深蓝学院
shenlanxueyuan.com

第五章作业思路提示

主讲人 陈彭鑫

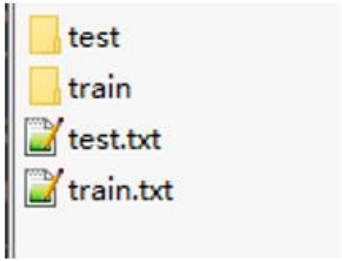
作业

- 实现dataset

  数据集格式为list，每行内容为图像路径+label，请实现一个新的dataset从此格式读取数据，并训练minist模型

- 实现transform

  随机旋转数据增广，图像可随机进行0,90,180,270度旋转，加入到minist模型训练中




```
train/0.jpg 5
train/1.jpg 0
train/2.jpg 4
train/3.jpg 1
train/4.jpg 9
train/5.jpg 2
train/6.jpg 1
```

```
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('data', train=True, download=True,
                   transform=transforms.Compose([
                       rotate_transform(),
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
```

纲要

深蓝学院
shenlanxueyuan.com

- 第一部分：dataset
- 第二部分：data augmentation

1. torch.utils.data.Dataset

`torch.utils.data.Dataset` 是代表自定义数据集方法的类，用户可以通过继承该类来自定义自己的数据集类，在继承时要求用户重载 `__len__()` 和 `__getitem__()` 这两个魔法方法。

- `__len__()`：**返回的是数据集的大小**。我们构建的数据集是一个对象，而数据集不像序列类型（列表、元组、字符串）那样可以直接用 `len()` 来获取序列的长度，魔法方法 `__len__()` 的目的就是方便像序列那样直接获取获取对象的长度。如果 `a` 是一个对象，当 `a` 中定义了魔法方法 `__len__()`，`len(a)` 则返回对象的大小。
- `__getitem__()`：**实现索引数据集中的某一个数据**。我们知道，序列可以通过索引的方法获取序列中的任意元素，`__getitem__()` 则实现了能够通过索引的方法获取对象中的任意元素。此外，我们可以在 `__getitem__()` 中**实现数据预处理**。

2. torch.utils.data.DataLoader

作用：

- `DataLoader` 将 `Dataset` 或其子类封装成一个迭代器；
- 这个迭代器可以迭代输出 `Dataset` 的内容；
- 同时可以实现多进程、shuffle、不同采样策略，数据校对等等处理过程。

```
tensor_dataloader = DataLoader(tensor_dataset,   # 封装的对象
                               batch_size=2,     # 输出的batch size
                               shuffle=True,     # 随机输出
                               num_workers=0)    # 只有1个进程

# 以for循环形式输出
for data, target in tensor_dataloader:
    print(data, target)
```

登录后复制

dataset

深蓝学院
shenlanxueyuan.com

```
class MyDataset(torch.utils.data.Dataset):
    def __init__(self, dataname, transform=None):
        # 初始化：数据位置、输入大小
        # 尽量在这里完成数据的读取和预处理
    def __getitem__(self, index):
        # 提取一对数据
        # 这里尽量不要运行open等函数，直接返回img和label
        return image, int(self.labels[index])
    def __len__(self):
        # 总共提取的个数
        return len(self.filenames)
```

纲要

- 第一部分：dataset
- 第二部分：data augmentation

```
import numpy as np
import cv2

def rotate_image(image, angle):
  image_center = tuple(np.array(image.shape[1::-1]) / 2)
  rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
  result = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)
  return result
```

Assuming you're using the cv2 version, that code finds the center of the image you want to rotate, calculates the transformation matrix and applies to the image.

- 1

```
def __init__(self, train=True, rotate=False):
    ...
    ...

        if rotate:
            angle = np.random.choice([0, 90, 180, 270])
            img_rot = self.rotate_image(img, angle)
            _img_rot = img_rot[np.newaxis, :, :].astype(np.float32)
            self.ALL_DATA.append((_img_rot, _label))
```

- 2

```
train_loader = torch.utils.data.DataLoader(
    datasets.MNIST('data', train=True, download=True,
                   transform=transforms.Compose([
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])),
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening