Difference between 2 reshaping operations (reshape vs permute)

nlp

---

coyote 2018-11-28 08:31:25 UTC #1

Hi everyone,

Recently, I asked **this** question. Though I got an answer for my original question, last **comment** confused me a little bit.

I don't understand the difference between these two cases:

1. According to answers, this is a safe operation:

```
bs,seq_len,input_size= 5,20,128
x = torch.rand(bs,seq_len,input_size)
torch.reshape(x,(x.shape[0],x.shape[1],1,x.shape[2])
```

2. However, if I want to change the shape `(bs, seq_len, input_size)` matrix x into `(seq_len, bs, input_size)`, it is said that I should use `x.permute(1, 0, 2)` rather than reshape.

Why these two cases differs from each other ?

JuanFMontesinos (Juan F Montesinos) 2018-11-28 10:10:07 UTC #2

Im not sure about reshape.
Permute is a multidimensional rotation saying somehow. It keeps the data ordering. View (which is another reshaping method) maps from one dimensionality to another sequentially reading data from the upper dimensions to the lower ones.

So if you want fuse two dimensions into one, you have to apply it over contiguous dimensions or u will modify the data ordering

---

coyote 2018-11-28 10:12:35 UTC #3

> JuanFMontesinos:
>
> So if you want fuse two dimensions into one, you have to apply it over contiguous dimensions or u will modify the data ordering

Thanks for the reply. How do I make sure that I am doing it over contiguous dimensions or not ? Could you, if possible, give me an example on a small tenser with a contiguous and non-contiguous dimensions ?

JuanFMontesinos (Juan F Montesinos) 2018-11-28 11:47:55 UTC #4

if u have a cube

```
c=torch.rand(3,4,5)
```

and you use permute

```
c=torch.rand(3,4,5)
rx  = c.permute(0,2,1)
ry  = c.permute(2,1,0)
rz = c.permute(1,0,2)
print(rx.size())
print(ry.size())
print(rz.size())


torch.Size([3, 5, 4])
torch.Size([5, 4, 3])
torch.Size([4, 3, 5])
```

you are just rotating the tensor, but order is preserved
On the other hand, if you reshape you can see you are modifying the ordering because this is not rotating the cube but mapping in an ordered way from right to left. It takes numbers until it fills the dimensions.

```
sx = c.view(3,5,4)
rx - sx
```

That’s why this operation is different from 0

So an example about how to apply view could be the following one

if you have a tensor m= BxNxHxW
and these tensor contains B batches of N images whose size is HxW and you want to make a montage of these images in a single one concatanating in the colums your outgoing dimension would be

B,H,WxN which is equivalent to B,H,NxW

So lets see what happens if you reshape vs permute + reshape vs permute without paying attention

Im gonna mix this images

with this code

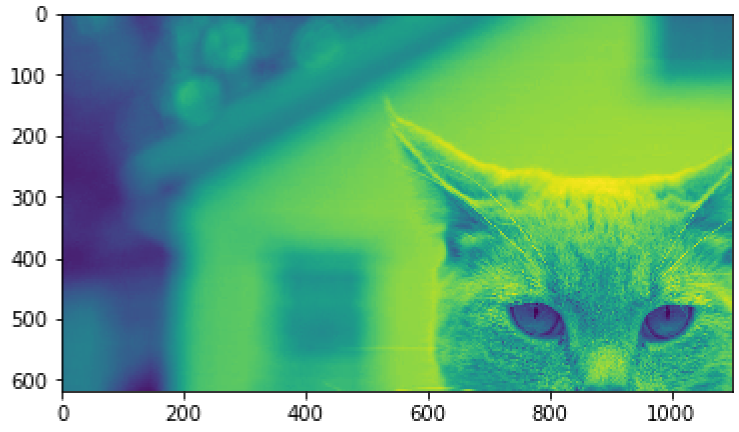
```
import imageio as skio
import matplotlib.pyplot as plt
import numpy as np
import torch
im1 = np.mean(skio.imread('/home/jfm/Downloads/dog.jpg'),axis=2)
im2 = np.mean(skio.imread('/home/jfm/Downloads/cat.jpg'),axis=2)[:618,:1100]

m = np.stack([im1,im2])
m = torch.from_numpy(np.stack([m,m,m]))
print(m.size())
#torch.Size([3, 2, 618, 1100])
```

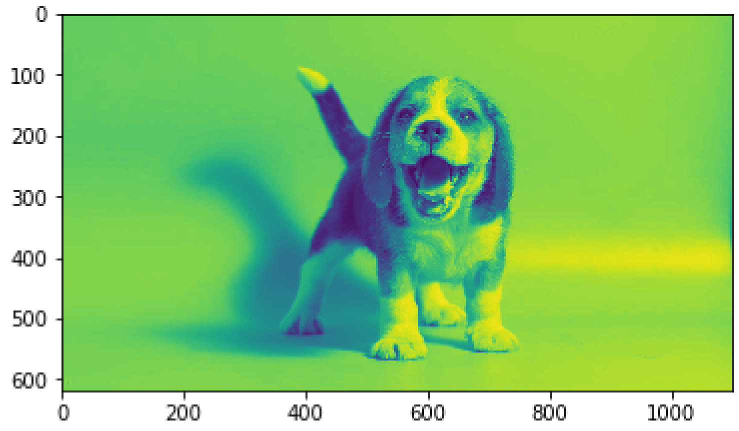
```
m_reshape = m.view(3,618,1100*2).numpy()

m_permute_wrong = m.permute(0,2,3,1).contiguous().view(3,618,1100*2).numpy()
m_permute_right = m.permute(0,2,1,3).contiguous().view(3,618,1100*2).numpy()
```

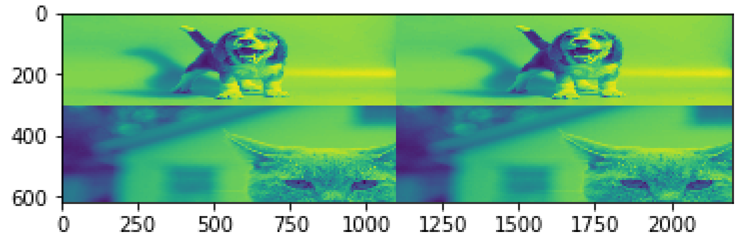
I'm converting RGB images to gray and croping to have same size
Then creating 3 batches of 2 images



The cat cropped looks like that (that's grayscale)

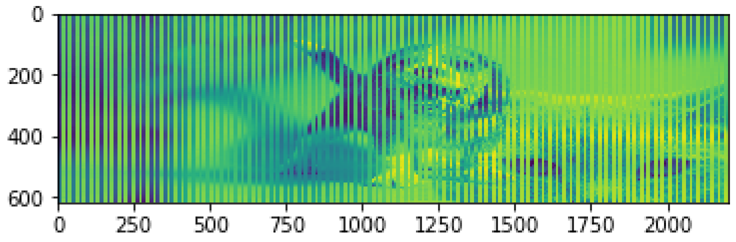


if you just reshape you get a wrong ordering

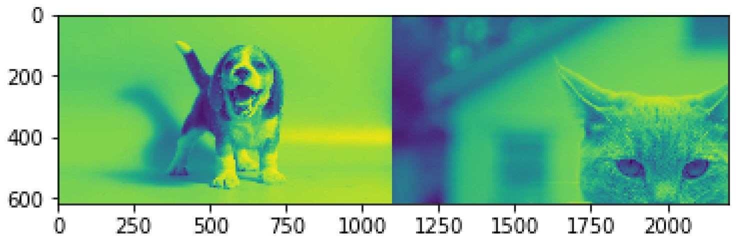


What's going on there? as you are reordering it's getting the information in the original order which is, all colums of image 1, all rows of image 1, all colums of image 2, all rows of image 2 and so on. If u pay attention it 's resized to be fit in the desired shape

if you permute and set dimensions before reshaping
but you do it wrongly
you get this



here you are filling taking the info of one image and then the other because u set N at the right. So it takes the information of the image1, colum 1, then image2, colum 1 and so on.
However if u properly order the dimensions



You achieve what you want which is all the colums of image 1, all the colums of image 2

Moral of this,

If you want to reshape the ordering only remains for contiguous dimensions. [1,2,3,4]
Contiguous here mens 1-2, 2-3 even 1-2-3, but not 1-3 for example.

To do so use permute and the brain first

Loss won't decrease if batch_size > 1

Concatenate dimensions in a tensor