在 PyTorch 中，提供了一种十分方便的数据读取机制，即使用TensorDataset 与 Dataloader 组合得到数据迭代器。在每次训练时，利用这个迭代器输出每一个 batch 数据。

流程是先把原始数据包装成 TensorDataset 类，形成训练集，随后再把得到的TensorDataset 类的数据集实例当作一个参数传递给DataLoader 类，得到一个数据加载器实例，这个数据加载器每次可以返回一个 batch\_size 的数据供模型训练使用。

除TensorDataset 类外，还可以用自定义的数据集类。

**train\_dataset =torch.utils.data.TensorDataset(samples,labels)**

samples表示样本值

labels表示样本标签

**构建自己的Dataset类**

torch.utils.data.Dataset是PyTorch中用来表示数据集的抽象类，Dataset是一个包装类，用来将数据包装为Dataset类，然后传入DataLoader中从而使DataLoader类更加快捷的对数据进行操作。当处理自定义的数据集的时候必须继承Dataset，然后重写\_\_len\_\_()和\_\_getitem\_\_()函数

(1)\_\_len\_\_(): 使得len(dataset)返回数据集的大小；

(2)\_\_getitem\_\_()：使得支持dataset[i]能够返回第i个数据样本

**torch.utils.data. DataLoader (train\_dataset,batch\_size,shuffle,num\_workers)**

train\_dataset表示一个数据集对象

batch\_size表示每次加载的数据批量  
shuffle表示是否打乱数据

num\_workers表示多线程读取数据

*#形成训练对列表，用于喂给train\_dataset*

pairs\_X = [pair[0] for pair in pairs]

pairs\_Y = [pair[1] for pair in pairs]

train\_dataset = DataSet.TensorDataset(pairs\_X, pairs\_Y) *#形成训练集*

train\_loader = DataSet.DataLoader(train\_dataset, batch\_size = 8, shuffle = True, num\_workers=8) *#形成数据加载器械*

for epoch in range(num\_epoch):

model.train()

for data in train\_loader:

input = data[0]*# input的大小：batch\_size, input\_size*

target = data[1]*# target的大小：batch\_size*

output = model(input) *# ouput大小：batch\_size, output\_size*

loss = criterion(output, target)

loss.backward()

optimizer.step()