Padding Sequences with ‘batch_first‘ in PyTorch’s DataLoader

Introduction

In PyTorch, the `DataLoader` class is used to load data in batches. When dealing with sequences of varying lengths, the `pad_sequence` function from `torch.nn.utils.rnn` is used to pad sequences to the same length. The `pad_sequence` function has a parameter `batch_first`, which controls the arrangement of the dimensions of the padded tensor. This note provides two code examples to demonstrate the impact of `batch_first=True` and `batch_first=False` on the padded sequences.

Code Example with **`batch_first=True`**

```
import torch
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence

data = [torch.tensor([1, 2]), torch.tensor([3]), torch.tensor([7,
    red↪  8, 9])]
label = torch.tensor([1, 0, 1])

class VariableLengthDataset(torch.utils.data.Dataset):
   def __getitem__(self, idx):
      return data[idx], label[idx]

   def __len__(self):
      return len(data)

def collate_fn(batch):
   data, labels = zip(*batch)
```

```
    data = pad_sequence(data, batch_first=True, padding_value=0)
    labels = torch.tensor(labels)
    return data, labels

dataset = VariableLengthDataset()
dataloader = DataLoader(dataset, batch_size=2, collate_fn=
    red↪ collate_fn)

for batch in dataloader:
    print(batch)
```

Output:

```
(tensor([[1, 2],
        [3, 0]]), tensor([1, 0]))
(tensor([[7, 8, 9]]), tensor([1]))
```

Here, the padded data tensor has the shape (batch_size, max_sequence_length).

Code Example with `batch_first=False`

```
import torch
from torch.utils.data import DataLoader
from torch.nn.utils.rnn import pad_sequence

data = [torch.tensor([1, 2]), torch.tensor([3]), torch.tensor([7,
    red↪  8, 9])]
label = torch.tensor([1, 0, 1])

class VariableLengthDataset(torch.utils.data.Dataset):
    def __getitem__(self, idx):
        return data[idx], label[idx]

    def __len__(self):
        return len(data)

def collate_fn(batch):
    data, labels = zip(*batch)
    data = pad_sequence(data, batch_first=False, padding_value=0)
    labels = torch.tensor(labels)
    return data, labels

dataset = VariableLengthDataset()
```

```
dataloader = DataLoader(dataset, batch_size=2, collate_fn=
    red↪ collate_fn)

for batch in dataloader:
    print(batch)
```

Output:

```
(tensor([[1, 3],
        [2, 0]]), tensor([1, 0]))
(tensor([[7],
        [8],
        [9]]), tensor([1]))
```

Here, the padded data tensor has the shape (max_sequence_length, batch_size).