Практическая работа №6

ст. преп. каф. ВпВ ИКИТ СФУ Тарасов С. А.

Цель работы

Сформировать навыки разработки `CUDA`-ядер, совместимых с `PyTorch`.

Задание

1. Разработать кёрнел `hsys::nn::functional::kernels::conv2d` (см. листинг **??**), который вычисляет пакетную многоканальную двумерную свертку (кросс-корреляцию):

$$(\mathbf{X} \star \mathbf{W})[n, c_{out}, h_{out}, w_{out}] = \sum_{c_{in}=0}^{C_{in}-1} \sum_{r=0}^{k-1} \sum_{q=0}^{k-1} \hat{\mathbf{X}}[n, c_{in}, h_{out} \cdot s + r \cdot d - p, w_{out} \cdot s + q \cdot d - p] \cdot \mathbf{W}[c_{out}, c_{in}, r, q] \tag{1}$$

$$\hat{\mathbf{X}}[n, c_{out}, h, w] = \begin{cases} \mathbf{X}[n, c_{out}, h, w], & 0 \leq h < H_{in} \; and \; 0 \leq w < W_{in} \\ 0, \; otherwise \end{cases} \tag{2}$$

   Где

   - $\mathbf{X}$ — массив размера $[N, C_{in}, H_{in}, W_{in}]$ (пакет входных изображений),
   - $\mathbf{W}$ — массив размера $[C_{out}, C_{in}, k, k]$ (набор ядер/весов свертки),
   - N — размер пакета (`batch_size`),
   - C_{in} — количество каналов входного изображения,
   - H_{in} — высота входного изображения,
   - W_{in} — ширина входного изображения,
   - C_{out} — количество каналов выходного изображения,
   - k — размер квадратного фильтра,
   - s — шаг свертки (`stride`),
   - p — отступ (`padding`),
   - d — коэффициент расширения (`dilation`).

2. Используя фреймворк `Google Test`, разработать модульные тесты для `hsys::nn::functional::kernels::conv2d`. Тестовые сценарии должны покрывать параметры, приведённые в табл. **??**. Корректность вычислений проверять путем сравнения с эталонной реализацией

Листинг 1: Декларация conv2d

```
template <class T>
using view_4d_t = torch::PackedTensorAccessor32<
    T, 4, torch::RestrictPtrTraits
>;

template <
    class AtomT,
    long filter_size,
    long stride = 1,
    long padding = 0,
    long dilation = 1,
    long block_size = 16
>
__global__ void conv2d(
    view_4d_t<AtomT> output,
    const view_4d_t<AtomT> input,
    const view_4d_t<AtomT> weights
);
```

`torch::nn::functional::conv2d`, используя функцию `torch::allclose` с допусками `atol` = `rtol` = 10^{-5} (`float`). Если вы используете типы пониженной точности, то допуск необходимо увеличить; например, для `half`: `atol` = `rtol` = 10^{-2}.

Таблица 1: Параметры для тестирования (декартово произведение)

k	s	p	d
3	2	1	1
3	1	1	1
7	2	3	1
1	2	0	1
3	1	2	2
3	2	2	2

$\times$

C_{in}	H_{in}	W_{in}	C_{out}
3	244	244	64
64	112	112	64
64	56	56	64
128	28	28	128
64	56	56	128
128	28	28	256
256	14	14	512
256	14	14	256
512	7	7	512

3. Подготовить отчёт, содержащий:
    - ключевые фрагменты кода;
    - ссылку на репозиторий с полной реализацией.

Критерии оценки

- **Корректность реализации и тестирование (65%):**
    - корректная работа с `CUDA API` и `PyTorch C++ API`;
    - полнота тестового покрытия, включая граничные случаи;
    - соответствие результатов эталонной реализации.
- **Качество кода и архитектура (25%):**
    - чистота архитектуры;
    - единообразие стиля, качество форматирования и читаемость кода.
- **Документация и оформление (10%):**
    - полнота и структурированность отчёта;
    - ясность изложения;
    - оформление репозитория;
    - оформление отчёта (СТУ 7.5–07–2021).

Рекомендации по выполнению

- Реализуйте эффективный алгоритм параллельной свертки (tiling, shared memory), рассмотренный на лекции.
- Обратите внимание, что параметры `filter_size`, `stride`, `padding`, `dilation` передаются как шаблонные аргументы (compile-time constants), что позволяет компилятору развернуть циклы (loop unrolling).
- Форма выходного тензора $[N, C_{out}, H_{out}, W_{out}]$ вычисляется следующим образом:

$$H_{out} = \left\lfloor \frac{H_{in} + 2p - d\,(k-1) - 1}{s} + 1 \right\rfloor, \tag{3}$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2p - d\,(k-1) - 1}{s} + 1 \right\rfloor. \tag{4}$$

- Полезные ссылки:
    - Installing C++ Distributions of PyTorch;
    - Tensor Basics;
    - torch.nn.functional.conv2d.