实验项目结构概述

202218018670003 马天行

00 概述

这个文档是下面四个实验报告的预置内容，主要介绍了一下实验项目文件结构和实验流程。

整体代码可以到仓库查看：https://github.com/Noitolar/CourseDL.git

01 文件结构

- checkpoints：保存的模型权重
  - mnist.conv.pt
  - dogs_vs_cats.resnet18.pt
  - poem(640).lstm.pt
  - movie.conv.pt
- datasets：数据集
  - mnist
  - dogs_vs_cats
  - poem(640)
  - movie
- logs：实验日志
  - mnist.conv.log
  - dogs_vs_cats.resnet18.log
  - poem(640).lstm.log
  - movie.conv.log
- utils：工具类和工具函数
  - __init__.py
  - cv：前两个实验用到的
    - __init__.py
    - config.py：实验配置
    - dataset.py：数据集处理
    - handler.py：模型操作
    - nnmodels.py：神经网络模型

        - recorder.py：实验记录
        - trainer.py：模型训练和评估
    - nlp：后两个实验用到的
        - __init__.py
        - config.py
        - dataset.py
        - handler.py
        - nnmodels.py
        - recorder.py
        - trainer.py
- task_01.py：实验一
- task_02.py：实验二
- task_03.py：实验三
- task_04.py：实验四

02 实验流程

由于面对对象式的编程风格，四个实验的实验流程基本一致，包括但不限于：

- 初始化实验配置：
    - 神经网络模型类名
    - 神经网络模型参数
    - 设备
    - 损失函数类名
    - 损失函数参数
    - 日志文件路径
    - 数据集名称
    - 随机种子
    - 训练集预处理（CV）
    - 验证集预处理（CV）
    - 批大小
    - 迭代轮数
    - 优化器类名
    - 优化器参数
    - 学习率调整策略类名

    - 学习率调整策略参数
    - 权重文件路径
- 基于配置初始化模型Handler
- 将配置内容记录在日志
- 基于配置初始化数据集和加载器
- 基于配置初始化训练器
- 进行对应轮次的训练和验证/生成，并将结果记录在日志中

下面以task_01.py为例：

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.utils.data as tdata
import torchvision.transforms as trans
import utils.cv as ucv

if __name__ == "__main__":
    # 实验配置
    config = ucv.config.ConfigObject()
    config.model_class = ucv.nnmodels.SimpleConvClassifier
    config.model_params = {"num_classes": 10, "num_channels": 1}
    config.device = "cuda:0"
    config.criterion_class = nn.CrossEntropyLoss
    config.criterion_params = {}
    config.log_path = "./logs/mnist.conv.log"
    config.dataset = "mnist"
    config.seed = 0
    config.trn_preprocess = trans.Compose([trans.ToTensor(), trans.Normalize(mean=[0.485], std=[0.229])])
    config.val_preprocess = config.trn_preprocess
    config.batch_size = 32
    config.num_epochs = 8
    config.optimizer_class = optim.SGD
    config.optimizer_params = {"lr": 0.001, "momentum": 0.9, "nesterov": True}
    config.scheduler_class = optim.lr_scheduler.MultiStepLR
    config.scheduler_params = {"milestones": [4, 6], "gamma": 0.1}
    config.checkpoint_path = "./checkpoints/mnist.conv.pt"

    # 初始化模型Handler
    handler = ucv.handler.ModelHandlerCv(config)

    # 将配置录入日志
    handler.log_config()
```

```
    # 初始化数据集和加载器
    trn_set, val_set = ucv.dataset.get_dataset(config)
    trn_loader = tdata.DataLoader(trn_set, batch_size=config.batch_size, shuffle=True)
    val_loader = tdata.DataLoader(val_set, batch_size=config.batch_size * 8)

    # 初始化训练器
    trainer = ucv.trainer.Trainer(handler)

    # 训练并记录结果
    best_val_accuracy = 0.0
    for epoch in range(config.num_epochs):
        handler.log("    " + "=" * 40)
        trn_report = trainer.train(trn_loader)
        handler.log(f"    [{epoch + 1:03d}] trn-loss: {trn_report['loss']:.4f} --- trn-acc:
{trn_report['accuracy']:.2%}")
        val_report = trainer.validate(val_loader)
        handler.log(f"    [{epoch + 1:03d}] val-loss: {val_report['loss']:.4f} --- val-acc:
{val_report['accuracy']:.2%}")
        if val_report["accuracy"] > best_val_accuracy:
            best_val_accuracy = val_report["accuracy"]
            if config.checkpoint_path is not None:
                torch.save(handler.model.state_dict, config.checkpoint_path)
    handler.log(f"[=] best-val-acc: {best_val_accuracy:.2%}")
```