Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт космических и информационных технологий
Кафедра «Вычислительной техники»

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2

«Изучение фреймворка PyTorch»

Преподаватель _____ С.А. Тарасов
подпись, дата инициалы, фамилия

Студент КИ20-07Б, 032049287 _____ А.С. Базаров
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023

ВВЕДЕНИЕ

Цель работы: ознакомление с фрейморков PyTorch

Задание:

1. На основе функции (kernel) из практической работы №1 создать расшиение для фреймворка PyTorch;
2. Написать программу на языке Python, которая компилирует и загружает расширение, а затем с его помощью производит вычисления;
3. Написать модульный тест, который проверяет, совпадает ли результат вычислений, полученный с помощью расширения, с результатом вычислений аналогичной математической функции, полученным с помощью стандартных методов PyTorch. Если программа верна, то значения должны совпадать с некоторой приемлемой точностью;
4. Подготовить отчет, который должен содержать исходный код расширения (1) и Python-скрипт (2).

Вариант: Сумма векторов

Ход работы

1 Реализация программы

Код программы программы на языке CUDA пресдтавлен на листинге 1.

Листинг 1 – Код программы на языке CUDA

```
%%writefile laba2.cu
#include <torch/extension.h>


__global__ void d_add(float *a, float *b, float *c, int n) {
    int i = blockDim.x * blockIdx.x + threadIdx.x;

    if (i < n) {
        c[i] = a[i] + b[i];
    }
}


#define CHECK_CUDA(x) TORCH_CHECK(x.device().is_cuda(), #x " must be a CUDA tensor")
#define CHECK_CONTIGUOUS(x) TORCH_CHECK(x.is_contiguous(), #x " must be contiguous")
#define CHECK_INPUT(x) CHECK_CUDA(x); CHECK_CONTIGUOUS(x)
#define CHECK_SIZE(x, y) TORCH_CHECK(x.is_same_size(y), #y " must be the same size
as " #x)

const int block_size = 128;


__forceinline__ int calc_grid_size(int m) {
    return (m + block_size - 1) / block_size;
}


torch::Tensor add(torch::Tensor a, torch::Tensor b) {
    CHECK_INPUT(a);
    CHECK_INPUT(b);
    CHECK_SIZE(a, b);

    auto c = torch::empty_like(a);
    int n = a.numel();
```

```
    d_add<<<calc_grid_size(n), block_size>>>(
        a.data_ptr<float>(),
        b.data_ptr<float>(),
        c.data_ptr<float>(),
        n
    );

    return c;
}


PYBIND11_MODULE(TORCH_EXTENSION_NAME, m) {
    m.def("my_add", &add, "Custom vector addition");
}
```

```
        dist.data_ptr<float>(),
        n
    );

    dist = sqrt(dist);

    return dist;
}


PYBIND11_MODULE(TORCH_EXTENSION_NAME, m) {
    m.def("d_dist", &add, "Custom vector addition");
}
```

2 Использование PyTorch

Код программы на языке Python представлен в листинге 2.

Листинг 2 – Код программы на Python

```
%%writefile laba2.py
import unittest
import torch
from torch.utils.cpp_extension import load


class LabTest(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.ext = load(
            name='my_extension',
            sources=['laba2.cu'],
            extra_cuda_cflags=['-O3'],
            extra_cflags=['-O3'],
        )

    def test_add(self):
        n = torch.randint(size=(1,), low=1, high=2048)

        x = torch.rand((n,), device='cuda')
        y = torch.rand((n,), device='cuda')
        z = LabTest.ext.my_add(x, y)

        z_ = x + y
```

```
        self.assertTrue(torch.allclose(z, z_, atol=1e-7, rtol=1e-6))


if __name__ == '__main__':
    unittest.main()
```

```
%run laba2.py
```

```
!cat laba1.cu
```