[Problem 4] 编程题说明

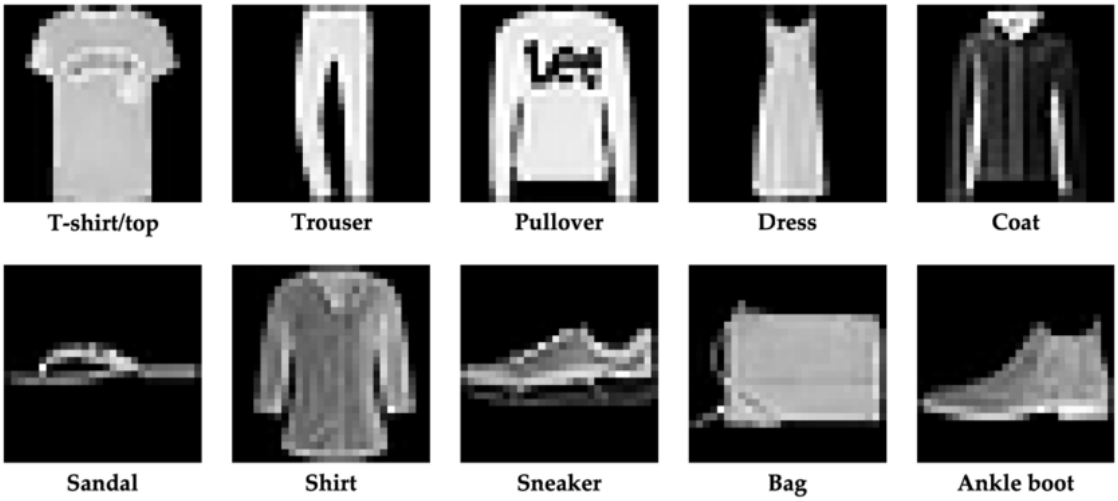
在本题中，我们尝试基于 `PyTorch` 实现一个神经网络，来完成 FashionMNIST 数据集上的图像分类任务。

FashionMNIST

FashionMNIST 是一个服装相关的图像分类 Benchmark 数据集，图像均为 28×28 单通道黑白位图，类别数为 10，完整数据集包括 60000 个训练样本与 10000 条测试样本。

为了减少对神经网络训练算力的需求，我们构造了一个 FashionMNIST 的较小子集，包含 1000 条训练样本 `train.npz` 与 1000 条测试样本 `test.npz`。此外，我们保证样本类别分布均衡：训练/测试集中每种类别的样本数量均为 100.

下图展示了 FashionMNIST 数据集中的部分图片及其对应类别。



PyTorch

PyTorch 是近年来学术界最主流的深度学习框架，在工业界亦有广泛应用。

PyTorch 可以用来进行各类神经网络的开发及部署。例如：许多神经网络相关的开源工作都提供了基于 PyTorch 编写的代码。此外，目前的开源大语言模型（LLM）也常常通过提供 PyTorch 模型权重文件的方式来分发。因此，对于希望在未来的学习与工作中利用深度学习模型，解决实际问题的同学们而言，了解甚至掌握 PyTorch 的使用方法是极为重要的。

> 我们计划第六次作业将采用大作业或 Kaggle 竞赛的形式，届时你可能需要利用机器学习技术解决一个较为复杂的现实问题。我们不会限制你使用何种模型，但很大一部分同学可能会考虑利用 PyTorch 实现深度神经网络，因此建议同学们利用这个机会，了解 PyTorch 的安装、配置方法及基本使用方式。

为了考察大家查询教程文档、解决问题的能力，此次我们不会提供详细的环境配置流程。你可以参考 PyTorch 的官方安装教程，尝试将 PyTorch 安装到你的本地 Python 环境中；或者，你也可以使用 Google Colab 等预先配置好 PyTorch 的在线编程环境，编写并运行你的代码。

> 请确保安装 2.0 以上版本的 PyTorch（我们的测试环境采用最新的 Stable 版本：2.2.2）
>
> 此外，本次作业的模型不会对算力有较大要求 —— 你可以自由选择安装 CPU 或 GPU 版本的 PyTorch，但 GPU 版本一般需要额外配置 CUDA，安装可能会更麻烦一些。该选择不会影响你实际编写的代码 —— 同一份 PyTorch 代码一般在 CPU 或 GPU 加速的环境下均可正常运行。

我们提供四个代码模板文件：`p4_models.py`，`p4_trainer.py`，`p4_utils.py` 与 `p4_main.py`，其中：

- `p4_models.py` 包含了神经网络分类器模型的定义，**需要在压缩包中提交**；
- `p4_trainer.py` 包含了神经网络训练相关的代码，**不强制提交**，但**若不提交则视为放弃题目 4.3**；
- `p4_utils.py` 包含了加载数据集，绘图等一些辅助功能的代码，**不强制提交**
- `p4_main.py` 为入口脚本，**不强制提交**

请不要更改脚本的文件名——我们会忽略压缩包的目录结构，仅根据文件名来判断文件的内容。

[4.1 - 10pts] 实现神经网络

> 在实现的过程中，你可能会希望参考 PyTorch 官方的 Tutorial，Documentation 或互联网上的其他资料。

请在 `p4_models.py` 中补全 `FashionClassifier` 类的 `__init__()` 及 `forward()` 相关代码，以实现一个结构如下的**全连接神经网络**：

> 其中：输入为 $1 \times 28 \times 28$ 的单通道位图数据，经 `nn.Flatten` 转化为 784 维的向量，即：输入层包括 784 个神经元，仅接受输入，不进行函数处理。

- 隐层 1：神经元个数为 512 个，使用 ReLU 作为激活函数
- 隐层 2：神经元个数为 256 个，使用 ReLU 作为激活函数
- 隐层 3：神经元个数为 128 个，使用 ReLU 作为激活函数
- 输出层：神经元个数为 10 个，不使用激活函数

你需要在 `__init__()` 中实现神经网络各层参数的定义，并在 `forward()` 中实现网络的前向传播过程。反向传播过程则无需编写 —— PyTorch 的自动微分机制会自动进行梯度计算。

> 提示：这部分代码量很小。如果你的实现正确，应当新增 10 行左右的代码就可以实现全部功能。不妨对比第三题中，我们实现一个简单神经网络的代码行数 —— 使用深度学习框架可以极大地减少模型开发的工作量。

实现完成后，运行 `p4_main.py`，控制台中会输出训练过程中每个 epoch 的训练 loss，测试 loss 及测试准确度，并最终生成 `plot.png` 描述训练过程中这些量的变化情况。

请确保你的代码可以正常运行。只要你的实现正确，满足基本的精度要求，且作业文档中展示的 `plot.png` 符合我们的预期，即可获得本题的全部分数。

[4.2 - 10pts] 缓解过拟合

从 (1) 中生成的训练过程图片 `plot.png` 中可以看出：模型明显出现了**过拟合**，即训练一定轮次后，训练集 loss 持续下降，但测试集 loss 保持不变或转为上升。

事实上，由于我们的训练集样本数较少，且 4.1 中的神经网络结构较为复杂，此时很容易出现过拟合的现象。因此，请提出**至少两种**缓解过拟合的方法，分别通过编程实现后，介绍并分析其效果。

对于每一种方法，你需要在文档中汇报如下内容：

- [2pts] 选择了哪一种方法，且大致是如何实现的
- [1pts] 展示应用该方法后的 `plot.png`
- [2pts] 结合图片，分析该方法有效（或无效）的原因

[4.3 - 5pts] （附加题）模型改进

本题需要你探索对神经网络结构及训练的方法的最优改进方案，使得模型取得最好的分类效果。

可选择的改进方法**包括但不限于**以下几种：

- 设计结构更复杂的神经网络（如增加卷积层，池化层等）
- 换用其他损失函数
- 使用其他优化器或优化方法
- 设计合适的 Validation 方法
- 使用 Dropout, Batch Normalization, 正则化等技术保证模型的收敛性质

具体的，你需要实现 `p4_models.py` 中的 `BetterFashionClassifier` 类，与 `p4_trainer.py` 中的 `BetterTrainer` 类。我们将使用同样的训练集，但在一组未公开的测试集上测试模型的分类准确率。测试时使用的代码如下：

```
from p4_models.py import BetterFashionClassifier
from p4_trainer.py import BetterTrainer

# Preparing datasets and others, identical to `p4_main.py` ...

model = BetterFashionClassifier().to(device)
trainer = BetterTrainer(model, train_dataloader, test_dataloader, device)

for t in range(trainer.MAX_EPOCH):
    trainer.train_step()

# Now we would test your `model` with our test function :)
```

请确保你的模型可以被上述代码所正常调用。

假设共有 N 名同学完成 4.3（即：`BetterFashionClassifier` 与 `BetterTrainer` 可以正常调用，且模型分类精度大于 0），我们将这 N 名同学模型的测试集分类准确率由高到低排列，对前 $K = \min\left(\lfloor N/10 \rfloor, 10\right)$ 名同学提供附加题分数奖励。对于序号为 i 的同学（$1 \leq i \leq k$），奖分为：$5 - \lfloor 5(i-1)/k \rfloor$.