Warmup实验报告

姓名：李俊哲

学号：2301213259

实验环境

软件环境：CUDA12.1 + pytorch2.1.0

硬件环境：NVIDIA A100 80GB PCIe

实验结果

实现描述

CUDA编程

由于实验要求只需实现 normlized_shape 为最后一维 dim size 且 elementwize_affine=False 的情况，所以只需要对tensor中的每个行向量在列维度上实现层归一化即可。

CUDA代码分为核函数（mylayerNorm_kernel）和主函数（mylayerNorm_cuda_forward）。

核函数输入分别为矩阵A、矩阵B、矩阵的维度M与N以及一个很小的eps保证分母不为0。核函数每个CUDA线程处理A的一行。首先，计算这一行的均值。接着，计算这一行的方差。最后，使用计算得到的均值和方差对这一行的数据进行归一化，并将结果存储到B中。

主函数输入是张量input，输出是归一化的张量。首先，使用`torch::zeros_like`函数在CUDA设备上为输出张量分配空间。接着，定义CUDA的线程块和网格大小，每个线程块包含256个线程，网格大小基于输入的M维度计算。使用`AT_DISPATCH_FLOATING_TYPES`宏确定输入的数据类型，并调用相应的核函数。

C++代码封装

C++代码的目标是为CUDA实现的LayerNorm提供C++接口，并确保传入的Tensor满足必要的条件（如为CUDA Tensor、连续存储等）。

使用`TORCH_CHECK()`保证传入的Tensor是CUDA Tensor并且是连续存储。

使用`PYBIND11_MODULE`宏定义一个扩展模块，使Python可以调用这些C++函数。

编译与安装

使用`setuptools`进行编译和安装。

在编译过程中，制定了额外的编译参数，设置了C++编译标准为C++17。使用pytorch提供的`BuildExtension`类来构建和编译C++和CUDA扩展。

Python代码封装

编写`myLayerNormFunciton`类并继承`torch.autograd.Function`类，来调用自定义的算子。并通过注解`@staticmethod`定义前向传播`forward`静态函数。

编写`myLayerNorm`类继承`torch.nn.Module`类，并重构`forward`方法，使用`myLayerNormFunction.apply(input)`进一步调用。

正确性比较

和`pytorch.nn.LayerNorm`进行比较，使用`np.testing.assert_allclose()`进行100次比较，设置`rtol=1e-3`、`atol=1e-5`，其中测试输入的大小为`size=(64, 128)`。

结果没有assert，说明在设定的误差范围内正确。

性能比较

进行100次测试，并对运行时间求平均，结果如图所示：

```
(pinns-gpu-env) [2301213259@l12gpu07 profile]$ python custom_layerNorm.py
My LayerNorm forward avg time: 5.429980087280274e-05s
PyTorch LayerNorm forward avg time: 3.206014633178711e-05s
```

因此前向传播的性能为5.43e-05秒/次。

选做任务

选做任务1：反向传播

反向传播的编程实现类似于前向传播过程。首先是编写CUDA的核函数和主函数，然后编写python可以调用的C++接口，接着编译和安装，最后在使用python代码封装继承`torch.nn.Module`类使之在`loss.backward()`时可以自动进行反向传播。

特别需要说明CUDA代码中的反向传播的实现过程：
令

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{1}$$

所以

$$d\hat{x}_i = \frac{\partial L}{\partial \hat{x}_i} = grad\_output[i] \tag{2}$$

然后计算方差的梯度$d\sigma^2$和均值的梯度$d\mu$

$$d\sigma^2 = \sum_i (x_i - \mu) * (-\frac{1}{2}) * (\sigma^2 + \epsilon)^{-\frac{3}{2}} * d\hat{x}_i \tag{3}$$

$$d\mu = \sum_i d\hat{x}_i * (-1) * \frac{1}{\sigma} + d\sigma^2 * (-2) * \frac{1}{N} \sum_i (x_i - \mu) \tag{4}$$

最后计算输入x的梯度

$$dx_i = d\hat{x}_i * \frac{1}{\sigma} + d\sigma^2 * \frac{2}{N} * (x_i - \mu) + d\mu * \frac{1}{N} \tag{5}$$

由于实验要求只需实现 normlized_shape 为最后一维 dim size 且 elementwize_affine=False 的情况，所以反向传播只需要返回输入x的梯度即可。

在测试代码中，定义损失函数为`torch.nn.MSELoss()`，作用在LayerNorm计算后的结果上，然后执行反向传播，对自定义算子和pytorch的算子进行正确性和性能比较。

使用`np.testing.assert_allclose()`进行100次比较，比较loss对输入的tensor的梯度，设置`rtol=1e-3`、`atol=1e-5`，发现没有assert，说明在设定的误差范围内是正确的。

性能测试依然采用运行100次求平均的方式，结果如下：

```
My LayerNorm backward avg time: 0.00012686014175415044s
PyTorch LayerNorm backward avg time: 9.627819061279297e-05s
```

因此反向传播的性能为1.269e-04秒/次。

选做任务2：profile

使用`torch.profiler.profile`进行性能的分析。

按照cuda_total_time进行排序，打印耗时的CUDA操作如下：

前向传播

自定义算子：

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	CPU Mem	Self CPU Mem	CUDA Mem	Self CUDA Mem	# of Calls
myLayerNormFunction	5.64%	35.000us	99.19%	616.000us	616.000us	33.000us	86.84%	34.000us	34.000us	0 b	0 b	32.00 Kb	0 b	1
void myLayerNorm_kernel<float>(float const*, float*,...	0.00%	0.000us	0.00%	0.000us	0.000us	33.000us	86.84%	33.000us	33.000us	0 b	0 b	0 b	0 b	1
Context Sync	0.00%	0.000us	0.00%	0.000us	0.000us	4.000us	10.53%	4.000us	2.000us	0 b	0 b	0 b	0 b	2
aten::zeros_like	1.13%	7.000us	92.75%	576.000us	576.000us	0.000us	0.00%	1.000us	1.000us	0 b	0 b	32.00 Kb	0 b	1
aten::zero_	0.64%	4.000us	38.65%	240.000us	240.000us	0.000us	0.00%	1.000us	1.000us	0 b	0 b	0 b	0 b	1
aten::fill_	1.77%	11.000us	38.00%	236.000us	236.000us	1.000us	2.63%	1.000us	1.000us	0 b	0 b	0 b	0 b	1
void at::native::vectorized_elementwise_kernel<4, at...	0.00%	0.000us	0.00%	0.000us	0.000us	1.000us	2.63%	1.000us	1.000us	0 b	0 b	0 b	0 b	1
aten::empty_like	0.81%	5.000us	52.98%	329.000us	329.000us	0.000us	0.00%	0.000us	0.000us	0 b	0 b	32.00 Kb	0 b	1
aten::empty_strided	52.17%	324.000us	52.17%	324.000us	324.000us	0.000us	0.00%	0.000us	0.000us	0 b	0 b	32.00 Kb	32.00 Kb	1
cudaLaunchKernel	37.04%	230.000us	37.04%	230.000us	115.000us	0.000us	0.00%	0.000us	0.000us	0 b	0 b	0 b	0 b	2

Self CPU time total: 621.000us
Self CUDA time total: 38.000us

torch算子：

```
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %     Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg       CPU Mem   Self CPU Mem      CUDA Mem  Self CUDA Mem    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                           Context Sync       0.00%       0.000us       0.00%       0.000us       0.000us       4.000us      57.14%       4.000us       2.000us           0 b           0 b           0 b           0 b             2
                                          aten::layer_norm       0.83%       5.000us      99.17%     598.000us     598.000us       0.000us       0.00%       3.000us       3.000us           0 b           0 b      32.50 Kb        -512 b             1
                                    aten::native_layer_norm      58.87%     355.000us      98.34%     593.000us     593.000us       3.000us      42.86%       3.000us       3.000us           0 b           0 b      33.00 Kb           0 b             1
void at::native::(anonymous namespace)::vectorized_l...       0.00%       0.000us       0.00%       0.000us       0.000us       3.000us      42.86%       3.000us       3.000us           0 b           0 b           0 b           0 b             1
                                               aten::empty       2.99%      18.000us       2.99%      18.000us       6.000us       0.000us       0.00%       0.000us       0.000us           0 b           0 b      33.00 Kb      33.00 Kb             3
                                             cudaLaunchKernel      36.15%     218.000us      36.15%     218.000us     218.000us       0.000us       0.00%       0.000us       0.000us           0 b           0 b           0 b           0 b             1
                                                aten::view       0.33%       2.000us       0.33%       2.000us       1.000us       0.000us       0.00%       0.000us       0.000us           0 b           0 b           0 b           0 b             2
                                                 [memory]       0.00%       0.000us       0.00%       0.000us       0.000us       0.000us       0.00%       0.000us       0.000us           0 b           0 b     -32.50 Kb     -32.50 Kb             2
                                       cudaDeviceSynchronize       0.83%       5.000us       0.83%       5.000us       2.500us       0.000us       0.00%       0.000us       0.000us           0 b           0 b           0 b           0 b             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 603.000us
Self CUDA time total: 7.000us
```

反向传播

自定义算子：

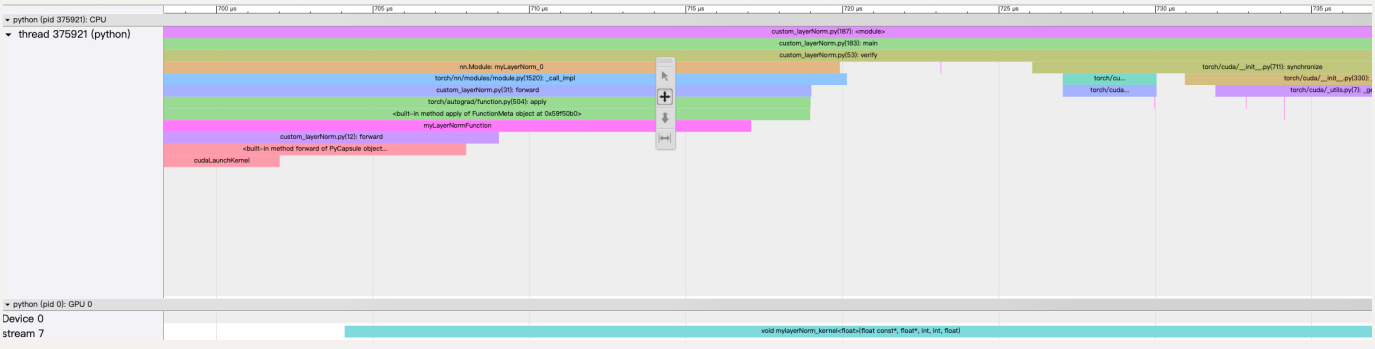
```
My LayerNorm backward avg time: 0.0032618284225463867s
Pytorch LayerNorm backward avg time: 0.0038667853355407716s
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %     Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg       CPU Mem   Self CPU Mem      CUDA Mem  Self CUDA Mem    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
autograd::engine::evaluate_function: myLayerNormFunc...       0.91%       6.000us       8.16%      54.000us      54.000us       0.000us       0.00%      60.000us      60.000us           0 b           0 b           0 b     -32.00 Kb             1
                                    myLayerNormFunctionBackward       3.93%      26.000us       7.25%      48.000us      48.000us      59.000us      84.29%      60.000us      60.000us           0 b           0 b      32.00 Kb           0 b             1
void mylayerNorm_backward_kernel<float>(float const*...       0.00%       0.000us       0.00%       0.000us       0.000us      59.000us      84.29%      59.000us      59.000us           0 b           0 b           0 b           0 b             1
                                         aten::mse_loss_backward       1.51%      10.000us       6.95%      46.000us      23.000us       3.000us       4.29%       7.000us       3.500us           0 b           0 b      32.00 Kb           0 b             2
                                                     Context Sync       0.00%       0.000us       0.00%       0.000us       0.000us       5.000us       7.14%       5.000us       2.500us           0 b           0 b           0 b           0 b             2
autograd::engine::evaluate_function: MseLossBackward...       0.91%       6.000us       6.95%      46.000us      46.000us       0.000us       0.00%       4.000us       4.000us           0 b           0 b      32.00 Kb           0 b             1
                                           MseLossBackward0       0.60%       4.000us       6.04%      40.000us      40.000us       0.000us       0.00%       4.000us       4.000us           0 b           0 b      32.00 Kb           0 b             1
                                                  aten::fill_       2.87%      19.000us      37.76%     250.000us      83.333us       3.000us       4.29%       3.000us       1.000us           0 b           0 b           0 b           0 b             3
void at::native::vectorized_elementwise_kernel<4, at...       0.00%       0.000us       0.00%       0.000us       0.000us       3.000us       4.29%       3.000us       1.000us           0 b           0 b           0 b           0 b             1
void at::native::elementwise_kernel<128, 2, at::nati...       0.00%       0.000us       0.00%       0.000us       0.000us       3.000us       4.29%       3.000us       3.000us           0 b           0 b           0 b           0 b             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 662.000us
Self CUDA time total: 70.000us
```

torch算子：

```
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %     Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg       CPU Mem   Self CPU Mem      CUDA Mem  Self CUDA Mem    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         aten::mse_loss_backward      -0.15%      -1.000us       7.57%      49.000us      24.500us       3.000us      25.00%       7.000us       3.500us           0 b           0 b      32.00 Kb           0 b             2
                                                     Context Sync       0.00%       0.000us       0.00%       0.000us       0.000us       5.000us      41.67%       5.000us       2.500us           0 b           0 b           0 b           0 b             2
autograd::engine::evaluate_function: MseLossBackward...       1.08%       7.000us       7.73%      50.000us      50.000us       0.000us       0.00%       4.000us       4.000us           0 b           0 b      32.00 Kb           0 b             1
                                           MseLossBackward0       0.62%       4.000us       6.65%      43.000us      43.000us       0.000us       0.00%       4.000us       4.000us           0 b           0 b      32.00 Kb           0 b             1
void at::native::elementwise_kernel<128, 2, at::nati...       0.00%       0.000us       0.00%       0.000us       0.000us       3.000us      25.00%       3.000us       3.000us           0 b           0 b           0 b           0 b             1
                                                  aten::fill_       2.47%      16.000us      37.56%     243.000us     121.500us       2.000us      16.67%       2.000us       1.000us           0 b           0 b           0 b           0 b             2
void at::native::vectorized_elementwise_kernel<4, at...       0.00%       0.000us       0.00%       0.000us       0.000us       2.000us      16.67%       2.000us       1.000us           0 b           0 b           0 b           0 b             2
autograd::engine::evaluate_function: NativeLayerNorm...       0.62%       4.000us       3.71%      24.000us      24.000us       0.000us       0.00%       2.000us       2.000us           0 b           0 b           0 b     -32.00 Kb             1
                                        NativeLayerNormBackward0       0.62%       4.000us       3.09%      20.000us      20.000us       0.000us       0.00%       2.000us       2.000us           0 b           0 b      32.00 Kb           0 b             1
                                  aten::native_layer_norm_backward       1.24%       8.000us       2.47%      16.000us      16.000us       2.000us      16.67%       2.000us       2.000us           0 b           0 b      32.00 Kb           0 b             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 647.000us
Self CUDA time total: 12.000us
```

Chrome浏览器可视化

使用`profile.export_chrome_trace("xxxx.json")`可以导出json文件，在Chrome浏览器中，使用chrome://tracing可以更加直观的看到堆栈调用过程和好耗时：



结论

可以看到自定义算子的性能不如torch实现的算子。在前向过程中，自定义算子和aten算子对CUDA显存的占用基本相同，自定义算子对GPU计算资源的占用比aten算子要高，但是时间却要慢于CUDA算子。

这说明计算成为了自定义算子的瓶颈，需要进一步优化自定义kernel来提高其性能，可能需要研究更多的CUDA并行策略，提高代码的并行性，并且确保有效地访问GPU内存，例如通过使用共享内存、确保连续内存访问等。

另外，可以观察到，自定义的算子函数调用次数较少，而PyTorch的版本包含更多的底层调用，这也说明pytorch进行了高度的优化。