TensorFlow

TensorFlow   API r1.4

tf.strided_slice

```
strided_slice(
    input_,
    begin,
    end,
    strides=None,
    begin_mask=0,
    end_mask=0,
    ellipsis_mask=0,
    new_axis_mask=0,
    shrink_axis_mask=0,
    var=None,
    name=None
)
```

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

See the guide: Tensor Transformations > Slicing and Joining

Extracts a strided slice of a tensor (generalized python array indexing).

Most users will want to use `tf.Tensor.__getitem__` and `tf.Variable.__getitem__`. That allows NumPy style slicing syntax (i.e. `tensor[..., 3:4:-1, tf.newaxis, 3]`). This op is the low-level interface that are used to implement operators. Those interfaces are much more friendly, and highly recommended.

To a first order, this operation extracts a slice of size `end - begin` from a tensor `input` starting at the location specified by `begin`. The slice continues by adding `stride` to the `begin` index until all dimensions are not less than `end`. Note that components of stride can be negative, which causes a reverse slice.

This operation can be thought of an encoding of a numpy style sliced range. Given a python slice input[, , ...,] this function will be called as follows.

`begin`, `end`, and `strides` will be all length n. n is in general not the same dimensionality as `input`.

For the ith spec, `begin_mask`, `end_mask`, `ellipsis_mask`, `new_axis_mask`, and `shrink_axis_mask` will have the ith bit corresponding to the ith spec.

If the ith bit of `begin_mask` is non-zero, `begin[i]` is ignored and the fullest possible range in that dimension is used instead. `end_mask` works analogously, except with the end range.

`foo[5:,:,:3]` on a 7x8x9 tensor is equivalent to `foo[5:7,0:8,0:3]`. `foo[::-1]` reverses a tensor with shape 8.

If the ith bit of `ellipsis_mask` is non-zero, as many unspecified dimensions as needed will be inserted between other dimensions. Only one non-zero bit is allowed in `ellipsis_mask`.

For example `foo[3:5,...,4:5]` on a shape 10x3x3x10 tensor is equivalent to `foo[3:5,:,:,4:5]` and `foo[3:5,...]` is equivalent to `foo[3:5,:,:,:]`.

If the ith bit of `new_axis_mask` is one, then `begin`, `end`, and `stride` are ignored and a new length 1 dimension is added at this point in the output tensor.

For example `foo[3:5,4]` on a 10x8 tensor produces a shape 2 tensor whereas `foo[3:5,4:5]` produces a shape 2x1 tensor with shrink_mask being 1<<1 == 2.

If the ith bit of `shrink_axis_mask` is one, then `begin`, `end[i]`, and `stride[i]` are used to do a slice in the appropriate dimension, but the output tensor will be reduced in dimensionality by one. This is only valid if the ith entry of slice[i]==1.

NOTE: `begin` and `end` are zero-indexed `.` strides` entries must be non-zero.

```
t = tf.constant([[[1, 1, 1], [2, 2, 2]],
                 [[3, 3, 3], [4, 4, 4]],
                 [[5, 5, 5], [6, 6, 6]]])
tf.strided_slice(t, [1, 0, 0], [2, 1, 3], [1, 1, 1])  # [[[3, 3, 3]]]
tf.strided_slice(t, [1, 0, 0], [2, 2, 3], [1, 1, 1])  # [[[3, 3, 3],
                                                      #   [4, 4, 4]]]
tf.strided_slice(t, [1, -1, 0], [2, -3, 3], [1, -1, 1])  # [[[4, 4, 4],
                                                         #   [3, 3, 3]]]
```

Args:

- `input_`: A `Tensor`.
- `begin`: An `int32` or `int64` `Tensor`.
- `end`: An `int32` or `int64` `Tensor`.
- `strides`: An `int32` or `int64` `Tensor`.
- `begin_mask`: An `int32` mask.
- `end_mask`: An `int32` mask.
- `ellipsis_mask`: An `int32` mask.
- `new_axis_mask`: An `int32` mask.
- `shrink_axis_mask`: An `int32` mask.
- `var`: The variable corresponding to `input_` or None
- `name`: A name for the operation (optional).

Returns:

A `Tensor` the same type as `input`.

Last updated November 2, 2017.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow