How-to create a PyTorch project from scratch

实验目的

通过本次实验，学习基于PyTorch的人工智能项目的框架设计，达到根据实际需求从头编写完整项目的目标。

实验内容

从一个最简单的例子讲起（20pts）

> 你将通过此案例，学习PyTorch项目的基本构成。源代码文件为`./code/01.toy.py`。

一个完整的PyTorch项目将分成若干个基本组成部分。一般来说，可以归纳如下：

- 数据处理
- 模型设计
- 损失函数与优化器
- 训练器
- 评估

好在Python中的很多模块为我们提供了功能强大的接口，我们在完成一个PyTorch项目的时候可以不用考虑过多底层实现的细节。例如，PyTorch为我们提供了自动求导、卷积、池化、优化器、常用预训练模型、异部数据加载器等功能，TorchVision为我们提供了常用图像数据集、图像数据增强等功能，Scikit为我们提供了混淆矩阵、交叉验证、决策树、ensembling等功能。

因此，我们在编码的过程中，可以不用像以往一样实现每个操作的细节，而是调用这些常用库的接口，快速完成一个项目。

在本例中，你需要完整阅读范例代码和注释，学习PyTorch项目需要的基本组成部分和它们之间的关系。同时，你需要完成代码中的留空部分，即实现交叉熵损失函数。

> 注意，交叉熵损失函数是PyTorch官方给出的一个功能，但是在这里，你需要自己实现一个交叉熵损失函数。
>
> 损失函数有两种实现方式：函数和类。
>
> 1. 函数实现方式：这种损失函数为一个有如下签名的函数
>
> ```
> def loss(output, target):
>     pass
> ```
>
> 其中`output`为模型输出的结果，`target`为这组输入对应的ground-truth。
>
> 2. 类实现方式：这种损失函数为继承`nn.Module`的一个类。由于篇幅原因，此处不详细阐述，可以参考PyTorch官方文档。
>
> 均方误差损失函数的公式如下：
>
> 1. 对单个数据：

$$MSE\_Loss(x, class) = \sum_{i \neq class} x[i]^2 + x[class]^2$$

2. 对数据集：

$$MSE\_Loss(X) = \sum_{x \in X} MSE\_Loss(x, class_x)$$

最后你可以运行本项目，训练一个可以实现“异或”操作的模型。

学习数数（30pts）

> 你将通过此案例，学习PyTorch项目中一个关键部分的实现方式——数据集与数据加载器（Dataset & DataLoader）。源代码文件为`./code/02.learn-to-count.py`。

数据集和数据加载器是PyTorch和TorchVision提供的处理数据的优秀工具。

- 数据集（Dataset）：定义了一个数据集类，用来指出图片数据的存放位置，数量和读取方法等。一般来说，一个Dataset Object至少需要三个成员函数：
    - `__init__`：定义了初始化方式
    - `__getitem__(self, index)`：定义了获取数据的方式，一般需要实现**读取文件、格式审查**等功能。按照索引返回一个元组`(image, label)`
    - `__len__()`：获取数据集中数据数量
- 数据加载器（DataLoader）：定义了一个高性能的异步数据加载器，可以实现多进程读取数据、mini-batch等功能。一般来说，Dataloader不需要我们手动实现，所以尽可能了解官方的接口功能即可。感兴趣的同学可以阅读Dataloader的源码。

在本例中，你需要完整阅读源代码，学习dataset、dataloader的写法。同时，完成代码中的留空部分，即利用PyTorch中模块接口实现一个简单的CNN模型。你需要实现的模型设计如下：

```
(input) - Conv - ReLU - FC - (output)
```

其中，卷积的参数为：

- 输入通道：1
- 输出通道：8
- 卷积核尺寸：3*3
- 步长：1

> 注意：要通过计算得出FC的输入尺寸和输出尺寸。

最后你可以运行本项目，训练出一个可以识别大多数手写体数字的分类器。

简单可用的AI（50pts）

> 你将通过此案例，自己动手完成一个简单的PyTorch项目，在CIFAR-10数据集上训练一个简单的图像分类器。源代码文件为`./code/03.simple-ai.py`。

通过上面两个案例，你已经学会了如何基于PyTorch构建一个完整的AI项目了。接下来的时间交给你，动手在CIFAR-10数据集上训练一个图片分类器！

在本例的源代码文件中，关键的内容均已留空。这些部分都是需要你来实现的，包括：

- 模型设计与实现
- 数据集和数据加载器的实现
- 评估功能的实现
- 训练功能的实现

本案例的目的是让你对PyTorch的项目框架有一个整体的认知，因此你可以参考PyTorch官网的文档，Github上的公开仓库或前两个问题的代码，但不可以整段粘贴，或者改变代码结构。

如果你觉得从头设计一个模型比较困难，不知道从何处下手的话，可以参考下面这个方案：

```
(input) - Conv1 - ReLU - Pooling - Conv2 - ReLU - Pooling - FC1 - ReLU - FC2 - (output)
```

其中，Conv1参数为：

- 输入通道：3
- 输出通道：6
- 卷积核尺寸：5*5
- 步长：1

Conv2参数为：

- 输入通道：6
- 输出通道：16
- 卷积核尺寸：5*5
- 步长：1

Pooling均为2*2的max pooling。

> 注意，此处FC1的输入参数需要根据上述参数计算，FC2的输入参数可以自行设计。

这不是一个很好的模型，但是此模型可以作为你的start up model。接下来你需要做的是：

1. 实现本案例的全部内容，正确运行，并可以正确训练；
2. 尽可能进行改进，使测试集上的准确率达到60%以上。（60%将会作为本案例是否完成的指标）