人工智能中的编程第三次作业

Oct 24th, 2024

1. 作业任务

- 使用 Pybind11 将第一次作业实现的 Tensor 类封装进 Python 中，并能在 Python 侧实例化你的 Tensor.
- 将前两次作业中实现的 7 种卷积神经网络中常用的 Module(Sigmoid，ReLU，Fully Connected Layer，Convolution Layer，Pooling Layer，Cross Entropy Loss，SoftMax) 封装进 Python 中，并支持在 Python 中传入你的 Tensor 并实现输入输出.
- 在 Python 中，自己选择合适的方法读取 MNIST 数据集，然后转换为 numpy 数组，并使用 pybind11/numpy.h 中的 pybind11::array_t 转换为自己的 Tensor，以便之后使用自己的 Tensor 训练 MNIST 数据集.
- 在 python 中对封装好的 7 个算子进行测试，并与 Pytorch 中 torch.nn.functional 提供的标准函数的计算结果进行 UnitTest(单元测试)，以尽可能减少在后续实现模块拼接和计算图构建时进行不必要的繁杂的 debug。

2. 评分标准

本次作业占课程总评的 10 分，截止日期为 2024 年 11 月 26 日 23:59，每迟交一天扣 1 分，扣完为止.

3. 其他事项

本次作业难度不大，但是需要同学们多查阅相关资料和文档，积极解决报错. 如果遇到问题，可以在课程文档的问答墙中寻找相关问题或向助教提问，助教将在不超过 24 小时内回复.

4. 实验指导及常见问题

- 如何使用 Pybind11?

你可以使用 cmake 来进行 pybind11 的绑定过程，也可以使用 python 的 setuptools 来构建模块，使用 python setuptools 时，由于项目包含 CUDA 代码，无法直接使用 Pybind11Extension，因此可以用 Pytorch 的 CUDAExtension 编译.

setup.py 定义 Pybind11 如何编译你的模块，示例如下:

```
import os
from setuptools import setup, find_packages
from torch.utils.cpp_extension import BuildExtension, CUDAExtension
__version__ = '0.0.1'
sources = ['src/pybind.cpp']
setup(
    name='mytensor',
    version=__version__,
    author='your_name',
    author_email='your_name@pku.edu.cn',
    packages=find_packages(),
    zip_safe=False,
    install_requires=['torch'],
    python_requires='>=3.8',
    license='MIT',
    ext_modules=[
```

```
        CUDAExtension(
            name='mytensor',
            sources=sources)
    ],
    cmdclass={
        'build_ext': BuildExtension
    },
    classifiers=[
        'License :: OSI Approved :: MIT License',
    ],
)
```

src/pybind.cpp 定义对你的模块对 Python 暴露的内容, 示例如下:

```
#include "tensor.h"
#include<pybind11/pybind11.h>
#include<pybind11/stl.h>
namespace py = pybind11;
PYBIND11_MODULE(mytensor, m) {
    py::class_<Tensor>(m,"Tensor")
    .def(py::init<const std::vector<int> &, const std::string &>())
    .def("size", &Tensor::size)
    .def("set_data", &Tensor::set_data)
    .def("cpu", &Tensor::cpu)
    .def("gpu", &Tensor::gpu)
    .def("print_data", &Tensor::print_data);
}
```

在项目文件夹下运行 python3 setup.py develop 即可得到动态链接库, 然后在同目录下新建 py 脚本, 即可 import mytensor. 需要注意, 如果借助 Pytorch, 那么要先 import torch 才能导入 mytensor, 否则会因依赖找不到而报错.

以上给出了一种方案, 助教也会在课程文档继续更新其他方案.

- Python 中绑定 Tensor 时要注意什么?

建议使用智能指针, 方便内存管理. 如果你不想使用智能指针, 请将复制构造函数写成深拷贝, 防止重复析构. (这部分内容可见课程文档问答墙)

- 如何读取 MNIST?

你可以在 MNIST 官网查询 MNIST 文件格式, 然后使用 struct 库读取. 如果不想用 struct 库, 可以用现成的 torch 或者 cv2 或者别的库读取.

- 还有别的问题?

可以关注一下课程文档!