在 kaggle 手写数字识别任务上进行

第一次任务

对该竞赛的代码，训练 与 推理 分开编写 (训练在一个notebook中，推理在另一个notebook中)

训练 notebook 流程可参考如下:

1. 导入相关库 (Import Libs)
2. 定义全局参数 (CONFIG)
    - 随机种子 (seed)
    - 训练轮数 (epochs)
    - 训练和验证的批次大小 (batch_size)
    - 学习率 (learning_rate)
    - 其他参数...
3. 固定随机种子 (Set Random Seed)
    - 固定随机种子，方便结果的复现
4. 处理数据 主要是进行读取等操作 (Data Progress)
    - 本次任务将 train.csv 中前 80% 的数据作为训练集，后 20% 的数据作为验证集
5. 对 Dataset and DataLoader 的编写 (Dataset and DataLoader)
    - 可以将 DataLoader 的操作封装到函数里，方便操作
6. 评估函数 用于计算本地分数 (Evaluation)
    - 此学习赛的评估函数为 Acc， 其他比赛要具体比赛具体分析
7. 定义模型 (Model)
    - 本次任务要求 使用 3层的 MLP 神经网络模型
    - 即 输入形状为 (batch_size, 784) 进行 10 分类任务后， 模型输出形状为 (batch_size, 10)
8. 对训练和验证的函数进行编写 (Train and Valid Function)
    - 将训练和验证过程编写成函数，后面其他比赛可以直接参考，修改细节部分就可运用到其他比赛
        - 可将训练一轮编写成一个函数
        - 可将验证一轮编写成一个函数
        - 编写第3个函数来将前两者组合在一起，在训练过程中根据在验证集上的 cv 分数选择是否保存模型权重
        - 保存模型时只保存权重文件 (torch.save(model.state_dict(), PATH))
    - 此次训练 损失函数选择 nn.CrossEntropyLoss()
9. 优化器 (Optimizer)
    - 优化器选择 Adam (torch.optim.Adam())
10. 开始训练 (Start Training)
11. (选做) 可将训练过程中的 loss CV(Acc) lr 等参数的变化通过可视化的方式表现出来

推理 notebook 流程可参考如下:

1. 导入相关库 (Import Libs)
2. 定义全局参数 (CONFIG)
    - 随机种子 (seed)
    - 推理的批次大小 (batch_size)

    - 其他参数...
3. 固定随机种子 (Set Random Seed)
    - 一般的推理过程不涉及随机种子的影响，以防一些特殊情况最好也将随机种子固定下来
4. 处理数据 主要是进行读取等操作 (Data Progress)
    - 读取 test.csv
5. 对 Dataset and DataLoader 的编写 (Dataset and DataLoader)
    - 要求同训练，不同的点在于 推理过程没有 label 这一列，可用其他变量代替
6. 定义模型 (Model)
    - 模型要求同训练
7. 加载在训练中得到的cv得分最高的权重文件 (Load Model)
8. 对推理的函数进行编写 (Infer Function)
    - 返回全部的预测结果
9. 开始推理 (Start Infer)
10. 创建提交的 .csv 文件 (Make Submission)
    - 按竞赛要求 创建相同的 submission.csv