# Combining and Shuffling Datasets in PyTorch with DataLoader

## Introduction

In this document, we demonstrate how to combine two datasets and shuffle them using PyTorch's 'DataLoader'. We will create a custom dataset class to handle the combination of two datasets, define a custom collate function, and iterate through the DataLoader to process the data in batches.

## Code Example

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

# Data and Labels
data1 = torch.tensor([[1, 2], [3, 4], [5, 6]])
data2 = torch.tensor([[7, 8], [9, 10], [11, 12]])
labels1 = torch.tensor([0, 1, 0])
labels2 = torch.tensor([1, 0, 1])

# Custom Dataset Class
class CombinedDataset(Dataset):
   def __getitem__(self, index):
      if index < len(data1):
         return data1[index], labels1[index]
      else:
         return data2[index - len(data1)], labels2[index - len(
            red↪ data1)]

   def __len__(self):
      return len(data1) + len(data2)
```

```
# Custom Collate Function
def combined_collate(batch):
   data, labels = zip(*batch)
   data = torch.stack(data)
   labels = torch.tensor(labels)
   return data, labels

# DataLoader Initialization
dataset = CombinedDataset()
dataloader = DataLoader(dataset, batch_size=2, shuffle=True,
   red↪ collate_fn=combined_collate)

# Iterating through DataLoader
for batch in dataloader:
   print(batch)
```

# Code Breakdown

## Data and Labels

```
data1 = torch.tensor([[1, 2], [3, 4], [5, 6]])
data2 = torch.tensor([[7, 8], [9, 10], [11, 12]])
labels1 = torch.tensor([0, 1, 0])
labels2 = torch.tensor([1, 0, 1])
```

- **data1** and **labels1**: Represent the first dataset.
- **data2** and **labels2**: Represent the second dataset.

## Custom Dataset Class

```
class CombinedDataset(torch.utils.data.Dataset):
   def __getitem__(self, index):
      if index < len(data1):
         return data1[index], labels1[index]
      else:
         return data2[index - len(data1)], labels2[index - len(
            red↪ data1)]

   def __len__(self):
```

```
        return len(data1) + len(data2)
```

- **CombinedDataset**: A custom dataset class that combines **data1** and **data2** along with their labels.
- __getitem__: Returns the data and label based on the index. If the index is within the range of **data1**, it returns the corresponding element from **data1** and **labels1**. If the index exceeds the length of **data1**, it fetches from **data2** and **labels2** after adjusting the index.
- __len__: Returns the total length of the combined datasets.

## Custom Collate Function

```
def combined_collate(batch):
    data, labels = zip(*batch)
    data = torch.stack(data)
    labels = torch.tensor(labels)
    return data, labels
```

- $\textbf{combined}_collate$ : $A custom collate function that stacks the batch data and labels into tensors.$

## DataLoader Initialization

```
dataset = CombinedDataset()
dataloader = DataLoader(dataset, batch_size=2, shuffle=True,
    red↪ collate_fn=combined_collate)
```

- **dataset**: An instance of the **CombinedDataset**.

- **dataloader**: A DataLoader that loads data from the **CombinedDataset** in batches of size 2, shuffles the data, and uses the $\textbf{combined}_collate function to process each batch.$

## Iterating through DataLoader

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

# Example Execution

Let's simulate one possible execution of the DataLoader with shuffling. Note that the actual output may vary due to shuffling.

## Data Preparation

- **data1**: `[[1, 2], [3, 4], [5, 6]]`
- **data2**: `[[7, 8], [9, 10], [11, 12]]`
- **labels1**: `[0, 1, 0]`
- **labels2**: `[1, 0, 1]`

## Combined Dataset

The total length of the combined dataset is 6 (3 + 3).

## Shuffling

Let's assume the indices after shuffling are `[4, 2, 5, 0, 1, 3]`.

## Batches

- **Batch 1**: Indices `[4, 2]` → `data2[1]` and `data1[2]`

```
data = [[9, 10], [5, 6]]
labels = [0, 0]
```

- **Batch 2**: Indices `[5, 0]` → `data2[2]` and `data1[0]`

```
data = [[11, 12], [1, 2]]
labels = [1, 0]
```

- **Batch 3**: Indices `[1, 3]` → `data1[1]` and `data2[0]`

```
data = [[3, 4], [7, 8]]
labels = [1, 1]
```

## Sample Output

```
(tensor([[ 9, 10],
       [ 5, 6]]), tensor([0, 0]))
(tensor([[11, 12],
       [ 1, 2]]), tensor([1, 0]))
(tensor([[3, 4],
       [7, 8]]), tensor([1, 1]))
```

## Explanation

- **Batch 1**:
  - Data: `[[9, 10], [5, 6]]`
  - Labels: `[0, 0]`
- **Batch 2**:
  - Data: `[[11, 12], [1, 2]]`
  - Labels: `[1, 0]`
- **Batch 3**:
  - Data: `[[3, 4], [7, 8]]`
  - Labels: `[1, 1]`

This example demonstrates how the DataLoader fetches and processes batches from a combined dataset, shuffling the indices and collating the data and labels into tensors.