Understanding Custom Collate Function in PyTorch DataLoader

Introduction

In this note, we will explain a custom collate function used in PyTorch's DataLoader. This function processes batches of data to ensure they meet specific requirements before the data is used in subsequent operations or passed to a model.

Code

```
import torch
from torch.utils.data import DataLoader, TensorDataset

# Sample dataset
data = torch.tensor([[1, 2], [3, 4], [5, 6], [7, 8]])
labels = torch.tensor([0, 1, 0, 1])

# Create TensorDataset
dataset = TensorDataset(data, labels)

# Define custom collate function
def custom_collate(batch):
    data, labels = zip(*batch)
    data = torch.stack(data).unsqueeze(1)
    labels = torch.tensor(labels)
    return data, labels

# Create DataLoader with custom collate function
dataloader = DataLoader(dataset, batch_size=2, shuffle=True,
    collate_fn=custom_collate)
```

```
20
21 # Iterate through DataLoader
22 for batch in dataloader:
23     print(batch)
```

Explanation

Input

`batch` is a list of tuples, where each tuple contains a data tensor and a label.

Operation 1: `data, labels = zip(*batch)`

`zip(*batch)` unpacks the batch of tuples into two separate tuples: one containing all the data tensors and the other containing all the labels.

- `data` now holds a tuple of data tensors.
- `labels` now holds a tuple of labels.

Operation 2: `data = torch.stack(data).unsqueeze(1)`

- `torch.stack(data)` stacks the individual data tensors along a new dimension to form a single tensor.
- `.unsqueeze(1)` adds an extra dimension at index 1. This can be useful if you need to add a channel dimension or similar.

Operation 3: `labels = torch.tensor(labels)`

- Converts the tuple of labels into a single tensor.

Return

Returns a tuple `(data, labels)` where `data` is a tensor with an extra dimension added, and `labels` is a tensor of labels.

Example

Let's assume we have a dataset with 4 samples as follows:

- Data: `[[1, 2], [3, 4], [5, 6], [7, 8]]`
- Labels: `[0, 1, 0, 1]`

Custom Collate Function

For the first batch (assuming `shuffle=True`, the order may vary):

- `batch = [([1, 2], 0), ([3, 4], 1)]`
- `zip(*batch)` results in: `data = ([1, 2], [3, 4])` and `labels = (0, 1)`
- `torch.stack(data)` results in: `tensor([[1, 2], [3, 4]])`
- `.unsqueeze(1)` adds an extra dimension: `tensor([[[1, 2]], [[3, 4]]])`
- `torch.tensor(labels)` results in: `tensor([0, 1])`
- The custom collate function returns: `(tensor([[[1, 2]], [[3, 4]]]), tensor([0, 1]))`

DataLoader Iteration

The `for` loop iterates through the DataLoader, and for each batch, the processed data and labels are printed.