PSAMPLE

CLASS `torch.nn.Upsample(size: Optional[Union[T, Tuple[T, ...]]] = None, scale_factor: Optional[Union[T, Tuple[T, ...]]] = None, mode: str = 'nearest', align_corners: Optional[bool] = None)` [SO RCE]

psamples a given multi-channel 1D (temporal), 2D (spatial) or 3D (volumetric) ata.

The input ata is assume to e of the form *minibatch x channels x* [*optional depth*] *x* [*optional height*] *x width*. Hence, for spatial inputs, we expect a 4D Tensor an for volumetric inputs, we expect a 5D Tensor.

The algorithms availa le for upsampling are nearest neigh or an linear, ilinear, icu ic an trilinear for 3D, 4D an 5D input Tensor, respectively.

One can either give a `scale_factor` or the target output `size` to calculate the output size. (You cannot give oth, as it is am iguous)

Parameters

- **size** (*int or Tuple*[*int*] *or Tuple*[*int, int*] *or Tuple*[*int, int, int*], *optional*) – output spatial sizes
- **scale_factor** (*float or Tuple*[*float*] *or Tuple*[*float, float*] *or Tuple*[*float, float, float*], *optional*) – multiplier for spatial size. Has to match input size if it is a tuple.
- **mo e** (*str*, *optional*) – the upsampling algorithm: one of `'nearest'`, `'linear'`, `'bilinear'`, `'bicubic'` an `'trilinear'`. Default: `'nearest'`
- **align_corners** (*bool*, *optional*) – if `True`, the corner pixels of the input an output tensors are aligne , an thus preserving the values at those pixels. This only has effect when `mode` is `'linear'`, `'bilinear'`, or `'trilinear'`. Default: `False`

Shape:

- Input: (N, C, W_{in}), (N, C, H_{in}, W_{in}) or $(N, C, D_{in}, H_{in}, W_{in})$
- Output: (N, C, W_{out}), (N, C, H_{out}, W_{out}) or $(N, C, D_{out}, H_{out}, W_{out})$, where

$$D_{out} = \left\lfloor D_{in} \times \text{scale\_factor} \right\rfloor$$

$$H_{out} = \left\lfloor H_{in} \times \text{scale\_factor} \right\rfloor$$

$$W_{out} = \left\lfloor W_{in} \times \text{scale\_factor} \right\rfloor$$

• WARNING

With `align_corners = True`, the linearly interpolating mo es (*linear*, *bilinear*, *bicubic*, an *trilinear*) on't proportionally align the output an input pixels, an thus the output values can epen on the input size. This was the efault ehavior for these mo es up to version 0.3.1. Since then, the efault ehavior is `align_corners = False`. See elow for concrete examples on how this affects the outputs.

• NOTE

If you want ownsampling/general resizing, you shoul use `interpolate()`.

Examples:

```
>>> input = torch.arange(1, 5, dtype=torch.float32).view(1, 1, 2, 2)
>>> input
tensor([[[[ 1.,  2.],
          [ 3.,  4.]]]])

>>> m = nn.Upsample(scale_factor=2, mode='nearest')
>>> m(input)
tensor([[[[ 1.,  1.,  2.,  2.],
          [ 1.,  1.,  2.,  2.],
          [ 3.,  3.,  4.,  4.],
          [ 3.,  3.,  4.,  4.]]]])

>>> m = nn.Upsample(scale_factor=2, mode='bilinear')  # align_corners=False
>>> m(input)
tensor([[[[ 1.0000,  1.2500,  1.7500,  2.0000],
          [ 1.5000,  1.7500,  2.2500,  2.5000],
          [ 2.5000,  2.7500,  3.2500,  3.5000],
          [ 3.0000,  3.2500,  3.7500,  4.0000]]]])

>>> m = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
>>> m(input)
tensor([[[[ 1.0000,  1.3333,  1.6667,  2.0000],
          [ 1.6667,  2.0000,  2.3333,  2.6667],
          [ 2.3333,  2.6667,  3.0000,  3.3333],
          [ 3.0000,  3.3333,  3.6667,  4.0000]]]])

>>> # Try scaling the same data in a larger tensor
>>>
>>> input_3x3 = torch.zeros(3, 3).view(1, 1, 3, 3)
>>> input_3x3[:, :, :2, :2].copy_(input)
tensor([[[[ 1.,  2.],
          [ 3.,  4.]]]])
>>> input_3x3
tensor([[[[ 1.,  2.,  0.],
          [ 3.,  4.,  0.],
          [ 0.,  0.,  0.]]]])

>>> m = nn.Upsample(scale_factor=2, mode='bilinear')  # align_corners=False
>>> # Notice that values in top left corner are the same with the small input (except at boundary)
>>> m(input_3x3)
tensor([[[[ 1.0000,  1.2500,  1.7500,  1.5000,  0.5000,  0.0000],
          [ 1.5000,  1.7500,  2.2500,  1.8750,  0.6250,  0.0000],
          [ 2.5000,  2.7500,  3.2500,  2.6250,  0.8750,  0.0000],
          [ 2.2500,  2.4375,  2.8125,  2.2500,  0.7500,  0.0000],
          [ 0.7500,  0.8125,  0.9375,  0.7500,  0.2500,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000]]]])

>>> m = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
>>> # Notice that values in top left corner are now changed
>>> m(input_3x3)
tensor([[[[ 1.0000,  1.4000,  1.8000,  1.6000,  0.8000,  0.0000],
          [ 1.8000,  2.2000,  2.6000,  2.2400,  1.1200,  0.0000],
          [ 2.6000,  3.0000,  3.4000,  2.8800,  1.4400,  0.0000],
          [ 2.4000,  2.7200,  3.0400,  2.5600,  1.2800,  0.0000],
          [ 1.2000,  1.3600,  1.5200,  1.2800,  0.6400,  0.0000],
          [ 0.0000,  0.0000,  0.0000,  0.0000,  0.0000,  0.0000]]]])
```

‹ Previous Next ›