第三步教辅

在第三步中主要是写代码，因此我主要围绕项目的流程来进行讲解。希望可以通过本篇教程来让同学们对这个项目有个更快、更深的理解。

项目地址

wxkang157/PytorchPipeline: Pytorch pipeline template (github.com)

流程

训练流程一般分为以下下三个步骤：
一、数据准备
二、构建模型
三、训练与评估

数据准备与读取

项目中提高的数据集的数据格式为：

```
data
|--train
    |--cat
    |--dog
    | ....
|
|--validation
    |--cat
    |--dog
|   | ....
```

可以看到，不同类别的图片是放在不同文件夹下的。

以及在这阶段分为train_data (或者train_data, valid_data)以及test_data。这里的train、validation、test就是用于区分训练集、验证集、测试集。

有了数据，就要对数据进行读取，然后送入到模型中进行训练或者测试。数据读取，主要是通过torch.utils.data.Dataset完成。

在这里，需要继承DataSet来构建数据集，在构建时最主要的是需要重载其中的`__getitem__()`函数。
getitem：输入是索引，输出是样本（加标签），也就是定义了索引到样本的映射规则。
其余两个比较重要的函数是`__init__()`函数和`__len__()`函数，通过`__init__()`此方法用于对象实例化，通常用来提供类中需要使用的变量。
而`__len__()`函数是返回数据集的size大小。

然后再将这个DataSet放入DataLoader（torch.utils.data.DataLoader）中， DataLoader的功能是构建可迭代的数据装载器。
这个类中的主要需要修改的参数有：
dataset：Dataset类，决定数据从哪里读取以及如何读取
batchsize：批处理的大小
num_works：是否多进程读取数据
shuffle：每个epoch是否乱序 (true or false)
drop_last：当样本数不能被batchsize整除时，是否丢弃最后一批数据。(true or false)

构建模型

构建模型我们这里一般使用的是CNN，通过Pytorch构建模型时需要注意两个点。
`__init__()`, 每个模型都会继承nn.Module,你需要指定Layer的结构,计算每个Layer的参数维度，比如self.conv1 = nn.Conv2d(3, 6, 5)、self.pool = nn.MaxPool2d(2, 2)。
forward(x), 通过诸如net(inputs)这样的方法，可以自动调用模型中forward这个方法，表示的是指明一个batch的输入数据x,流经网络的过程。

训练与评估

在这里，你需要定义网络,损失函数,参数优化方法。
整体的流程为：
1.设定参数,定义网络
2.定义损失函数,参数优化方法
3.送入设定的设备中
4.遍历数据集开始训练，训练时使用train()模式
5.进行模型的前向推理和梯度的反向传播计算
6.模型的测试，测试时使用eval()模式,关闭dropout和batch normalization
7.根据测试时的指标进行计算，比方说需要计算acc准确率。
8.根据你所指定的指标如acc保存模型。

附录 常用的库、常用的函数

os常用命令：

os.getcwd()--得到当前工作的目录。
os.listdir()--指定所有目录下所有的文件和目录名,列表的形式全部列举出来，其中没有区分目录和文件。
os.remove()--删除指定文件
os.rmdir()--删除指定目录
os.mkdir()--创建目录,注意：这样只能建立一层，要想递归建立可用：os.makedirs()
os.path.isfile()--判断指定对象是否为文件。是返回True,否则False
os.path.isdir()--判断指定对象是否为目录。是True,否则False。
os.path.exists()--检验指定的对象是否存在。是True,否则False。
os.path.abspath()--获得绝对路径。
os.path.join(path, name)--连接目录和文件名。
os.path.basename(path)--返回文件名
os.path.dirname(path)--返回文件路径

PIL常用命令：

1、open()打开图片

```
from PIL import Image
pil_im = Image.open('empire.jpg')
"""open打开图片，pil_im 返回是一个PIL图像对象"""
```

2、convert()图像的颜色转换，L将图像装换为灰度图像

```
pil_im = Image.open('empire.jpg').convert('L')
```

3、save()保存图片

```
from PIL import Image
im = Image.open("E:\mywife.jpg")
im.save("E:\mywife.png")      ## 将"E:\mywife.jpg"保存为"E:\mywife.png"
im = Image.open("E:\mywife.png")  ##打开新的png图片
```

numpy常用命令：

np.array()，将输入数据（列表、元组、数组或其它序列类型）转换为ndarray。要么推断出dtype，要么指定特别dtype。默认直接复制输入数据。
np.zeros() ： 创建指定长度或形状的全0数组，只需传入一个表示形状的元组。
np.zeros_like() ： 返回与指定数组同样形状和数据类型的全0数组。
np.ones() ： 创建指定长度或形状的全1数组，只需传入一个表示形状的元组。
np.ones_like() ： 返回与指定数组同样形状和数据类型的全1数组。
np.empty()| np.empty_like() ： 创建没有任何具体值的数组。
np.arange()：类似于内置的range，但返回的是一个ndarray而不是列表。
np.full(shape, fill_value)|np.full_like()：用fill_value中的所有值，根据指定的形状和dtype创建数组。
查看shape: a.shape
查看数据类型： a.dtype
更改数据类型：a.astype(dtype)
检查是否为某类型：np.issubdtype(a.dtype, np.integer)，np.issubdtype(a.dtype, np.floating)，np.issubdtype(a.dtype, np.number)

Matplotlib下plt常用命令：

plt.imshow(image)：表示对图像image进行处理，可以打印出来图像的数字形式，但是就是无法可视化的显示出来
plt.show()：将plt.imshow()处理后的函数显示出来
plt.xlabel('x-axis')：设置X轴的标签文本
plt.ylabel('y-axis')：设置Y轴的标签文本
plt.title('title')：用于设置图像标题
plt.xticks(ticks, labels, **kwargs)：画图时默认的横纵坐标显示的值达不到自己的要求时使用xticks()对横坐标轴进行设置
plt.yticks(ticks, labels, **kwargs)：同上，对纵坐标轴进行修改
plt.savefig('fig.png')：用于保存生成的图片

pytroch常用命令：

1.torch.eye(n,m)
创建一个对角线为1，其他为0的2维tensor
eg: x = torch.eys(3)

2.torch.linspace(start,end,steps)
创建一个1维tensor，start为3，end为10，共5个数
eg: x = torch.linspace(3,10,steps=5)

3.torch.ones(*size)
torch.zeros(*size)
返回一个全1的tensor，形状由size决定
eg: x = torch.ones(3,4)

4.torch.rand(*size)
返回从[0，1）均匀分布随机抽取的数，形状由size决定
eg: x = torch.rand(3,4)

5.torch.randn(*size)

返回从均值为0，方差为1的高斯分布中随机抽取的数，形状有size决定
eg: x = torch.randn(3,4)

6.torch.arange(start,end,step)
返回从start到end的tensor，步长为step
eg: x = torch.arange(3,10,2)

7.torch.from_numpy(ndarray)
把numpy类型数组转化为tensor

8.torch.Tensor(data) == FloatTensor()
创建一个FloatTensor
eg：x = torch.Tensor(1)
x:[0],这里的标量1传入的是维度
eg: x = torch.Tensor([1])
x:[1]，这里传入的是tensor[1]值

9.torch.tensor(data)
这里的data可以是tuple，list，ndarray，scalar，根据原始的数据类型生成
LongTensor,FloatTensor,DoubleTensor
eg:x = torch.tensor(1)
x:1，这里的的类型是LongTensor

基本操作：
1.torch.cat((x,y),dim=-1)
在指定维度进行拼接

2.torch.chunk(tensor, chunks, dim=0)
与cat相反，在指定维度进行分割
chunks:分割的数目

3.torch.gather(input, dim, index)
根据index在指定的维度进行聚合
eg: t = torch.Tensor([[1,2],[3,4]])
x = torch.gather(t, 1, torch.LongTensor([[0,0],[1,0]]))
x: [1,1],[4,3]

4.torch.index_select(input,dim,index)
根据index从指定维度选取向量

5.torch.masked_select(input,mask)
取mask矩阵中为True的值

6.torch.squeeze(input,dim)
压缩指定为1的维度

7.torch.transpose(input,dim0,dim1)
对于指定的两个维度进行交换
torch.permute(index)
可以根据指定index进行交换
torch.t()
进行转置

8.torch.expand()
把tensor每个维度扩充为指定维度
torch.repeat()
把tensor每个维度重复多少倍