tf.image.sample_distorted_bounding_box

```
sample_distorted_bounding_box(
    image_size,
    bounding_boxes,
    seed=None,
    seed2=None,
    min_object_covered=None,
    aspect_ratio_range=None,
    area_range=None,
    max_attempts=None,
    use_image_if_no_bounding_boxes=None,
    name=None
)
```

Defined in `tensorflow/python/ops/image_ops_impl.py`.

See the guide: Images > Working with Bounding Boxes

Generate a single randomly distorted bounding box for an image.

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` to crop the image. The latter may be supplied to `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)

    # 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.image_summary('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 either `seed` or `seed2` are set to non-zero, the random number generator is seeded by the given `seed`. Otherwise, it is seeded by a random seed.
- `seed2`: An optional `int`. Defaults to `0`. A second seed to avoid seed collision.
- `min_object_covered`: An optional `float`. 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 in 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`.
- `size`: A `Tensor`. Has the same type as `image_size`. 1-D, containing `[target_height, target_width, -1]`. Provide as input to `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`.