See playlist (https://www.youtube.com/playlist?list=PLQY2H8rRoyvxtOJfAzfs0Cq5FqBuYO8e8)

tf.image.resize

View source on GitHub (https://github.com/tensorflow/tensorflow/blob/v2.5.0/tensorflow/python/ops/image_ops_impl.py#L1547-L1723)

Resize `images` to `size` using the specified `method`.

```
tf.image.resize(
    images, size, method=ResizeMethod.BILINEAR, preserve_aspect_ratio=False,
    antialias=False, name=None
)
```

Used in the notebooks

Used in the guide	Used in the tutorials
• Transfer learning and fine-tuning (https://www.tensorflow.org/guide/keras/transfer_learning) • tf.data: Build TensorFlow input pipelines (https://www.tensorflow.org/guide/data) • Estimators (https://www.tensorflow.org/guide/estimator)	• DeepDream (https://www.tensorflow.org/tutorials/generative/deepdream • Image segmentation (https://www.tensorflow.org/tutorials/images/segmentation) • Pix2Pix (https://www.tensorflow.org/tutorials/generative/pix2 • Neural style transfer (https://www.tensorflow.org/tutorials/generative/style_trans • Adversarial example using FGSM (https://www.tensorflow.org/tutorials/generative/adversarial

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/api_docs/python/tf/image/resize_with_pad).

```
>>> image = tf.constant([
...  [1,0,0,0,0],
...  [0,1,0,0,0],
...  [0,0,1,0,0],
...  [0,0,0,1,0],
...  [0,0,0,0,1],
... ])
>>> # Add "batch" and "channels" dimensions
>>> image = image[tf.newaxis, ..., tf.newaxis]
>>> image.shape.as_list()  # [batch, height, width, channels]
```

```
[1, 5, 5, 1]
>>> tf.image.resize(image, [3,5])[0,...,0].numpy()
array([[0.6666667, 0.3333333, 0.       , 0.       , 0.       ],
       [0.       , 0.       , 1.       , 0.       , 0.       ],
       [0.       , 0.       , 0.       , 0.3333335, 0.6666665]],
      dtype=float32)
```

It works equally well with a single image instead of a batch of images:

```
>>> tf.image.resize(image[0], [3,5]).shape.as_list()
[3, 5, 1]
```

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:

```
>>> a = tf.image.resize(image, [5,10])
>>> b = tf.image.resize(image, [5,10], antialias=True)
>>> tf.reduce_max(abs(a - b)).numpy()
0.0
```

The **`method`** argument expects an item from the **`image.ResizeMethod`** (https://www.tensorflow.org/api_docs/python/tf/image/ResizeMethod) enum, or the string equivalent. The options are:

- **`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.

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

The return value has type **float32**, unless the **method** is **ResizeMethod.NEAREST_NEIGHBOR** (https://www.tensorflow.org/api_docs/python/tf/image/ResizeMethod#NEAREST_NEIGHBOR), then the return dtype is the dtype of **images**:

```
>>> nn = tf.image.resize(image, [5,7], method='nearest')
>>> nn[0,...,0].numpy()
array([[1, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 1, 0],
       [0, 0, 0, 0, 0, 0, 1]], dtype=int32)
```

With **preserve_aspect_ratio=True**, the aspect ratio is preserved, so **size** is the maximum for each dimension:

```
>>> max_10_20 = tf.image.resize(image, [10,20], preserve_aspect_ratio=True)
>>> max_10_20.shape.as_list()
[1, 10, 10, 1]
```

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	An **image.ResizeMethod** (https://www.tensorflow.org/api_docs/python/tf/image/ResizeMethod), or string equivalent. 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 an 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]`.

Last updated 2021-05-14 UTC.