Introduction to Tensors

Deep Learning Lab - Week 1

Author: Sambhav Jain

Date: 9 January 2025 @ 9:30AM

Lab was held on Friday, 3 January 2025 and has been repeated on the 9th of January

Q1. Squeeze, Unsqueeze, View, Stack, Reshape Tensors

In [12]:

```
import torch
```

In [18]:

```
print(torch.cuda.is_available())

device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

```
True
```

Out[18]:

```
'cuda'
```

In [62]:

```
tensor = torch.arange(9)
print(tensor)
print()

# reshaping a tensor to a 3x3 matrix
reshaped_tensor = torch.reshape(tensor, (3, 3))
print(reshaped_tensor)
print()

# viewing a tensor (creating a new view of the same data without copying it)
view_tensor = tensor.view(3, 3)
print(view_tensor)
print()

# stacking 2 tensors along a new dimension (dim=0, creating a 2x2 tensor)
tensor_1 = torch.tensor([1, 2])
tensor_2 = torch.tensor([3, 4])
stacked_tensor = torch.stack([tensor_1, tensor_2], dim=0)
print(stacked_tensor)
print()

# squeezing a tensor (removes dimensions with size 1)
new_tensor = torch.zeros(1, 2, 3)  # tensor with shape (1, 2, 3)
print(new_tensor, new_tensor.size())
print()
squeeze_tensor = torch.squeeze(new_tensor, dim=0)  # removes the dimension with size 1 at index 0
print(squeeze_tensor)
print()

# unsqueezing a tensor (adds a new dimension with size 1 at the specified index)
unsqueeze_tensor = torch.unsqueeze(tensor_1, dim=0)  # adds a new dimension at index 0
print(unsqueeze_tensor)
```

```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8])

tensor([[0, 1, 2],
        [3, 4, 5],
        [6, 7, 8]])

tensor([[0, 1, 2],
        [3, 4, 5],
        [6, 7, 8]])

tensor([[1, 2],
        [3, 4]])

tensor([[[[0., 0., 0.],
         [0., 0., 0.]]]]) torch.Size([1, 2, 3])

tensor([[0., 0., 0.],
        [0., 0., 0.]])

tensor([[1, 2]])
```

In [14]:

```
# # Creating a Tensor
# t = torch.tensor([[1,2,3,4],[5,6,7,8]])
# t1 = torch.tensor([[11,12],[13,14],[15,16],[17,18]])
# print("Tensor: ",t)
# print("Tensor Shape: ", t.size())

# # Reshaping Tensor
# print("Reshaped Tensor: ",t.reshape(4,2))

# # Stacking Tensors
# print("Stacked Tensor: ",torch.stack([t1, torch.transpose(t,1,0)],dim=1))
```

```
Tensor:  tensor([[1, 2, 3, 4],
        [5, 6, 7, 8]])
Tensor Shape:  torch.Size([2, 4])
Reshaped Tensor:  tensor([[1, 2],
        [3, 4],
        [5, 6],
        [7, 8]])
Stacked Tensor:  tensor([[[11, 12],
         [ 1,  5]],

        [[13, 14],
         [ 2,  6]],

        [[15, 16],
         [ 3,  7]],

        [[17, 18],
         [ 4,  8]]])
```

Q2. Demonstrate permute()

In [64]:

```
original = torch.randn(1, 3, 5)
print(f"original tensor dimensions:  {original.shape}")

# This swaps axes 0 and 1.
# the new shape becomes (3, 1, 5)
permuted = original.permute(1, 0, 2)
print(f"permuted tensor dimensions:  {permuted.shape}")
```

```
original tensor dimensions:  torch.Size([1, 3, 5])
permuted tensor dimensions:  torch.Size([3, 1, 5])
```

Q3. Indexing in Tensors

In [67]:

```
tensor = torch.rand(5)
print(tensor)

# element at index 4 (5th element) of the tensor
print(tensor[4])
```

```
# accessing the last element
print(tensor[-1])

# accessing elements at indices 1, 3, and 4
print(tensor[[1, 3, 4]])
```

```
tensor([0.2816, 0.6679, 0.7878, 0.5070, 0.3055])
tensor(0.3055)
tensor(0.3055)
tensor([0.6679, 0.5070, 0.3055])
```

Q4. Numpy <-> Tensors Conversion

In [63]:

```
t = torch.tensor([[1,2,3,4],[5,6,7,8]])
t1 = torch.tensor([[11,12],[13,14],[15,16],[17,18]])

npT = t.numpy()

print("Numpy Array: ", npT)

pyTtensor = torch.from_numpy(npT)
print("PyTorch Tensor: ", pyTtensor)
```

```
Numpy Array:  [[1 2 3 4]
 [5 6 7 8]]
PyTorch Tensor:  tensor([[1, 2, 3, 4],
        [5, 6, 7, 8]])
```

Q5. Random Tensor with 7,7 shape

In [16]:

```
ranT = torch.randn(size=(7,7))
print("Random Tensor: ", ranT, '\n' ,ranT.size())
```

```
Random Tensor:  tensor([[-1.4268,  0.2472, -1.0552, -1.2551, -0.0147,  0.9517,  0.0883],
        [-0.5338, -1.1770,  1.4360, -0.7218, -0.1743,  0.0325, -0.4588],
        [ 0.7765, -0.7703, -0.1804, -0.5596, -0.3212, -0.0074, -2.3763],
        [-0.3617, -0.9903, -2.4624, -0.0425, -0.4172,  0.1501,  0.9069],
        [ 1.0727, -0.8715,  1.1657,  1.3334, -0.6850, -0.3945,  1.6860],
        [-1.1309,  2.1067,  1.4360, -0.1509,  1.9820, -0.1197,  0.3296],
        [ 0.0972,  1.1556,  0.3890,  1.4442,  0.7443, -2.2397,  0.4780]])
 torch.Size([7, 7])
```

Q6. Matrix Multiplication between previously defined tensor and a new (1,7) tensor.

In [17]:

```
tensor6 = torch.randn((1,7))

# Regular Matrix Multiplication
print(ranT @ torch.transpose(tensor6,1,0))
#                                     Tensor, dim0, dim1
print(torch.matmul(ranT,torch.transpose(tensor6,1,0)))

# EXTRA: Element-wise Matrix Multiplication
print(ranT * torch.transpose(tensor6,1,0))
```

```
tensor([[-0.3765],
        [-0.5213],
        [ 1.3634],
        [-2.8580],
        [-1.7691],
        [ 1.8861],
        [ 1.0618]])
tensor([[-0.3765],
        [-0.5213],
        [ 1.3634],
        [-2.8580],
        [-1.7691],
        [ 1.8861],
        [ 1.0618]])
tensor([[-0.3232,  0.0560, -0.2390, -0.2843, -0.0033,  0.2156,  0.0200],
        [-0.6943, -1.5309,  1.8677, -0.9388, -0.2267,  0.0422, -0.5967],
        [ 0.2828, -0.2806, -0.0657, -0.2038, -0.1170, -0.0027, -0.8656],
        [ 0.0753,  0.2060,  0.5123,  0.0088,  0.0868, -0.0312, -0.1887],
        [-0.4867,  0.3954, -0.5288, -0.6049,  0.3108,  0.1790, -0.7649],
        [ 0.2190, -0.4080, -0.2781,  0.0292, -0.3839,  0.0232, -0.0638],
        [-0.0814, -0.9685, -0.3260, -1.2103, -0.6237,  1.8770, -0.4006]])
```

Q7. Create 2 random tensors of (2,3) shape and send them to the GPU

In [23]:

```
tensorGPU1 = torch.randn(size=(2,3))
tensorGPU2 = torch.randn(size=(2,3))

print(tensorGPU1.to(device))
print(tensorGPU2.to(device))
```

```
tensor([[ 1.4820,  0.7286, -0.2261],
        [ 1.4599,  0.6856,  0.7849]], device='cuda:0')
tensor([[ 0.0910, -0.3467,  1.7950],
        [ 0.5767,  0.2679,  0.1201]], device='cuda:0')
```

Q8. Perform Matrix Multiplication on Tensors in Question 7.

In [27]:

```
result = torch.matmul(tensorGPU1, tensorGPU2.reshape((3,2))).to(device)
print(result)
```

```
tensor([[ 1.3822, -0.1208],
        [ 1.5738, -0.0165]], device='cuda:0')
```

Q9. Max and Minimum values of Output of 8.

In [28]:

```
print("Max Value: ",torch.max(result))
print("Min Value: ",torch.min(result))
```

```
Max:  tensor(1.5738, device='cuda:0')
Min:  tensor(-0.1208, device='cuda:0')
```

Q10. Max and Minimum values of INDICES of Output of 8.

In [29]:

```
print("Max Index Value: ",torch.argmax(result))
print("Min Index Value: ",torch.argmin(result))
```

```
Max Index Value:  tensor(2, device='cuda:0')
Min Index Value:  tensor(1, device='cuda:0')
```

Q11. Make a random tensor with shape (1, 1, 1, 10) and then create a new tensor with all the 1 dimensions removed to be left with a tensor of shape (10). Set the seed to 7 when you create it and print out the first tensor and it's shape as well as the second tensor and it's shape.

In [60]:

```
# Creating Random Tensor
torch.manual_seed(7)

randomTensor = torch.randn(size=(1,1,1,10))
print("Original Tensor: ", randomTensor)
print("Original Tensor Shape: ", randomTensor.size())
print()
# Remove the 1 Dimensions using the SQUEEZE Function
squeezed = torch.squeeze(randomTensor)
```

```
print("Squeezed Tensor: ", squeezed)
print("Squeezed Tensor Shape: ", squeezed.size())
```

```
Original Tensor:  tensor([[[[-0.1468,  0.7861,  0.9468, -1.1143,  1.6908, -0.8948, -0.3556,
            1.2324,  0.1382, -1.6822]]]])
Original Tensor Shape:  torch.Size([1, 1, 1, 10])

Squeezed Tensor:  tensor([-0.1468,  0.7861,  0.9468, -1.1143,  1.6908, -0.8948, -0.3556,  1.2324,
         0.1382, -1.6822])
Squeezed Tensor Shape:  torch.Size([10])
```