分支管理

由于需要在多个训练集上进行略有区别的网络的训练，故使用多个分支进行管理。
分支的命名规则如下： 使用 `fully_quantum` 前缀的均为全量子模型，使用 `hybrid` 前缀的均为经典-量子混合模型。

对于 `fully_quantum`（全量子模型），默认为 2 分类器。对于 `hybrid` （经典-量子混合模型）而言，默认为 10 分类器，带有 `binary` 标记的为 2 分类器，带有 `quaternary` 标记的为 4 分类器。

使用数据集子集训练得到的 2 分类器 或 4 分类器，默认使用 MNIST --> USPS 数据集（USPS 数据集过小，裁剪后已不适合作为 源域 使用）；若带有 `rgb` 标记的则为使用数据集 MNIST --> mnist_m 的模型，目标域为彩色图片。

10 分类器默认使用 USPS --> MNIST 数据集；若带有 `mnist2mnistm` 标记的则为使用数据集 MNIST --> mnist_m 的模型，目标域为彩色图片；若带有 `hybrid_usps2mnistm` 标记的则为使用数据集 USPS --> mnist_m 的模型，目标域也为彩色图片。

代码结构

代码入口

- **main.py**：训练入口
- **test.py**：测试入口（在整个测试集进行预测并输出准确率）
- **demo.py**：展示入口（随机小样本测试并可视化地显示结果）

定义模型（经典-量子混合）

- **model.py**：主模型文件，定义了模型的主结构
- **class_qmodel.py**：标签分类器，包含调用量子层的代码，和输入量子层前的预处理、量子层输出后的后处理
- **domain_qmodel.py**：域分类器，包含调用量子层的代码，和输入量子层前的预处理、量子层输出后的后处理
- **quantum_net.py**：提供量子层的创建函数

定义模型（全量子）

- **model.py**：主模型文件，定义了模型的主结构
- **quantum_net.py**：提供量子层的创建函数

量子层

量子电路在各平台的实现分别编写在以下文件：

- **pypennylane_fc_layer.py**：基于 Pennylane 框架的量子全连接（FC）层实现（PyTorch 层）
- **pennylane_feature_layer.py**：基于 Pennylane 框架的QCNN实现（PyTorch 层）
- **guodun_fc_layer_executor.py**：基于国盾平台的量子全连接层（FC）电路实现
- **guodun_feature_layer_executor.py**：基于国盾平台的QCNN实现
- **qiskit_fc_layer_executor.py**：基于 Qiskit 框架的量子全连接（FC）电路实现
- **qiskit_feature_layer_executor.py**：基于 Qiskit 框架的QCNN实现

quantum_net_via_executor.py 文件提供了一个量子层包装器，用于将为国盾 / Qiskit 平台编写的量子电路（xxx_executor.py）包装为一个层（`torch.nn.Module`）。

辅助文件 **quantum_net.py** 提供了一个统一的分发结构，便于在 Pennylane / 国盾 / Qiskit 间进行切换，其内容大致如下：

```
if current_quantum_backend == QuantumBackend.Pennylane:
    # import ...
    def QuantumFCLayer():
        return PennylaneFCLayer()
elif current_quantum_backend == QuantumBackend.GuoDun:
    # import ...
    # token = ...
    fc_executor = GuoDunFCLayerExecutor(account, 2, 4)
    def QuantumFCLayer():
        return QuantumNetLayerViaExecutor(fc_executor)
elif current_quantum_backend == QuantumBackend.Qiskit:
    # import ...
    # token = ...
    fc_executor = QiskitFCLayerExecutor(2, 4, backend)
    def QuantumFCLayer():
        return QuantumNetLayerViaExecutor(fc_executor)
```

数据加载

- **dataset_transform.py**：统一定义了源域和目标域所使用的数据集（对象），供 `main.py`、`test.py`、`demo.py` 等文件使用。
- **data_loader.py**：定义了数据加载器（类），用于加载数据集。**dataset_transform.py** 会首先拼接路径，然后在加载数据集时调用该文件中的函数。

工具函数

- functions.py：定义了一些辅助的可微函数（`torch.autograd.Function`），如梯度反转算子（`ReverseLayerF`）
- **qcis_builder.py**：定义了 QCIS 指令构造器，便于使用 QCIS 指令集（在国盾平台上）构造量子电路而不必时刻进行大量的字符串拼接操作

执行流程

修改 main.py 可切换是否开启领域自适应：

```
domain_adaptation_enabled = True
```

- 通过运行 `main.py` 进行训练
- 通过运行 `test.py` 可进行测试集的全量测试，并得到准确率
- 通过运行 `demo.py` 可进行可视化的效果展示

运行 `test.py` 和 `demo.py` 时，程序将扫描 models 文件夹下的 *.pt 文件，并提示用户选择使用其中的具体某个模型：

> Found the following files of model parameters:
>
> 1. hybrid_best.pt
> 2. hybrid_current.pt
>
> Choose a file (enter the corresponding number):

请选择您需要使用的模型文件进行测试，通常为带有 best 的那个