AI编程 HW3 作业报告

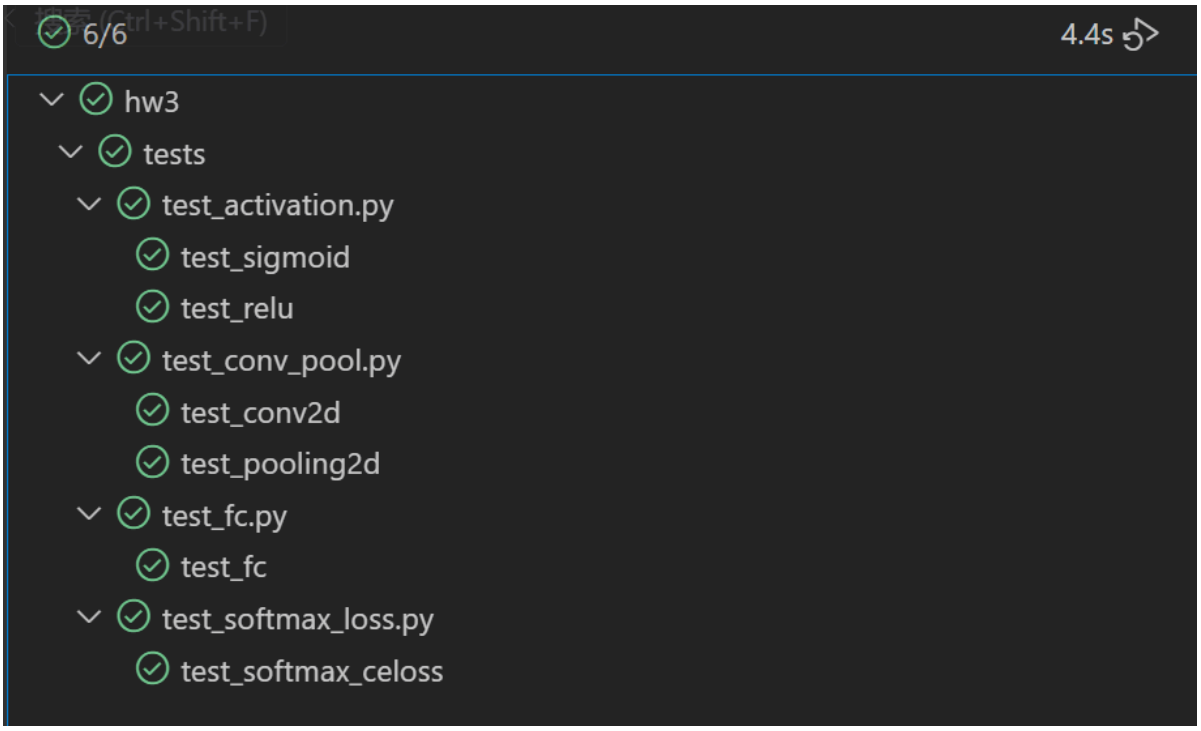
2200013212 吴童 CUDA version=12.0

代码框架

- 本文件为实验报告
- `csrc/` 为CUDA源码
  - `bind.cu` 中用pybind11将接口暴露给Python，便于使用和测试
  - `ndarray.cu` 中定义 `NDArray` 类和相关方法
  - `fn.cu` 中实现作业要求的神经网络层的前向和反向传播算子
- `tensor.py` 中实现了 `NDArray` 的子类 `Tensor`
  - 实现了更加user-friendly的接口，如：
    - `__repr__`, `__str__` 等函数
    - 相关属性如 `shape`, `size`, `device`, `ndim`
    - 成员函数进一步封装，包括类型和参数检查等
- `nn.py` 中在 `Tensor` 基础上进一步封装了 `fn.cu` 中的算子，对每个算子实现 `nn.Module` 的子类，并实现 `forward()` 和 `backward()`
- `tests/` 为pytest构建的对7个CUDA算子封装而成的nn.Module单元测试。与 `torch.nn.functional` 的计算结果进行比较，验证正向和反向传播的计算的正确性
- `load_mnist.py` 中 `load_mnist()` 能用torchvision下载MNIST数据集，并转换为 `tensor.Tensor` 格式返回
- `utils.py` 中实现了一些工具函数
- `env.yaml` 为conda环境文件
- `setup.py` 为打包文件,**正确build CUDA extension需要在其中手动修改include dir**(详见该文件)

设计思路

- 我选择用CUDA实现NDArray类, 然后在Python中实现其子类Tensor, Tensor类不仅具有autodiff功能(尚未实现), 而且提供更为便利的用户接口和函数调用。
- `ndarray.cu` 中定义了 `NDArray` 类, 并实现常用操作。在hw2基础上:
  - 实现了以 `numpy.ndarray` 为参数的构造函数
  - 实现rand函数，通过curand采样U(0, 1)随机NDArray，然后将其map到任意闭区间
- `tensor.py` 中 `Tensor` 类继承自 `NDArray`，这样既能做进一步封装和重载，又无需修改hw2中实现的CUDA operators(其参数为NDArray)
  - `Tensor` 类还提供多种初始化方式及和 `torch.Tensor`， `numpy.ndarray` 之间便捷的转换
  - 封装了一些 `NDArray` 的方法，比如 `T()`，`reshape()`，`rand()` 并提供类型检查
- `nn.py` 实现了不同的神经网络module，并分别实现其前向和反向传播函数，可以用于方便地构建网络结构，记录激活值及后续实现自动微分。
- `tests/test_*.py` 中将 `nn.py` 中定义的各NN layer与 `torch.nn.functional` 的计算结果进行比较，验证正向和反向传播的计算的正确性。测试结果如下：



运行方式

环境配置

```
conda create -n aip -f env.yaml
conda activate aip
```

其他依赖：（需要在setup.py中手动修改相关package路径）

- thrust
- cublas
- pybind11
- curand?

构建CUDA拓展

```
python setup.py develop
```

用法

- 调用NDArray

```
from backend import NDArray, Device, fn
```

- 调用Tensor(继承自NDArray)

```
from tensor import Tensor
```

- 调用nn layers

```
from nn import Linear, Conv2d ...
```