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/random_saturation): Adjust the saturation of RGB images by a random factor.

[resize(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize): Resize images to size using the specified method.

[resize\_with\_crop\_or\_pad(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_crop_or_pad): Crops and/or pads an image to a target width and height.

[resize\_with\_pad(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_pad): Resizes and pads an image to a target width and height.

[rgb\_to\_grayscale(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_grayscale): Converts one or more images from RGB to Grayscale.

[rgb\_to\_hsv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_hsv): Converts one or more images from RGB to HSV.

[rgb\_to\_yiq(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_yiq): Converts one or more images from RGB to YIQ.

[rgb\_to\_yuv(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rgb_to_yuv): Converts one or more images from RGB to YUV.

[rot90(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90): Rotate image(s) counter-clockwise by 90 degrees.

[sample\_distorted\_bounding\_box(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sample_distorted_bounding_box): Generate a single randomly distorted bounding box for an image.

[sobel\_edges(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/sobel_edges): Returns a tensor holding Sobel edge maps.

[ssim(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ssim): Computes SSIM index between img1 and img2.

[ssim\_multiscale(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ssim_multiscale): Computes the MS-SSIM between img1 and img2.

[total\_variation(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/total_variation): Calculate and return the total variation for one or more images.

[transpose(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/transpose): Transpose image(s) by swapping the height and width dimension.

[yiq\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/yiq_to_rgb): Converts one or more images from YIQ to RGB.

[yuv\_to\_rgb(...)](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/yuv_to_rgb): Converts one or more images from YUV to RGB.

# tf.compat.v1.image.resize\_area

Resize images to size using area interpolation.

tf.compat.v1.image.resize\_area(  
    images,  
    size,  
    align\_corners=False,  
    name=None  
)

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

Input images can be of different types but output images are always float.

The range of pixel values for the output image might be slightly different from the range for the input image because of limited numerical precision. To guarantee an output range, for example [0.0, 1.0], apply [tf.clip\_by\_value](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/clip_by_value) to the output.

Each output pixel is computed by first transforming the pixel's footprint into the input tensor and then averaging the pixels that intersect the footprint. An input pixel's contribution to the average is weighted by the fraction of its area that intersects the footprint. This is the same as OpenCV's INTER\_AREA.

#### Args:

* images: A Tensor. Must be one of the following types: int8, uint8, int16, uint16, int32, int64, half, float32, float64. 4-D with shape [batch, height, width, channels].
* size: A 1-D int32 Tensor of 2 elements: new\_height, new\_width. The new size for the images.
* align\_corners: An optional bool. Defaults to False. If true, the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. Defaults to false.
* name: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.compat.v1.image.resize\_bicubic

tf.compat.v1.image.resize\_bicubic(  
    images,  
    size,  
    align\_corners=False,  
    name=None,  
    half\_pixel\_centers=False  
)

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

# tf.compat.v1.image.resize\_bilinear

tf.compat.v1.image.resize\_bilinear(  
    images,  
    size,  
    align\_corners=False,  
    name=None,  
    half\_pixel\_centers=False  
)

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

# tf.compat.v1.image.resize\_image\_with\_pad

Resizes and pads an image to a target width and height.

tf.compat.v1.image.resize\_image\_with\_pad(  
    image,  
    target\_height,  
    target\_width,  
    method=ResizeMethodV1.BILINEAR,  
    align\_corners=False  
)

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

Resizes an image to a target width and height by keeping the aspect ratio the same without distortion. If the target dimensions don't match the image dimensions, the image is resized and then padded with zeroes to match requested dimensions.

#### Args:

* image: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* target\_height: Target height.
* target\_width: Target width.
* method: Method to use for resizing image. See resize\_images()
* align\_corners: bool. If True, the centers of the 4 corner pixels of the input and output tensors are aligned, preserving the values at the corner pixels. Defaults to False.

#### Raises:

* ValueError: if target\_height or target\_width are zero or negative.

#### Returns:

Resized and padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.compat.v1.image.resize\_nearest\_neighbor

tf.compat.v1.image.resize\_nearest\_neighbor(  
    images,  
    size,  
    align\_corners=False,  
    name=None,  
    half\_pixel\_centers=False  
)

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

# tf.image.adjust\_brightness

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

Adjust the brightness of RGB or Grayscale images.

### Aliases:

* tf.compat.v1.image.adjust\_brightness
* tf.compat.v2.image.adjust\_brightness
* tf.image.adjust\_brightness

tf.image.adjust\_brightness(  
    image,  
    delta  
)

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

This is a convenience method that converts RGB images to float representation, adjusts their brightness, and then converts them back to the original data type. If several adjustments are chained, it is advisable to minimize the number of redundant conversions.

The value delta is added to all components of the tensor image. image is converted to float and scaled appropriately if it is in fixed-point representation, and delta is converted to the same data type. For regular images, delta should be in the range [0,1), as it is added to the image in floating point representation, where pixel values are in the [0,1) range.

#### Args:

* **image**: RGB image or images to adjust.
* **delta**: A scalar. Amount to add to the pixel values.

#### Returns:

A brightness-adjusted tensor of the same shape and type as image.

# tf.image.adjust\_contrast

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

Adjust contrast of RGB or grayscale images.

### Aliases:

* tf.compat.v1.image.adjust\_contrast
* tf.compat.v2.image.adjust\_contrast
* tf.image.adjust\_contrast

tf.image.adjust\_contrast(  
    images,  
    contrast\_factor  
)

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

This is a convenience method that converts RGB images to float representation, adjusts their contrast, and then converts them back to the original data type. If several adjustments are chained, it is advisable to minimize the number of redundant conversions.

images is a tensor of at least 3 dimensions. The last 3 dimensions are interpreted as [height, width, channels]. The other dimensions only represent a collection of images, such as [batch, height, width, channels].

Contrast is adjusted independently for each channel of each image.

For each channel, this Op computes the mean of the image pixels in the channel and then adjusts each component x of each pixel to (x - mean) \* contrast\_factor + mean.

#### Args:

* **images**: Images to adjust. At least 3-D.
* **contrast\_factor**: A float multiplier for adjusting contrast.

#### Returns:

The contrast-adjusted image or images.

# tf.image.adjust\_gamma

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

Performs Gamma Correction on the input image.

### Aliases:

* tf.compat.v1.image.adjust\_gamma
* tf.compat.v2.image.adjust\_gamma
* tf.image.adjust\_gamma

tf.image.adjust\_gamma(  
    image,  
    gamma=1,  
    gain=1  
)

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

Also known as Power Law Transform. This function converts the input images at first to float representation, then transforms them pixelwise according to the equation Out = gain \* In\*\*gamma, and then converts the back to the original data type.

#### Args:

* **image**: RGB image or images to adjust.
* **gamma**: A scalar or tensor. Non negative real number.
* **gain**: A scalar or tensor. The constant multiplier.

#### Returns:

A Tensor. A Gamma-adjusted tensor of the same shape and type as image.

#### Raises:

* **ValueError**: If gamma is negative.

#### Notes:

For gamma greater than 1, the histogram will shift towards left and the output image will be darker than the input image. For gamma less than 1, the histogram will shift towards right and the output image will be brighter than the input image.

#### References:

[1] http://en.wikipedia.org/wiki/Gamma\_correction

# tf.image.adjust\_hue

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

Adjust hue of RGB images.

### Aliases:

* tf.compat.v1.image.adjust\_hue
* tf.compat.v2.image.adjust\_hue
* tf.image.adjust\_hue

tf.image.adjust\_hue(  
    image,  
    delta,  
    name=None  
)

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

This is a convenience method that converts an RGB image to float representation, converts it to HSV, add an offset to the hue channel, converts back to RGB and then back to the original data type. If several adjustments are chained it is advisable to minimize the number of redundant conversions.

image is an RGB image. The image hue is adjusted by converting the image(s) to HSV and rotating the hue channel (H) by delta. The image is then converted back to RGB.

delta must be in the interval [-1, 1].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **delta**: float. How much to add to the hue channel.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.adjust\_jpeg\_quality

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

Adjust jpeg encoding quality of an RGB image.

### Aliases:

* tf.compat.v1.image.adjust\_jpeg\_quality
* tf.compat.v2.image.adjust\_jpeg\_quality
* tf.image.adjust\_jpeg\_quality

tf.image.adjust\_jpeg\_quality(  
    image,  
    jpeg\_quality,  
    name=None  
)

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

This is a convenience method that adjusts jpeg encoding quality of an RGB image.

image is an RGB image. The image's encoding quality is adjusted to jpeg\_quality. jpeg\_qualitymust be in the interval [0, 100].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **jpeg\_quality**: Python int or Tensor of type int32. jpeg encoding quality.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.adjust\_saturation

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

Adjust saturation of RGB images.

### Aliases:

* tf.compat.v1.image.adjust\_saturation
* tf.compat.v2.image.adjust\_saturation
* tf.image.adjust\_saturation

tf.image.adjust\_saturation(  
    image,  
    saturation\_factor,  
    name=None  
)

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

This is a convenience method that converts RGB images to float representation, converts them to HSV, add an offset to the saturation channel, converts back to RGB and then back to the original data type. If several adjustments are chained it is advisable to minimize the number of redundant conversions.

image is an RGB image or images. The image saturation is adjusted by converting the images to HSV and multiplying the saturation (S) channel by saturation\_factor and clipping. The images are then converted back to RGB.

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **saturation\_factor**: float. Factor to multiply the saturation by.
* **name**: A name for this operation (optional).

#### Returns:

Adjusted image(s), same shape and DType as image.

# tf.image.central\_crop

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

Crop the central region of the image(s).

### Aliases:

* tf.compat.v1.image.central\_crop
* tf.compat.v2.image.central\_crop
* tf.image.central\_crop

tf.image.central\_crop(  
    image,  
    central\_fraction  
)

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

Remove the outer parts of an image but retain the central region of the image along each dimension. If we specify central\_fraction = 0.5, this function returns the region marked with "X" in the below diagram.

 --------  
|        |  
|  XXXX  |  
|  XXXX  |  
|        |   where "X" is the central 50% of the image.  
 --------

This function works on either a single image (image is a 3-D Tensor), or a batch of images (image is a 4-D Tensor).

#### Args:

* **image**: Either a 3-D float Tensor of shape [height, width, depth], or a 4-D Tensor of shape [batch\_size, height, width, depth].
* **central\_fraction**: float (0, 1], fraction of size to crop

#### Raises:

* **ValueError**: if central\_crop\_fraction is not within (0, 1].

#### Returns:

3-D / 4-D float Tensor, as per the input.

# tf.image.combined\_non\_max\_suppression

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.combined\_non\_max\_suppression
* tf.compat.v2.image.combined\_non\_max\_suppression
* tf.image.combined\_non\_max\_suppression

tf.image.combined\_non\_max\_suppression(  
    boxes,  
    scores,  
    max\_output\_size\_per\_class,  
    max\_total\_size,  
    iou\_threshold=0.5,  
    score\_threshold=float('-inf'),  
    pad\_per\_class=False,  
    clip\_boxes=True,  
    name=None  
)

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

This operation performs non\_max\_suppression on the inputs per batch, across all classes. Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Also note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is the final boxes, scores and classes tensor returned after performing non\_max\_suppression.

#### Args:

* **boxes**: A 4-D float Tensor of shape [batch\_size, num\_boxes, q, 4]. If q is 1 then same boxes are used for all classes otherwise, if q is equal to number of classes, class-specific boxes are used.
* **scores**: A 3-D float Tensor of shape [batch\_size, num\_boxes, num\_classes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size\_per\_class**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression per class
* **max\_total\_size**: A scalar representing maximum number of boxes retained over all classes.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **pad\_per\_class**: If false, the output nmsed boxes, scores and classes are padded/clipped to max\_total\_size. If true, the output nmsed boxes, scores and classes are padded to be of length max\_size\_per\_class\*num\_classes, unless it exceeds max\_total\_size in which case it is clipped to max\_total\_size. Defaults to false.
* **clip\_boxes**: If true, the coordinates of output nmsed boxes will be clipped to [0, 1]. If false, output the box coordinates as it is. Defaults to true.
* **name**: A name for the operation (optional).

#### Returns:

'nmsed\_boxes': A [batch\_size, max\_detections, 4] float32 tensor containing the non-max suppressed boxes. 'nmsed\_scores': A [batch\_size, max\_detections] float32 tensor containing the scores for the boxes. 'nmsed\_classes': A [batch\_size, max\_detections] float32 tensor containing the class for boxes. 'valid\_detections': A [batch\_size] int32 tensor indicating the number of valid detections per batch item. Only the top valid\_detections[i] entries in nms\_boxes[i], nms\_scores[i] and nms\_class[i] are valid. The rest of the entries are zero paddings.

# tf.image.convert\_image\_dtype

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

Convert image to dtype, scaling its values if needed.

### Aliases:

* tf.compat.v1.image.convert\_image\_dtype
* tf.compat.v2.image.convert\_image\_dtype
* tf.image.convert\_image\_dtype

tf.image.convert\_image\_dtype(  
    image,  
    dtype,  
    saturate=False,  
    name=None  
)

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

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

Images that are represented using floating point values are expected to have values in the range [0,1). Image data stored in integer data types are expected to have values in the range [0,MAX], where MAXis the largest positive representable number for the data type.

This op converts between data types, scaling the values appropriately before casting.

Note that converting from floating point inputs to integer types may lead to over/underflow problems. Set saturate to True to avoid such problem in problematic conversions. If enabled, saturation will clip the output into the allowed range before performing a potentially dangerous cast (and only before performing such a cast, i.e., when casting from a floating point to an integer type, and when casting from a signed to an unsigned type; saturate has no effect on casts between floats, or on casts that increase the type's range).

#### Args:

* **image**: An image.
* **dtype**: A DType to convert image to.
* **saturate**: If True, clip the input before casting (if necessary).
* **name**: A name for this operation (optional).

#### Returns:

image, converted to dtype.

# tf.image.crop\_and\_resize

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

Extracts crops from the input image tensor and resizes them.

### Aliases:

* tf.compat.v2.image.crop\_and\_resize
* tf.image.crop\_and\_resize

tf.image.crop\_and\_resize(  
    image,  
    boxes,  
    box\_indices,  
    crop\_size,  
    method='bilinear',  
    extrapolation\_value=0,  
    name=None  
)

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

Extracts crops from the input image tensor and resizes them using bilinear sampling or nearest neighbor sampling (possibly with aspect ratio change) to a common output size specified by crop\_size. This is more general than the crop\_to\_bounding\_box op which extracts a fixed size slice from the input image and does not allow resizing or aspect ratio change.

Returns a tensor with crops from the input image at positions defined at the bounding box locations in boxes. The cropped boxes are all resized (with bilinear or nearest neighbor interpolation) to a fixedsize = [crop\_height, crop\_width]. The result is a 4-D tensor [num\_boxes, crop\_height, crop\_width, depth]. The resizing is corner aligned. In particular, if boxes = [[0, 0, 1, 1]], the method will give identical results to using tf.compat.v1.image.resize\_bilinear() ortf.compat.v1.image.resize\_nearest\_neighbor()(depends on the method argument) withalign\_corners=True.

#### Args:

* **image**: A 4-D tensor of shape [batch, image\_height, image\_width, depth]. Both image\_height and image\_width need to be positive.
* **boxes**: A 2-D tensor of shape [num\_boxes, 4]. The i-th row of the tensor specifies the coordinates of a box in the box\_ind[i] image and is specified in normalized coordinates [y1, x1, y2, x2]. A normalized coordinate value of y is mapped to the image coordinate at y \* (image\_height - 1), so as the [0, 1] interval of normalized image height is mapped to [0, image\_height - 1] in image height coordinates. We do allow y1 > y2, in which case the sampled crop is an up-down flipped version of the original image. The width dimension is treated similarly. Normalized coordinates outside the [0, 1] range are allowed, in which case we use extrapolation\_value to extrapolate the input image values.
* **box\_indices**: A 1-D tensor of shape [num\_boxes] with int32 values in [0, batch). The value of box\_ind[i] specifies the image that the i-th box refers to.
* **crop\_size**: A 1-D tensor of 2 elements, size = [crop\_height, crop\_width]. All cropped image patches are resized to this size. The aspect ratio of the image content is not preserved. Both crop\_height and crop\_width need to be positive.
* **method**: An optional string specifying the sampling method for resizing. It can be either "bilinear" or "nearest" and default to "bilinear". Currently two sampling methods are supported: Bilinear and Nearest Neighbor.
* **extrapolation\_value**: An optional float. Defaults to 0. Value used for extrapolation, when applicable.
* **name**: A name for the operation (optional).

#### Returns:

A 4-D tensor of shape [num\_boxes, crop\_height, crop\_width, depth].

# tf.image.crop\_to\_bounding\_box

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

Crops an image to a specified bounding box.

### Aliases:

* tf.compat.v1.image.crop\_to\_bounding\_box
* tf.compat.v2.image.crop\_to\_bounding\_box
* tf.image.crop\_to\_bounding\_box

tf.image.crop\_to\_bounding\_box(  
    image,  
    offset\_height,  
    offset\_width,  
    target\_height,  
    target\_width  
)

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

This op cuts a rectangular part out of image. The top-left corner of the returned image is at offset\_height, offset\_width in image, and its lower-right corner is at offset\_height + target\_height, offset\_width + target\_width.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **offset\_height**: Vertical coordinate of the top-left corner of the result in the input.
* **offset\_width**: Horizontal coordinate of the top-left corner of the result in the input.
* **target\_height**: Height of the result.
* **target\_width**: Width of the result.

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, target\_height, target\_width, channels] If image was 3-D, a 3-D float Tensor of shape [target\_height, target\_width, channels]

#### Raises:

* **ValueError**: If the shape of image is incompatible with the offset\_\* or target\_\*arguments, or either offset\_height or offset\_width is negative, or either target\_heightor target\_width is not positive.

# tf.image.draw\_bounding\_boxes

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

Draw bounding boxes on a batch of images.

### Aliases:

* tf.compat.v2.image.draw\_bounding\_boxes
* tf.image.draw\_bounding\_boxes

tf.image.draw\_bounding\_boxes(  
    images,  
    boxes,  
    colors,  
    name=None  
)

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

Outputs a copy of images but draws on top of the pixels zero or more bounding boxes specified by the locations in boxes. The coordinates of the each bounding box in boxes are encoded as [y\_min, x\_min, y\_max, x\_max]. The bounding box coordinates are floats in [0.0, 1.0] relative to the width and height of the underlying image.

For example, if an image is 100 x 200 pixels (height x width) and the bounding box is [0.1, 0.2, 0.5, 0.9], the upper-left and bottom-right coordinates of the bounding box will be (40, 10) to (180, 50) (in (x,y) coordinates).

Parts of the bounding box may fall outside the image.

#### Args:

* **images**: A Tensor. Must be one of the following types: float32, half. 4-D with shape [batch, height, width, depth]. A batch of images.
* **boxes**: A Tensor of type float32. 3-D with shape [batch, num\_bounding\_boxes, 4]containing bounding boxes.
* **colors**: A Tensor of type float32. 2-D. A list of RGBA colors to cycle through for the boxes.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.encode\_png

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

PNG-encode an image.

### Aliases:

* tf.compat.v1.image.encode\_png
* tf.compat.v2.image.encode\_png
* tf.image.encode\_png

tf.image.encode\_png(  
    image,  
    compression=-1,  
    name=None  
)

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

image is a 3-D uint8 or uint16 Tensor of shape [height, width, channels] where channels is:

* 1: for grayscale.
* 2: for grayscale + alpha.
* 3: for RGB.
* 4: for RGBA.

The ZLIB compression level, compression, can be -1 for the PNG-encoder default or a value from 0 to 9. 9 is the highest compression level, generating the smallest output, but is slower.

#### Args:

* **image**: A Tensor. Must be one of the following types: uint8, uint16. 3-D with shape [height, width, channels].
* **compression**: An optional int. Defaults to -1. Compression level.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type string.

# tf.image.extract\_glimpse

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

Extracts a glimpse from the input tensor.

### Aliases:

* tf.compat.v2.image.extract\_glimpse
* tf.image.extract\_glimpse

tf.image.extract\_glimpse(  
    input,  
    size,  
    offsets,  
    centered=True,  
    normalized=True,  
    noise='uniform',  
    name=None  
)

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

Returns a set of windows called glimpses extracted at location offsets from the input tensor. If the windows only partially overlaps the inputs, the non overlapping areas will be filled with random noise.

The result is a 4-D tensor of shape [batch\_size, glimpse\_height, glimpse\_width, channels]. The channels and batch dimensions are the same as that of the input tensor. The height and width of the output windows are specified in the size parameter.

The argument normalized and centered controls how the windows are built:

* If the coordinates are normalized but not centered, 0.0 and 1.0 correspond to the minimum and maximum of each height and width dimension.
* If the coordinates are both normalized and centered, they range from -1.0 to 1.0. The coordinates (-1.0, -1.0) correspond to the upper left corner, the lower right corner is located at (1.0, 1.0) and the center is at (0, 0).
* If the coordinates are not normalized they are interpreted as numbers of pixels.

#### Args:

* **input**: A Tensor of type float32. A 4-D float tensor of shape [batch\_size, height, width, channels].
* **size**: A Tensor of type int32. A 1-D tensor of 2 elements containing the size of the glimpses to extract. The glimpse height must be specified first, following by the glimpse width.
* **offsets**: A Tensor of type float32. A 2-D integer tensor of shape [batch\_size, 2]containing the y, x locations of the center of each window.
* **centered**: An optional bool. Defaults to True. indicates if the offset coordinates are centered relative to the image, in which case the (0, 0) offset is relative to the center of the input images. If false, the (0,0) offset corresponds to the upper left corner of the input images.
* **normalized**: An optional bool. Defaults to True. indicates if the offset coordinates are normalized.
* **noise**: An optional string. Defaults to uniform. indicates if the noise should be uniform(uniform distribution), gaussian (gaussian distribution), or zero (zero padding).
* **name**: A name for the operation (optional).

#### Returns:

A Tensor of type float32.

# tf.image.extract\_patches

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

Extract patches from images and put them in the "depth" output dimension.

### Aliases:

* tf.compat.v1.image.extract\_patches
* tf.compat.v2.image.extract\_patches
* tf.image.extract\_patches

tf.image.extract\_patches(  
    images,  
    sizes,  
    strides,  
    rates,  
    padding,  
    name=None  
)

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

#### Args:

* **images**: A 4-D Tensor with shape `[batch, in\_rows, in\_cols, depth]
* **sizes**: The size of the sliding window for each dimension of images.
* **strides**: A 1-D Tensor of length 4. How far the centers of two consecutive patches are in the images. Must be: [1, stride\_rows, stride\_cols, 1].
* **rates**: A 1-D Tensor of length 4. Must be: [1, rate\_rows, rate\_cols, 1]. This is the input stride, specifying how far two consecutive patch samples are in the input. Equivalent to extracting patches with patch\_sizes\_eff = patch\_sizes + (patch\_sizes - 1) \* (rates - 1), followed by subsampling them spatially by a factor of rates. This is equivalent to ratein dilated (a.k.a. Atrous) convolutions.
* **padding**: The type of padding algorithm to use. We specify the size-related attributes as: ```python ksizes = [1, ksize\_rows, ksize\_cols, 1] strides = [1, strides\_rows, strides\_cols, 1] rates = [1, rates\_rows, rates\_cols, 1]
* **name**: A name for the operation (optional).

#### Returns:

A 4-D Tensor. Has the same type as images, and with shape [batch, out\_rows, out\_cols, ksize\_rows \* ksize\_cols \* depth] containing image patches with size ksize\_rows x ksize\_cols x depth vectorized in the "depth" dimension. Note out\_rows and out\_cols are the dimensions of the output patches.

# tf.image.flip\_left\_right

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

Flip an image horizontally (left to right).

### Aliases:

* tf.compat.v1.image.flip\_left\_right
* tf.compat.v2.image.flip\_left\_right
* tf.image.flip\_left\_right

tf.image.flip\_left\_right(image)

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

### Used in the tutorials:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Outputs the contents of image flipped along the width dimension.

See also reverse().

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.flip\_up\_down

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

Flip an image vertically (upside down).

### Aliases:

* tf.compat.v1.image.flip\_up\_down
* tf.compat.v2.image.flip\_up\_down
* tf.image.flip\_up\_down

tf.image.flip\_up\_down(image)

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

Outputs the contents of image flipped along the height dimension.

See also reverse().

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].

#### Returns:

A Tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.grayscale\_to\_rgb

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

Converts one or more images from Grayscale to RGB.

### Aliases:

* tf.compat.v1.image.grayscale\_to\_rgb
* tf.compat.v2.image.grayscale\_to\_rgb
* tf.image.grayscale\_to\_rgb

tf.image.grayscale\_to\_rgb(  
    images,  
    name=None  
)

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

Outputs a tensor of the same DType and rank as images. The size of the last dimension of the output is 3, containing the RGB value of the pixels. The input images' last dimension must be size 1.

#### Args:

* **images**: The Grayscale tensor to convert. Last dimension must be size 1.
* **name**: A name for the operation (optional).

#### Returns:

The converted grayscale image(s).

# tf.image.hsv\_to\_rgb

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

Convert one or more images from HSV to RGB.

### Aliases:

* tf.compat.v1.image.hsv\_to\_rgb
* tf.compat.v2.image.hsv\_to\_rgb
* tf.image.hsv\_to\_rgb

tf.image.hsv\_to\_rgb(  
    images,  
    name=None  
)

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

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the value in images are in [0,1].

See rgb\_to\_hsv for a description of the HSV encoding.

#### Args:

* **images**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 1-D or higher rank. HSV data to convert. Last dimension must be size 3.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.image\_gradients

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

Returns image gradients (dy, dx) for each color channel.

### Aliases:

* tf.compat.v1.image.image\_gradients
* tf.compat.v2.image.image\_gradients
* tf.image.image\_gradients

tf.image.image\_gradients(image)

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

Both output tensors have the same shape as the input: [batch\_size, h, w, d]. The gradient values are organized so that [I(x+1, y) - I(x, y)] is in location (x, y). That means that dy will always have zeros in the last row, and dx will always have zeros in the last column.

#### Arguments:

* **image**: Tensor with shape [batch\_size, h, w, d].

#### Returns:

Pair of tensors (dy, dx) holding the vertical and horizontal image gradients (1-step finite difference).

#### Raises:

* **ValueError**: If image is not a 4D tensor.

# tf.image.non\_max\_suppression

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression
* tf.compat.v2.image.non\_max\_suppression
* tf.image.non\_max\_suppression

tf.image.non\_max\_suppression(  
    boxes,  
    scores,  
    max\_output\_size,  
    iou\_threshold=0.5,  
    score\_threshold=float('-inf'),  
    name=None  
)

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

Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather)operation. For example:

selected\_indices = tf.image.non\_max\_suppression(  
    boxes, scores, max\_output\_size, iou\_threshold)  
selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.

# tf.image.non\_max\_suppression\_overlaps

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_overlaps
* tf.compat.v2.image.non\_max\_suppression\_overlaps
* tf.image.non\_max\_suppression\_overlaps

tf.image.non\_max\_suppression\_overlaps(  
    overlaps,  
    scores,  
    max\_output\_size,  
    overlap\_threshold=0.5,  
    score\_threshold=float('-inf'),  
    name=None  
)

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

Prunes away boxes that have high overlap with previously selected boxes. N-by-n overlap values are supplied as square matrix. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) operation. For example:

selected\_indices = tf.image.non\_max\_suppression\_overlaps(  
    overlaps, scores, max\_output\_size, iou\_threshold)  
selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **overlaps**: A 2-D float Tensor of shape [num\_boxes, num\_boxes].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **overlap\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to the provided overlap values.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the overlaps tensor, where M <= max\_output\_size.

# tf.image.non\_max\_suppression\_padded

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_padded
* tf.compat.v2.image.non\_max\_suppression\_padded
* tf.image.non\_max\_suppression\_padded

tf.image.non\_max\_suppression\_padded(  
    boxes,  
    scores,  
    max\_output\_size,  
    iou\_threshold=0.5,  
    score\_threshold=float('-inf'),  
    pad\_to\_max\_output\_size=False,  
    name=None  
)

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

Performs algorithmically equivalent operation to tf.image.non\_max\_suppression, with the addition of an optional parameter which zero-pads the output to be of size max\_output\_size. The output of this operation is a tuple containing the set of integers indexing into the input collection of bounding boxes representing the selected boxes and the number of valid indices in the index set. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice) and [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather) operations. For example:

selected\_indices\_padded, num\_valid = tf.image.non\_max\_suppression\_padded(  
    boxes, scores, max\_output\_size, iou\_threshold,  
    score\_threshold, pad\_to\_max\_output\_size=True)  
selected\_indices = tf.slice(  
    selected\_indices\_padded, tf.constant([0]), num\_valid)  
selected\_boxes = tf.gather(boxes, selected\_indices)

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **pad\_to\_max\_output\_size**: bool. If True, size of selected\_indices output is padded to max\_output\_size.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.
* **valid\_outputs**: A scalar integer Tensor denoting how many elements in selected\_indicesare valid. Valid elements occur first, then padding.

# tf.image.non\_max\_suppression\_with\_scores

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

Greedily selects a subset of bounding boxes in descending order of score.

### Aliases:

* tf.compat.v1.image.non\_max\_suppression\_with\_scores
* tf.compat.v2.image.non\_max\_suppression\_with\_scores
* tf.image.non\_max\_suppression\_with\_scores

tf.image.non\_max\_suppression\_with\_scores(  
    boxes,  
    scores,  
    max\_output\_size,  
    iou\_threshold=0.5,  
    score\_threshold=float('-inf'),  
    soft\_nms\_sigma=0.0,  
    name=None  
)

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

Prunes away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm. The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the [tf.gather](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/gather)operation. For example:

selected\_indices, selected\_scores = tf.image.non\_max\_suppression\_v2(  
    boxes, scores, max\_output\_size, iou\_threshold=1.0, score\_threshold=0.1,  
    soft\_nms\_sigma=0.5)  
selected\_boxes = tf.gather(boxes, selected\_indices)

This function generalizes the [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression) op by also supporting a Soft-NMS (with Gaussian weighting) mode (c.f. Bodla et al, https://arxiv.org/abs/1704.04503) where boxes reduce the score of other overlapping boxes instead of directly causing them to be pruned. Consequently, in contrast to [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression),tf.image.non\_max\_suppression\_v2 returns the new scores of each input box in the second output, selected\_scores.

To enable this Soft-NMS mode, set the soft\_nms\_sigma parameter to be larger than 0. When soft\_nms\_sigma equals 0, the behavior of tf.image.non\_max\_suppression\_v2 is identical to that of [tf.image.non\_max\_suppression](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/non_max_suppression) (except for the extra output) both in function and in running time.

#### Args:

* **boxes**: A 2-D float Tensor of shape [num\_boxes, 4].
* **scores**: A 1-D float Tensor of shape [num\_boxes] representing a single score corresponding to each box (each row of boxes).
* **max\_output\_size**: A scalar integer Tensor representing the maximum number of boxes to be selected by non max suppression.
* **iou\_threshold**: A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
* **score\_threshold**: A float representing the threshold for deciding when to remove boxes based on score.
* **soft\_nms\_sigma**: A scalar float representing the Soft NMS sigma parameter; See Bodla et al, https://arxiv.org/abs/1704.04503). When soft\_nms\_sigma=0.0 (which is default), we fall back to standard (hard) NMS.
* **name**: A name for the operation (optional).

#### Returns:

* **selected\_indices**: A 1-D integer Tensor of shape [M] representing the selected indices from the boxes tensor, where M <= max\_output\_size.
* **selected\_scores**: A 1-D float tensor of shape [M] representing the corresponding scores for each selected box, where M <= max\_output\_size. Scores only differ from corresponding input scores when using Soft NMS (i.e. when soft\_nms\_sigma>0)

# tf.image.pad\_to\_bounding\_box

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

Pad image with zeros to the specified height and width.

### Aliases:

* tf.compat.v1.image.pad\_to\_bounding\_box
* tf.compat.v2.image.pad\_to\_bounding\_box
* tf.image.pad\_to\_bounding\_box

tf.image.pad\_to\_bounding\_box(  
    image,  
    offset\_height,  
    offset\_width,  
    target\_height,  
    target\_width  
)

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

Adds offset\_height rows of zeros on top, offset\_width columns of zeros on the left, and then pads the image on the bottom and right with zeros until it has dimensions target\_height, target\_width.

This op does nothing if offset\_\* is zero and the image already has size target\_height by target\_width.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **offset\_height**: Number of rows of zeros to add on top.
* **offset\_width**: Number of columns of zeros to add on the left.
* **target\_height**: Height of output image.
* **target\_width**: Width of output image.

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, target\_height, target\_width, channels] If image was 3-D, a 3-D float Tensor of shape [target\_height, target\_width, channels]

#### Raises:

* **ValueError**: If the shape of image is incompatible with the offset\_\* or target\_\*arguments, or either offset\_height or offset\_width is negative.

# tf.image.per\_image\_standardization

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

Linearly scales each image in image to have mean 0 and variance 1.

### Aliases:

* tf.compat.v1.image.per\_image\_standardization
* tf.compat.v2.image.per\_image\_standardization
* tf.image.per\_image\_standardization

tf.image.per\_image\_standardization(image)

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

For each 3-D image x in image, computes (x - mean) / adjusted\_stddev, where

* mean is the average of all values in x
* adjusted\_stddev = max(stddev, 1.0/sqrt(N)) is capped away from 0 to protect against division by 0 when handling uniform images
  + N is the number of elements in x
  + stddev is the standard deviation of all values in x

#### Args:

* **image**: An n-D Tensor with at least 3 dimensions, the last 3 of which are the dimensions of each image.

#### Returns:

A Tensor with same shape and dtype as image.

#### Raises:

* **ValueError**: if the shape of 'image' is incompatible with this function.

# tf.image.psnr

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

Returns the Peak Signal-to-Noise Ratio between a and b.

### Aliases:

* tf.compat.v1.image.psnr
* tf.compat.v2.image.psnr
* tf.image.psnr

tf.image.psnr(  
    a,  
    b,  
    max\_val,  
    name=None  
)

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

This is intended to be used on signals (or images). Produces a PSNR value for each image in batch.

The last three dimensions of input are expected to be [height, width, depth].

#### Example:

    # Read images from file.  
    im1 = tf.decode\_png('path/to/im1.png')  
    im2 = tf.decode\_png('path/to/im2.png')  
    # Compute PSNR over tf.uint8 Tensors.  
    psnr1 = tf.image.psnr(im1, im2, max\_val=255)  
  
    # Compute PSNR over tf.float32 Tensors.  
    im1 = tf.image.convert\_image\_dtype(im1, tf.float32)  
    im2 = tf.image.convert\_image\_dtype(im2, tf.float32)  
    psnr2 = tf.image.psnr(im1, im2, max\_val=1.0)  
    # psnr1 and psnr2 both have type tf.float32 and are almost equal.

#### Arguments:

* **a**: First set of images.
* **b**: Second set of images.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **name**: Namespace to embed the computation in.

#### Returns:

The scalar PSNR between a and b. The returned tensor has type [tf.float32](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#float32) and shape [batch\_size, 1].

# tf.image.random\_brightness

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

Adjust the brightness of images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_brightness
* tf.compat.v2.image.random\_brightness
* tf.image.random\_brightness

tf.image.random\_brightness(  
    image,  
    max\_delta,  
    seed=None  
)

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

Equivalent to adjust\_brightness() using a delta randomly picked in the interval [-max\_delta, max\_delta).

#### Args:

* **image**: An image or images to adjust.
* **max\_delta**: float, must be non-negative.
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

The brightness-adjusted image(s).

#### Raises:

* **ValueError**: if max\_delta is negative.

# tf.image.random\_contrast

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

Adjust the contrast of an image or images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_contrast
* tf.compat.v2.image.random\_contrast
* tf.image.random\_contrast

tf.image.random\_contrast(  
    image,  
    lower,  
    upper,  
    seed=None  
)

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

Equivalent to adjust\_contrast() but uses a contrast\_factor randomly picked in the interval [lower, upper].

#### Args:

* **image**: An image tensor with 3 or more dimensions.
* **lower**: float. Lower bound for the random contrast factor.
* **upper**: float. Upper bound for the random contrast factor.
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

The contrast-adjusted image(s).

#### Raises:

* **ValueError**: if upper <= lower or if lower < 0.

# tf.image.random\_crop

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

Randomly crops a tensor to a given size.

### Aliases:

* tf.compat.v1.image.random\_crop
* tf.compat.v1.random\_crop
* tf.compat.v2.image.random\_crop
* tf.image.random\_crop

tf.image.random\_crop(  
    value,  
    size,  
    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:

* [Pix2Pix](https://www.tensorflow.org/beta/tutorials/generative/pix2pix)

Slices a shape size portion out of value at a uniformly chosen offset. Requires value.shape >= size.

If a dimension should not be cropped, pass the full size of that dimension. For example, RGB images can be cropped with size = [crop\_height, crop\_width, 3].

#### Args:

* **value**: Input tensor to crop.
* **size**: 1-D tensor with size the rank of value.
* **seed**: Python integer. Used to create a random seed. 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 cropped tensor of the same rank as value and shape size.

# tf.image.random\_flip\_left\_right

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

Randomly flip an image horizontally (left to right).

### Aliases:

* tf.compat.v1.image.random\_flip\_left\_right
* tf.compat.v2.image.random\_flip\_left\_right
* tf.image.random\_flip\_left\_right

tf.image.random\_flip\_left\_right(  
    image,  
    seed=None  
)

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

With a 1 in 2 chance, outputs the contents of image flipped along the second dimension, which is width. Otherwise output the image as-is.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.random\_flip\_up\_down

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

Randomly flips an image vertically (upside down).

### Aliases:

* tf.compat.v1.image.random\_flip\_up\_down
* tf.compat.v2.image.random\_flip\_up\_down
* tf.image.random\_flip\_up\_down

tf.image.random\_flip\_up\_down(  
    image,  
    seed=None  
)

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

With a 1 in 2 chance, outputs the contents of image flipped along the first dimension, which is height. Otherwise output the image as-is.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **seed**: A Python integer. Used to create a random seed. See [tf.compat.v1.set\_random\_seed](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/compat/v1/set_random_seed)for behavior.

#### Returns:

A tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.random\_hue

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

Adjust the hue of RGB images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_hue
* tf.compat.v2.image.random\_hue
* tf.image.random\_hue

tf.image.random\_hue(  
    image,  
    max\_delta,  
    seed=None  
)

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

Equivalent to adjust\_hue() but uses a delta randomly picked in the interval [-max\_delta, max\_delta].

max\_delta must be in the interval [0, 0.5].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **max\_delta**: float. Maximum value for the random delta.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if max\_delta is invalid.

# tf.image.random\_jpeg\_quality

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

Randomly changes jpeg encoding quality for inducing jpeg noise.

### Aliases:

* tf.compat.v1.image.random\_jpeg\_quality
* tf.compat.v2.image.random\_jpeg\_quality
* tf.image.random\_jpeg\_quality

tf.image.random\_jpeg\_quality(  
    image,  
    min\_jpeg\_quality,  
    max\_jpeg\_quality,  
    seed=None  
)

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

min\_jpeg\_quality must be in the interval [0, 100] and less than max\_jpeg\_quality.max\_jpeg\_quality must be in the interval [0, 100].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **min\_jpeg\_quality**: Minimum jpeg encoding quality to use.
* **max\_jpeg\_quality**: Maximum jpeg encoding quality to use.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if min\_jpeg\_quality or max\_jpeg\_quality is invalid.

# tf.image.random\_saturation

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

Adjust the saturation of RGB images by a random factor.

### Aliases:

* tf.compat.v1.image.random\_saturation
* tf.compat.v2.image.random\_saturation
* tf.image.random\_saturation

tf.image.random\_saturation(  
    image,  
    lower,  
    upper,  
    seed=None  
)

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

Equivalent to adjust\_saturation() but uses a saturation\_factor randomly picked in the interval [lower, upper].

#### Args:

* **image**: RGB image or images. Size of the last dimension must be 3.
* **lower**: float. Lower bound for the random saturation factor.
* **upper**: float. Upper bound for the random saturation factor.
* **seed**: An operation-specific seed. It will be used in conjunction with the graph-level seed to determine the real seeds that will be used in this operation. Please see the documentation of set\_random\_seed for its interaction with the graph-level random seed.

#### Returns:

Adjusted image(s), same shape and DType as image.

#### Raises:

* **ValueError**: if upper <= lower or if lower < 0.

# tf.image.resize

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

Resize images to size using the specified method.

### Aliases:

* tf.compat.v2.image.resize
* tf.image.resize

tf.image.resize(  
    images,  
    size,  
    method=ResizeMethod.BILINEAR,  
    preserve\_aspect\_ratio=False,  
    antialias=False,  
    name=None  
)

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

### Used in the tutorials:

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

Resized images will be distorted if their original aspect ratio is not the same as size. To avoid distortions see [tf.image.resize\_with\_pad](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/resize_with_pad).

When 'antialias' is true, the sampling filter will anti-alias the input image as well as interpolate. When downsampling an image with [anti-aliasing](https://en.wikipedia.org/wiki/Spatial_anti-aliasing) the sampling filter kernel is scaled in order to properly anti-alias the input image signal. 'antialias' has no effect when upsampling an image.

* **bilinear**: [Bilinear interpolation.](https://en.wikipedia.org/wiki/Bilinear_interpolation) If 'antialias' is true, becomes a hat/tent filter function with radius 1 when downsampling.
* **lanczos3**: [Lanczos kernel](https://en.wikipedia.org/wiki/Lanczos_resampling) with radius 3. High-quality practical filter but may have some ringing especially on synthetic images.
* **lanczos5**: [Lanczos kernel](https://en.wikipedia.org/wiki/Lanczos_resampling) with radius 5. Very-high-quality filter but may have stronger ringing.
* **bicubic**: [Cubic interpolant](https://en.wikipedia.org/wiki/Bicubic_interpolation) of Keys. Equivalent to Catmull-Rom kernel. Reasonably good quality and faster than Lanczos3Kernel, particularly when upsampling.
* **gaussian**: [Gaussian kernel](https://en.wikipedia.org/wiki/Gaussian_filter) with radius 3, sigma = 1.5 / 3.0.
* **nearest**: [Nearest neighbor interpolation.](https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation) 'antialias' has no effect when used with nearest neighbor interpolation.
* **area**: Anti-aliased resampling with area interpolation. 'antialias' has no effect when used with area interpolation; it always anti-aliases.
* **mitchellcubic**: Mitchell-Netravali Cubic non-interpolating filter. For synthetic images (especially those lacking proper prefiltering), less ringing than Keys cubic kernel but less sharp.

Note that near image edges the filtering kernel may be partially outside the image boundaries. For these pixels, only input pixels inside the image will be included in the filter sum, and the output value will be appropriately normalized.

The return value has the same type as images if method is ResizeMethod.NEAREST\_NEIGHBOR. Otherwise, the return value has type float32.

#### Args:

* **images**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **size**: A 1-D int32 Tensor of 2 elements: new\_height, new\_width. The new size for the images.
* **method**: ResizeMethod. Defaults to bilinear.
* **preserve\_aspect\_ratio**: Whether to preserve the aspect ratio. If this is set, then images will be resized to a size that fits in size while preserving the aspect ratio of the original image. Scales up the image if size is bigger than the current size of the image. Defaults to False.
* **antialias**: Whether to use an anti-aliasing filter when downsampling an image.
* **name**: A name for this operation (optional).

#### Raises:

* **ValueError**: if the shape of images is incompatible with the shape arguments to this function
* **ValueError**: if size has invalid shape or type.
* **ValueError**: if an unsupported resize method is specified.

#### Returns:

If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.image.ResizeMethod

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod#top_of_page)
* [Class ResizeMethod](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod#class_resizemethod)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/ResizeMethod#class_members)

## Class ResizeMethod

### Aliases:

* Class tf.compat.v2.image.ResizeMethod
* Class tf.image.ResizeMethod

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

## Class Members

* AREA = 'area'
* BICUBIC = 'bicubic'
* BILINEAR = 'bilinear'
* GAUSSIAN = 'gaussian'
* LANCZOS3 = 'lanczos3'
* LANCZOS5 = 'lanczos5'
* MITCHELLCUBIC = 'mitchellcubic'
* NEAREST\_NEIGHBOR = 'nearest'

# tf.image.resize\_with\_crop\_or\_pad

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

Crops and/or pads an image to a target width and height.

### Aliases:

* tf.compat.v1.image.resize\_image\_with\_crop\_or\_pad
* tf.compat.v1.image.resize\_with\_crop\_or\_pad
* tf.compat.v2.image.resize\_with\_crop\_or\_pad
* tf.image.resize\_with\_crop\_or\_pad

tf.image.resize\_with\_crop\_or\_pad(  
    image,  
    target\_height,  
    target\_width  
)

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

Resizes an image to a target width and height by either centrally cropping the image or padding it evenly with zeros.

If width or height is greater than the specified target\_width or target\_height respectively, this op centrally crops along that dimension. If width or height is smaller than the specified target\_width or target\_height respectively, this op centrally pads with 0 along that dimension.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **target\_height**: Target height.
* **target\_width**: Target width.

#### Raises:

* **ValueError**: if target\_height or target\_width are zero or negative.

#### Returns:

Cropped and/or padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape[new\_height, new\_width, channels].

# tf.image.resize\_with\_pad

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

Resizes and pads an image to a target width and height.

### Aliases:

* tf.compat.v2.image.resize\_with\_pad
* tf.image.resize\_with\_pad

tf.image.resize\_with\_pad(  
    image,  
    target\_height,  
    target\_width,  
    method=ResizeMethod.BILINEAR,  
    antialias=False  
)

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

Resizes an image to a target width and height by keeping the aspect ratio the same without distortion. If the target dimensions don't match the image dimensions, the image is resized and then padded with zeroes to match requested dimensions.

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **target\_height**: Target height.
* **target\_width**: Target width.
* **method**: Method to use for resizing image. See image.resize()
* **antialias**: Whether to use anti-aliasing when resizing. See 'image.resize()'.

#### Raises:

* **ValueError**: if target\_height or target\_width are zero or negative.

#### Returns:

Resized and padded image. If images was 4-D, a 4-D float Tensor of shape [batch, new\_height, new\_width, channels]. If images was 3-D, a 3-D float Tensor of shape [new\_height, new\_width, channels].

# tf.image.rgb\_to\_grayscale

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

Converts one or more images from RGB to Grayscale.

### Aliases:

* tf.compat.v1.image.rgb\_to\_grayscale
* tf.compat.v2.image.rgb\_to\_grayscale
* tf.image.rgb\_to\_grayscale

tf.image.rgb\_to\_grayscale(  
    images,  
    name=None  
)

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

Outputs a tensor of the same DType and rank as images. The size of the last dimension of the output is 1, containing the Grayscale value of the pixels.

#### Args:

* **images**: The RGB tensor to convert. Last dimension must have size 3 and should contain RGB values.
* **name**: A name for the operation (optional).

#### Returns:

The converted grayscale image(s).

# tf.image.rgb\_to\_hsv

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

Converts one or more images from RGB to HSV.

### Aliases:

* tf.compat.v1.image.rgb\_to\_hsv
* tf.compat.v2.image.rgb\_to\_hsv
* tf.image.rgb\_to\_hsv

tf.image.rgb\_to\_hsv(  
    images,  
    name=None  
)

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

Outputs a tensor of the same shape as the images tensor, containing the HSV value of the pixels. The output is only well defined if the value in images are in [0,1].

output[..., 0] contains hue, output[..., 1] contains saturation, and output[..., 2]contains value. All HSV values are in [0,1]. A hue of 0 corresponds to pure red, hue 1/3 is pure green, and 2/3 is pure blue.

#### Args:

* **images**: A Tensor. Must be one of the following types: half, bfloat16, float32, float64. 1-D or higher rank. RGB data to convert. Last dimension must be size 3.
* **name**: A name for the operation (optional).

#### Returns:

A Tensor. Has the same type as images.

# tf.image.rgb\_to\_yiq

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

Converts one or more images from RGB to YIQ.

### Aliases:

* tf.compat.v1.image.rgb\_to\_yiq
* tf.compat.v2.image.rgb\_to\_yiq
* tf.image.rgb\_to\_yiq

tf.image.rgb\_to\_yiq(images)

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

Outputs a tensor of the same shape as the images tensor, containing the YIQ value of the pixels. The output is only well defined if the value in images are in [0,1].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.rgb\_to\_yuv

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

Converts one or more images from RGB to YUV.

### Aliases:

* tf.compat.v1.image.rgb\_to\_yuv
* tf.compat.v2.image.rgb\_to\_yuv
* tf.image.rgb\_to\_yuv

tf.image.rgb\_to\_yuv(images)

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

Outputs a tensor of the same shape as the images tensor, containing the YUV value of the pixels. The output is only well defined if the value in images are in [0,1].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.rot90

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90#top_of_page)
* [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/rot90#aliases)

Rotate image(s) counter-clockwise by 90 degrees.

### Aliases:

* tf.compat.v1.image.rot90
* tf.compat.v2.image.rot90
* tf.image.rot90

tf.image.rot90(  
    image,  
    k=1,  
    name=None  
)

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

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **k**: A scalar integer. The number of times the image is rotated by 90 degrees.
* **name**: A name for this operation (optional).

#### Returns:

A rotated tensor of the same type and shape as image.

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.sample\_distorted\_bounding\_box

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

Generate a single randomly distorted bounding box for an image.

### Aliases:

* tf.compat.v2.image.sample\_distorted\_bounding\_box
* tf.image.sample\_distorted\_bounding\_box

tf.image.sample\_distorted\_bounding\_box(  
    image\_size,  
    bounding\_boxes,  
    seed=0,  
    min\_object\_covered=0.1,  
    aspect\_ratio\_range=None,  
    area\_range=None,  
    max\_attempts=None,  
    use\_image\_if\_no\_bounding\_boxes=None,  
    name=None  
)

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

Bounding box annotations are often supplied in addition to ground-truth labels in image recognition or object localization tasks. A common technique for training such a system is to randomly distort an image while preserving its content, i.e. data augmentation. This Op outputs a randomly distorted localization of an object, i.e. bounding box, given an image\_size, bounding\_boxes and a series of constraints.

The output of this Op is a single bounding box that may be used to crop the original image. The output is returned as 3 tensors: begin, size and bboxes. The first 2 tensors can be fed directly into [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice) to crop the image. The latter may be supplied to [tf.image.draw\_bounding\_boxes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes) to visualize what the bounding box looks like.

Bounding boxes are supplied and returned as [y\_min, x\_min, y\_max, x\_max]. The bounding box coordinates are floats in [0.0, 1.0] relative to the width and height of the underlying image.

For example,

    # Generate a single distorted bounding box.  
    begin, size, bbox\_for\_draw = tf.image.sample\_distorted\_bounding\_box(  
        tf.shape(image),  
        bounding\_boxes=bounding\_boxes,  
        min\_object\_covered=0.1)  
  
    # Draw the bounding box in an image summary.  
    image\_with\_box = tf.image.draw\_bounding\_boxes(tf.expand\_dims(image, 0),  
                                                  bbox\_for\_draw)  
    tf.compat.v1.summary.image('images\_with\_box', image\_with\_box)  
  
    # Employ the bounding box to distort the image.  
    distorted\_image = tf.slice(image, begin, size)

Note that if no bounding box information is available, setting use\_image\_if\_no\_bounding\_boxes = true will assume there is a single implicit bounding box covering the whole image. If use\_image\_if\_no\_bounding\_boxes is false and no bounding boxes are supplied, an error is raised.

#### Args:

* **image\_size**: A Tensor. Must be one of the following types: uint8, int8, int16, int32, int64. 1-D, containing [height, width, channels].
* **bounding\_boxes**: A Tensor of type float32. 3-D with shape [batch, N, 4] describing the N bounding boxes associated with the image.
* **seed**: An optional int. Defaults to 0. If seed is set to non-zero, the random number generator is seeded by the given seed. Otherwise, it is seeded by a random seed.
* **min\_object\_covered**: A Tensor of type float32. Defaults to 0.1. The cropped area of the image must contain at least this fraction of any bounding box supplied. The value of this parameter should be non-negative. In the case of 0, the cropped area does not need to overlap any of the bounding boxes supplied.
* **aspect\_ratio\_range**: An optional list of floats. Defaults to [0.75, 1.33]. The cropped area of the image must have an aspect ratio = width / height within this range.
* **area\_range**: An optional list of floats. Defaults to [0.05, 1]. The cropped area of the image must contain a fraction of the supplied image within this range.
* **max\_attempts**: An optional int. Defaults to 100. Number of attempts at generating a cropped region of the image of the specified constraints. After max\_attempts failures, return the entire image.
* **use\_image\_if\_no\_bounding\_boxes**: An optional bool. Defaults to False. Controls behavior if no bounding boxes supplied. If true, assume an implicit bounding box covering the whole input. If false, raise an error.
* **name**: A name for the operation (optional).

#### Returns:

A tuple of Tensor objects (begin, size, bboxes).

* **begin**: A Tensor. Has the same type as image\_size. 1-D, containing [offset\_height, offset\_width, 0]. Provide as input to [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice).
* **size**: A Tensor. Has the same type as image\_size. 1-D, containing [target\_height, target\_width, -1]. Provide as input to [tf.slice](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/slice).
* **bboxes**: A Tensor of type float32. 3-D with shape [1, 1, 4] containing the distorted bounding box. Provide as input to [tf.image.draw\_bounding\_boxes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/image/draw_bounding_boxes).

# tf.image.sobel\_edges

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

Returns a tensor holding Sobel edge maps.

### Aliases:

* tf.compat.v1.image.sobel\_edges
* tf.compat.v2.image.sobel\_edges
* tf.image.sobel\_edges

tf.image.sobel\_edges(image)

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

### Used in the tutorials:

* [Neural style transfer](https://www.tensorflow.org/beta/tutorials/generative/style_transfer)

#### Arguments:

* **image**: Image tensor with shape [batch\_size, h, w, d] and type float32 or float64. The image(s) must be 2x2 or larger.

#### Returns:

Tensor holding edge maps for each channel. Returns a tensor with shape [batch\_size, h, w, d, 2] where the last two dimensions hold [[dy[0], dx[0]], [dy[1], dx[1]], ..., [dy[d-1], dx[d-1]]] calculated using the Sobel filter.

# tf.image.ssim

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

Computes SSIM index between img1 and img2.

### Aliases:

* tf.compat.v1.image.ssim
* tf.compat.v2.image.ssim
* tf.image.ssim

tf.image.ssim(  
    img1,  
    img2,  
    max\_val,  
    filter\_size=11,  
    filter\_sigma=1.5,  
    k1=0.01,  
    k2=0.03  
)

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

This function is based on the standard SSIM implementation from: Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing.

**Note:** The true SSIM is only defined on grayscale. This function does not perform any colorspace transform. (If input is already YUV, then it will compute YUV SSIM average.)

#### Details:

* 11x11 Gaussian filter of width 1.5 is used.
* k1 = 0.01, k2 = 0.03 as in the original paper.

The image sizes must be at least 11x11 because of the filter size.

#### Example:

    # Read images from file.  
    im1 = tf.decode\_png('path/to/im1.png')  
    im2 = tf.decode\_png('path/to/im2.png')  
    # Compute SSIM over tf.uint8 Tensors.  
    ssim1 = tf.image.ssim(im1, im2, max\_val=255, filter\_size=11,  
                          filter\_sigma=1.5, k1=0.01, k2=0.03)  
  
    # Compute SSIM over tf.float32 Tensors.  
    im1 = tf.image.convert\_image\_dtype(im1, tf.float32)  
    im2 = tf.image.convert\_image\_dtype(im2, tf.float32)  
    ssim2 = tf.image.ssim(im1, im2, max\_val=1.0, filter\_size=11,  
                          filter\_sigma=1.5, k1=0.01, k2=0.03)  
    # ssim1 and ssim2 both have type tf.float32 and are almost equal.

#### Args:

* **img1**: First image batch.
* **img2**: Second image batch.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **filter\_size**: Default value 11 (size of gaussian filter).
* **filter\_sigma**: Default value 1.5 (width of gaussian filter).
* **k1**: Default value 0.01
* **k2**: Default value 0.03 (SSIM is less sensitivity to K2 for lower values, so it would be better if we taken the values in range of 0< K2 <0.4).

#### Returns:

A tensor containing an SSIM value for each image in batch. Returned SSIM values are in range (-1, 1], when pixel values are non-negative. Returns a tensor with shape: broadcast(img1.shape[:-3], img2.shape[:-3]).

# tf.image.ssim\_multiscale

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

Computes the MS-SSIM between img1 and img2.

### Aliases:

* tf.compat.v1.image.ssim\_multiscale
* tf.compat.v2.image.ssim\_multiscale
* tf.image.ssim\_multiscale

tf.image.ssim\_multiscale(  
    img1,  
    img2,  
    max\_val,  
    power\_factors=\_MSSSIM\_WEIGHTS,  
    filter\_size=11,  
    filter\_sigma=1.5,  
    k1=0.01,  
    k2=0.03  
)

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

This function assumes that img1 and img2 are image batches, i.e. the last three dimensions are [height, width, channels].

**Note:** The true SSIM is only defined on grayscale. This function does not perform any colorspace transform. (If input is already YUV, then it will compute YUV SSIM average.)

Original paper: Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik. "Multiscale structural similarity for image quality assessment." Signals, Systems and Computers, 2004.

#### Arguments:

* **img1**: First image batch.
* **img2**: Second image batch. Must have the same rank as img1.
* **max\_val**: The dynamic range of the images (i.e., the difference between the maximum the and minimum allowed values).
* **power\_factors**: Iterable of weights for each of the scales. The number of scales used is the length of the list. Index 0 is the unscaled resolution's weight and each increasing scale corresponds to the image being downsampled by 2. Defaults to (0.0448, 0.2856, 0.3001, 0.2363, 0.1333), which are the values obtained in the original paper.
* **filter\_size**: Default value 11 (size of gaussian filter).
* **filter\_sigma**: Default value 1.5 (width of gaussian filter).
* **k1**: Default value 0.01
* **k2**: Default value 0.03 (SSIM is less sensitivity to K2 for lower values, so it would be better if we taken the values in range of 0< K2 <0.4).

#### Returns:

A tensor containing an MS-SSIM value for each image in batch. The values are in range [0, 1]. Returns a tensor with shape: broadcast(img1.shape[:-3], img2.shape[:-3]).

# tf.image.total\_variation

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

Calculate and return the total variation for one or more images.

### Aliases:

* tf.compat.v1.image.total\_variation
* tf.compat.v2.image.total\_variation
* tf.image.total\_variation

tf.image.total\_variation(  
    images,  
    name=None  
)

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

The total variation is the sum of the absolute differences for neighboring pixel-values in the input images. This measures how much noise is in the images.

This can be used as a loss-function during optimization so as to suppress noise in images. If you have a batch of images, then you should calculate the scalar loss-value as the sum: loss = tf.reduce\_sum(tf.image.total\_variation(images))

This implements the anisotropic 2-D version of the formula described here:

https://en.wikipedia.org/wiki/Total\_variation\_denoising

#### Args:

* **images**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: if images.shape is not a 3-D or 4-D vector.

#### Returns:

The total variation of images.

If images was 4-D, return a 1-D float Tensor of shape [batch] with the total variation for each image in the batch. If images was 3-D, return a scalar float with the total variation for that image.

# tf.image.transpose

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

Transpose image(s) by swapping the height and width dimension.

### Aliases:

* tf.compat.v1.image.transpose
* tf.compat.v1.image.transpose\_image
* tf.compat.v2.image.transpose
* tf.image.transpose

tf.image.transpose(  
    image,  
    name=None  
)

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

#### Args:

* **image**: 4-D Tensor of shape [batch, height, width, channels] or 3-D Tensor of shape [height, width, channels].
* **name**: A name for this operation (optional).

#### Returns:

If image was 4-D, a 4-D float Tensor of shape [batch, width, height, channels] If image was 3-D, a 3-D float Tensor of shape [width, height, channels]

#### Raises:

* **ValueError**: if the shape of image not supported.

# tf.image.yiq\_to\_rgb

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

Converts one or more images from YIQ to RGB.

### Aliases:

* tf.compat.v1.image.yiq\_to\_rgb
* tf.compat.v2.image.yiq\_to\_rgb
* tf.image.yiq\_to\_rgb

tf.image.yiq\_to\_rgb(images)

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

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the Y value in images are in [0,1], I value are in [-0.5957,0.5957] and Q value are in [-0.5226,0.5226].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.

# tf.image.yuv\_to\_rgb

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

Converts one or more images from YUV to RGB.

### Aliases:

* tf.compat.v1.image.yuv\_to\_rgb
* tf.compat.v2.image.yuv\_to\_rgb
* tf.image.yuv\_to\_rgb

tf.image.yuv\_to\_rgb(images)

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

Outputs a tensor of the same shape as the images tensor, containing the RGB value of the pixels. The output is only well defined if the Y value in images are in [0,1], U and V value are in [-0.5,0.5].

#### Args:

* **images**: 2-D or higher rank. Image data to convert. Last dimension must be size 3.

#### Returns:

* **images**: tensor with the same shape as images.