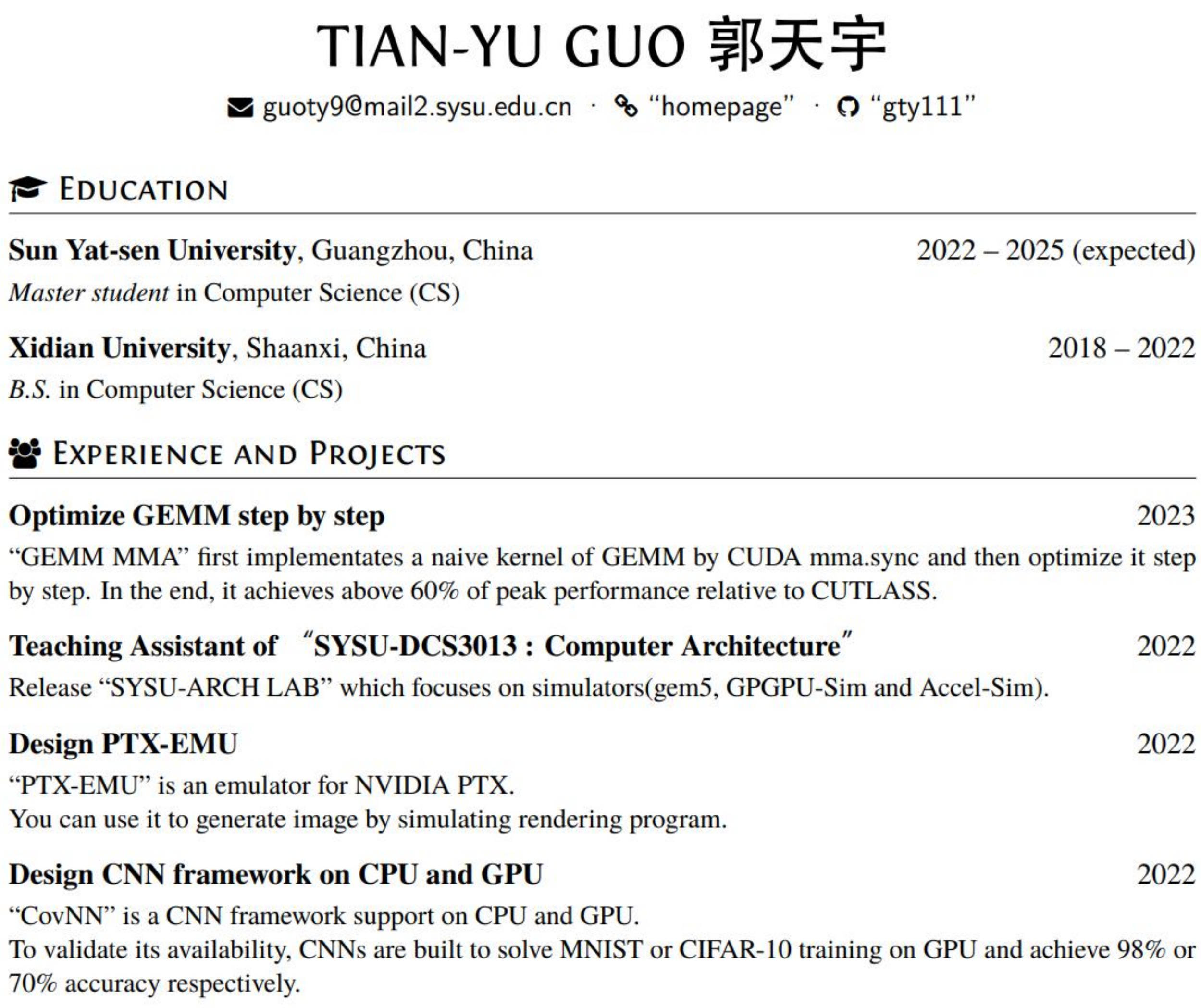
2023A3-算子实现和性能优化

队伍：arcsysu1

队长：郭天宇

自我介绍



TIAN-YU GUO 郭天宇

guoty9@mail2.sysu.edu.cn · "homepage" · "gty111"

EDUCATION

Sun Yat-sen University, Guangzhou, China 2022 – 2025 (expected)
Master student in Computer Science (CS)

Xidian University, Shaanxi, China 2018 – 2022
B.S. in Computer Science (CS)

EXPERIENCE AND PROJECTS

Optimize GEMM step by step 2023
"GEMM MMA" first implementates a naive kernel of GEMM by CUDA mma.sync and then optimize it step by step. In the end, it achieves above 60% of peak performance relative to CUTLASS.

Teaching Assistant of "SYSU-DCS3013 : Computer Architecture" 2022
Release "SYSU-ARCH LAB" which focuses on simulators(gem5, GPGPU-Sim and Accel-Sim).

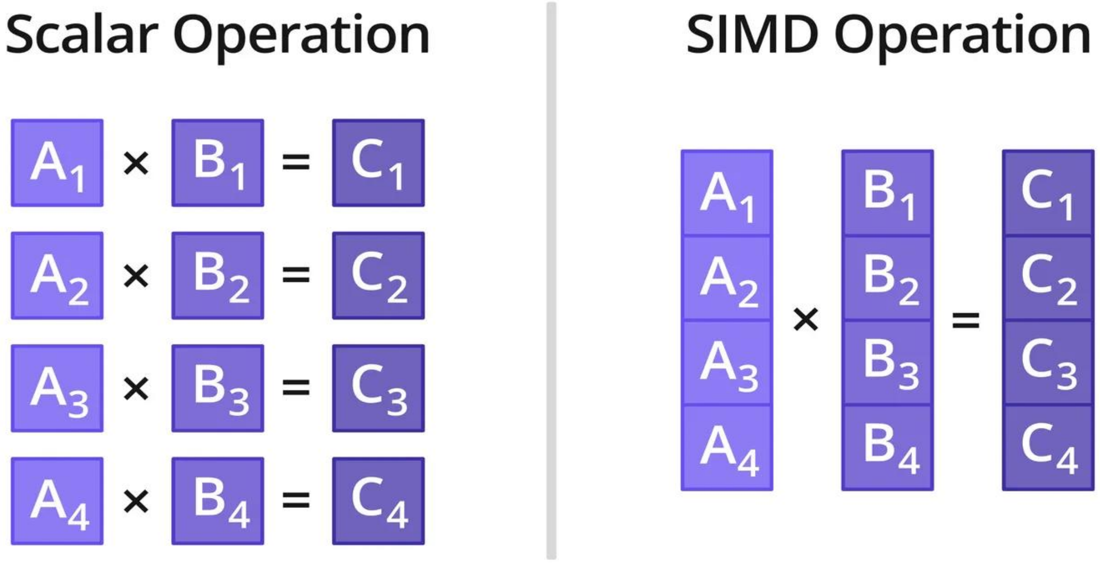
Design PTX-EMU 2022
"PTX-EMU" is an emulator for NVIDIA PTX.
You can use it to generate image by simulating rendering program.

Design CNN framework on CPU and GPU 2022
"CovNN" is a CNN framework support on CPU and GPU.
To validate its availability, CNNs are built to solve MNIST or CIFAR-10 training on GPU and achieve 98% or 70% accuracy respectively.

https://gty111.github.io

背景介绍

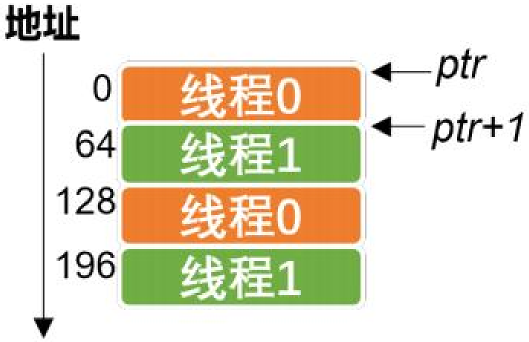
SIMD (Single Instruction, Multiple Data) 通常也被称为“单指令多数据”，是一种较为常见的并行计算技术。它能够同时对多个数据元素执行相同的操作，从而提高程序的执行效率。



SIMD向量处理器模型

GCU中的SIMD编程

- 数据类型v16f32，代表16个单精度浮点数
- 多线程SIMD访存，竞赛平台采用两线程硬件
- SIMD中的分支分歧与GCU的掩码操作
- SIMD内置函数接口

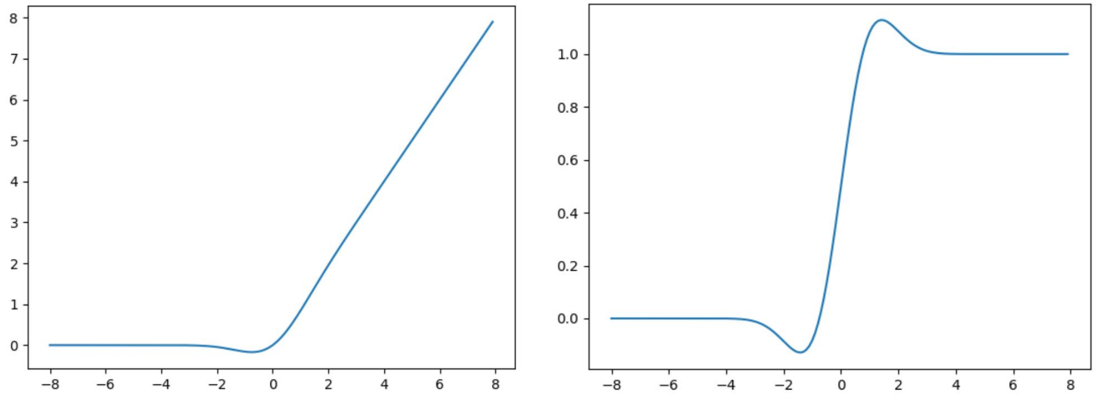


两线程交替处理数据

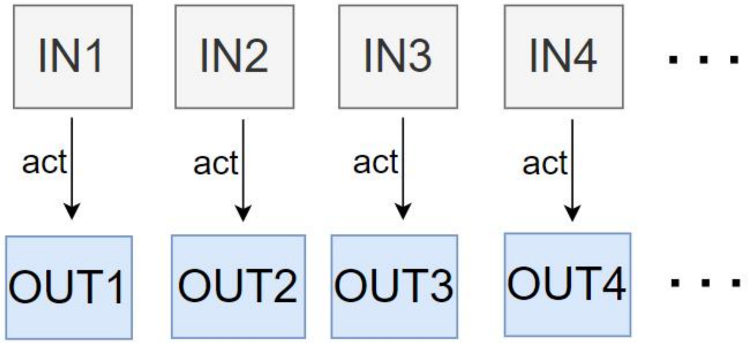
GELU激活函数(98. 41分)

GELU激活函数的近似表示：

$$\mathrm{GELU}(x) = 0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.044715x^3)))$$

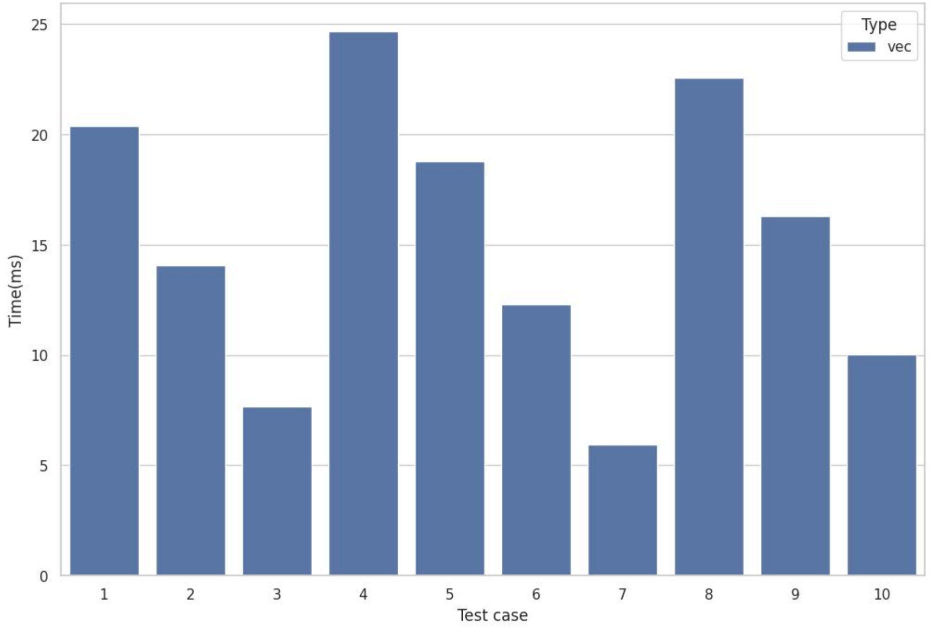


GELU激活函数及其导数



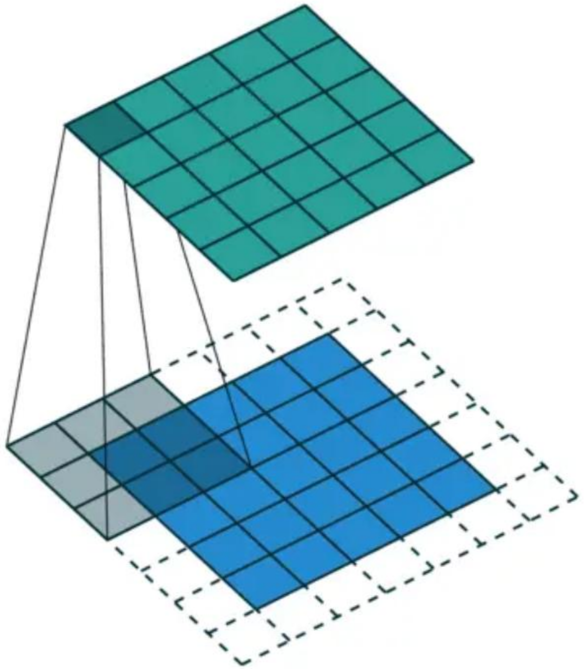
激活函数属于Elementwise操作

- 边界处理：不足以打包成向量化处理的数据可以退化为标量处理
- 数学库：使用提供好的数学库实现GELU激活函数(tanh和power)
- 常量向量的初始化：由于需要调用提供的接口实现SIMD操作，需要常量向量作为辅助计算



GELU赛题的用时

高斯模糊算子(78.56分)



通用卷积算法图示

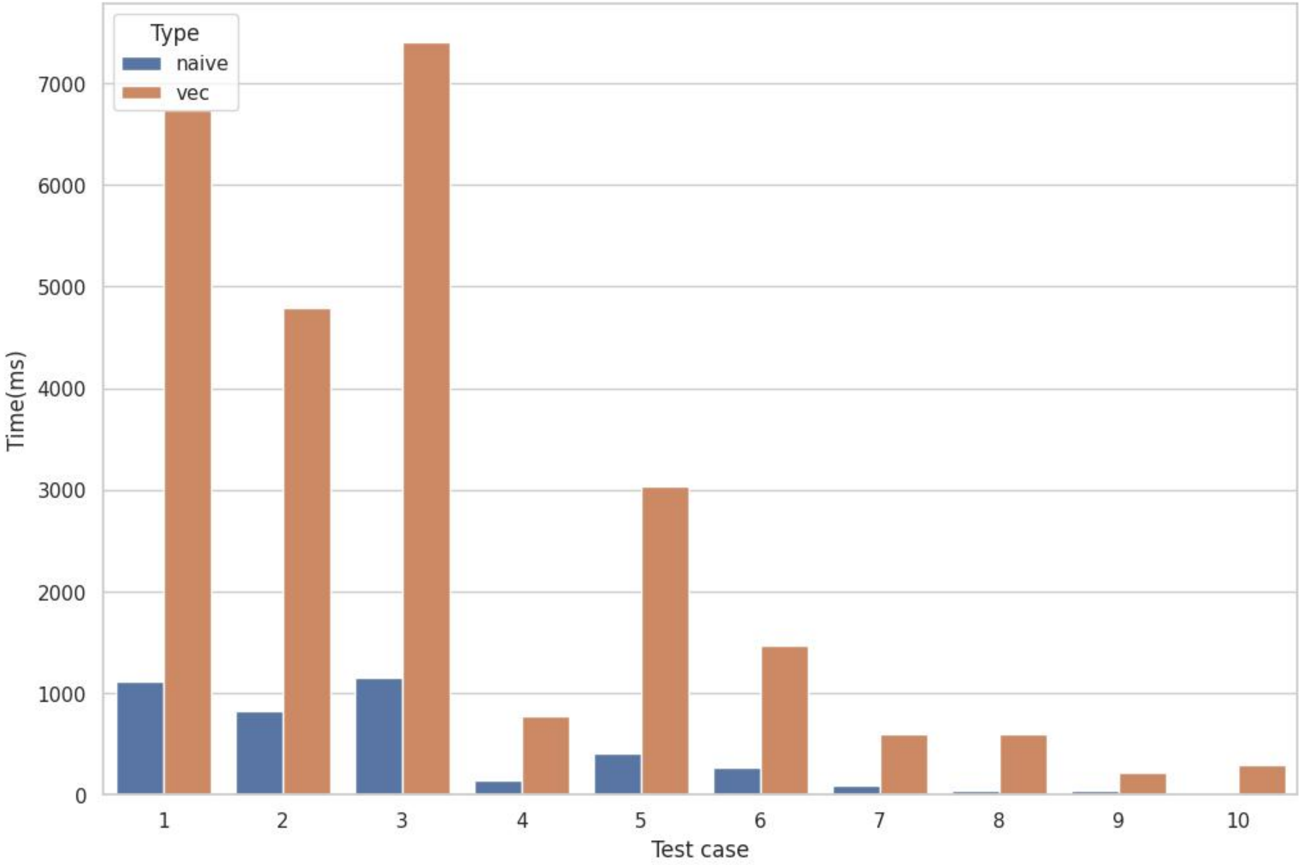
卷积操作的后端实现

(CUDNN、MKL等库)

- FFT
- img2col
- Winograd

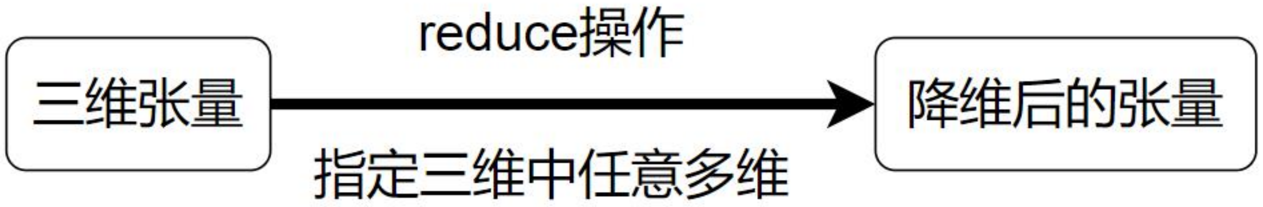
本次竞赛尝试的方法

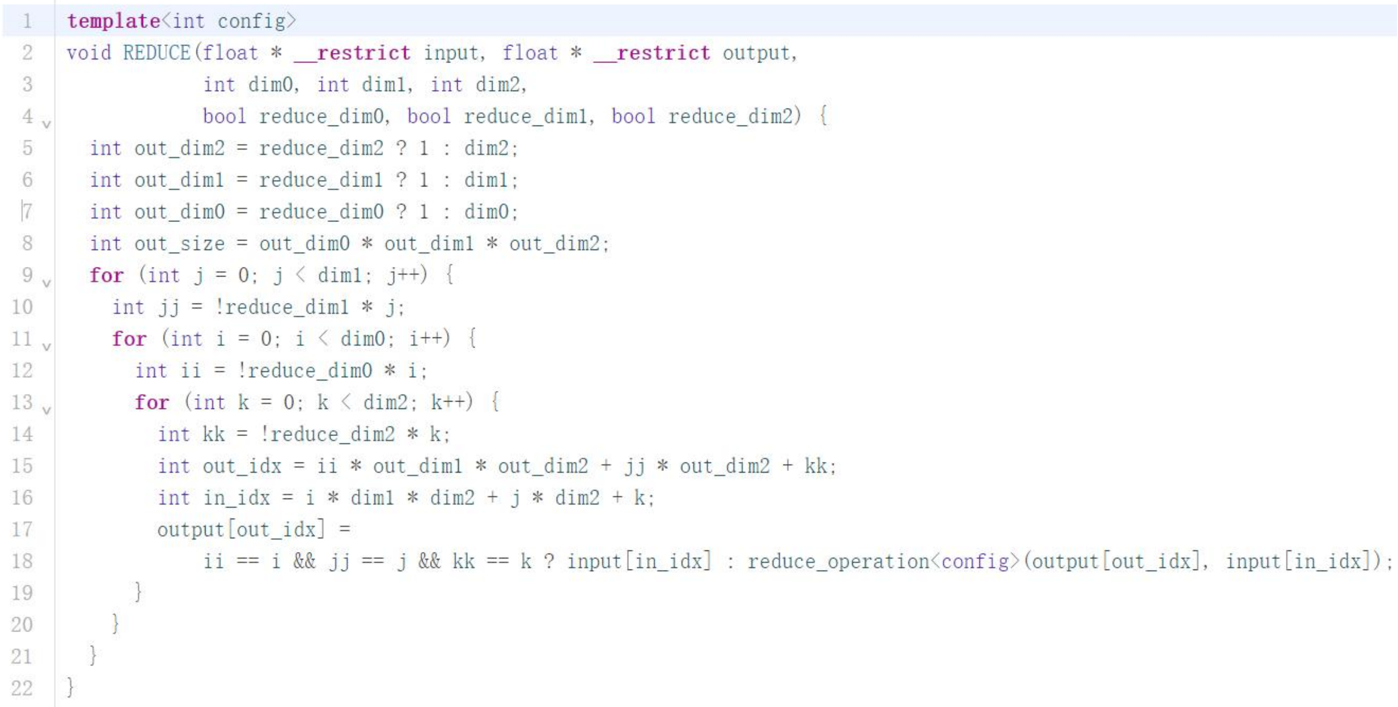
- 直接卷积计算
- 向量化



直接卷积(naive)和向量化(vec)方法所用时间对比

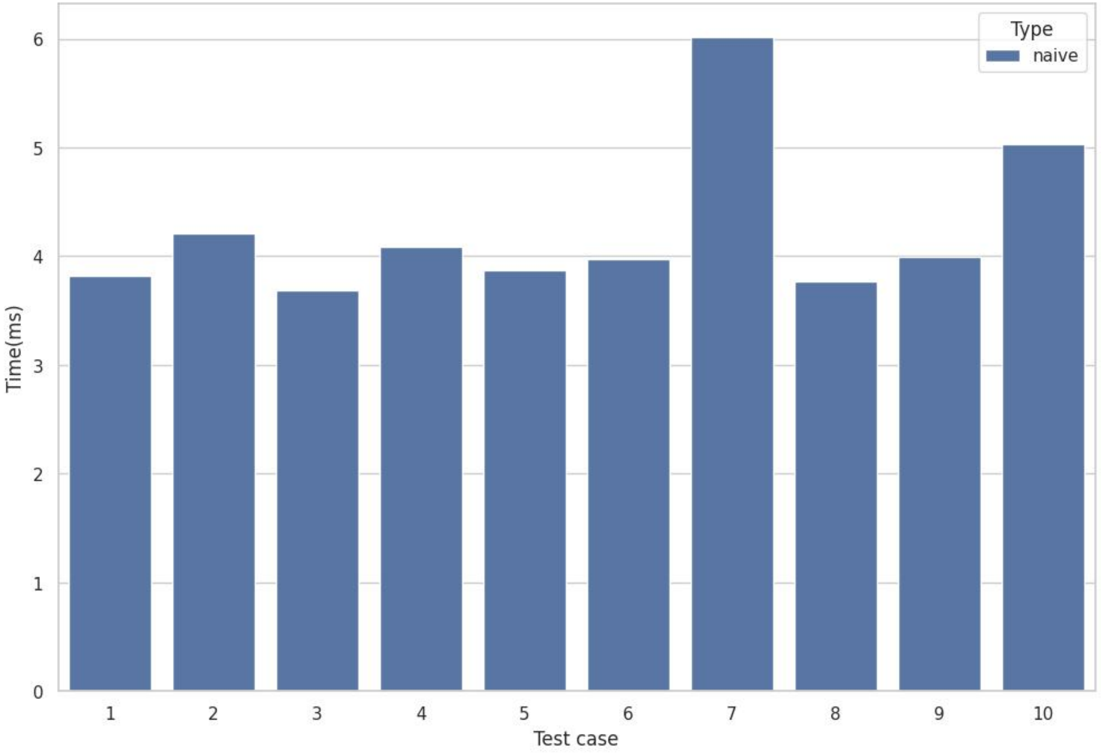
reduce规约算子(52.83分)





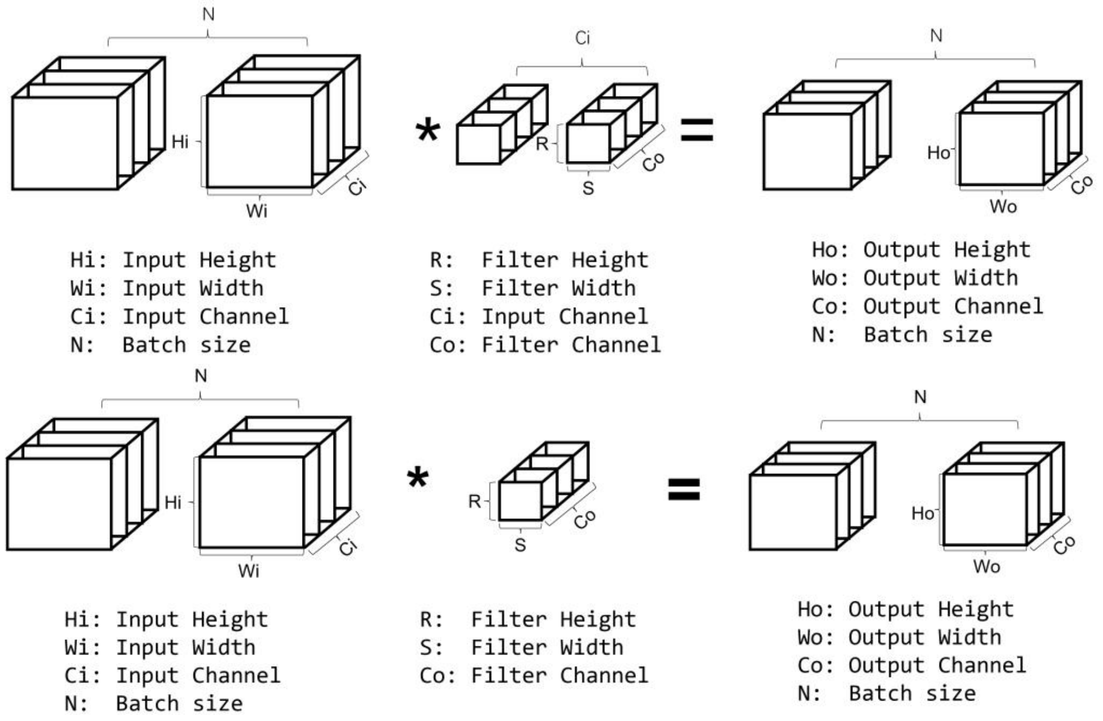
```
template<int config>
void REDUCE(float * __restrict input, float * __restrict output,
            int dim0, int dim1, int dim2,
            bool reduce_dim0, bool reduce_dim1, bool reduce_dim2) {
  int out_dim2 = reduce_dim2 ? 1 : dim2;
  int out_dim1 = reduce_dim1 ? 1 : dim1;
  int out_dim0 = reduce_dim0 ? 1 : dim0;
  int out_size = out_dim0 * out_dim1 * out_dim2;
  for (int j = 0; j < dim1; j++) {
    int jj = !reduce_dim1 * j;
    for (int i = 0; i < dim0; i++) {
      int ii = !reduce_dim0 * i;
      for (int k = 0; k < dim2; k++) {
        int kk = !reduce_dim2 * k;
        int out_idx = ii * out_dim1 * out_dim2 + jj * out_dim2 + kk;
        int in_idx = i * dim1 * dim2 + j * dim2 + k;
        output[out_idx] =
            ii == i && jj == j && kk == k ? input[in_idx] : reduce_operation<config>(output[out_idx], input[in_idx]);
      }
    }
  }
}
```

实现reduce操作的代码



reduce赛题的用时

Depthwise 2D卷积算子(44.04分)



Hi: Input Height
Wi: Input Width
Ci: Input Channel
N: Batch size

R: Filter Height
S: Filter Width
Ci: Input Channel
Co: Filter Channel

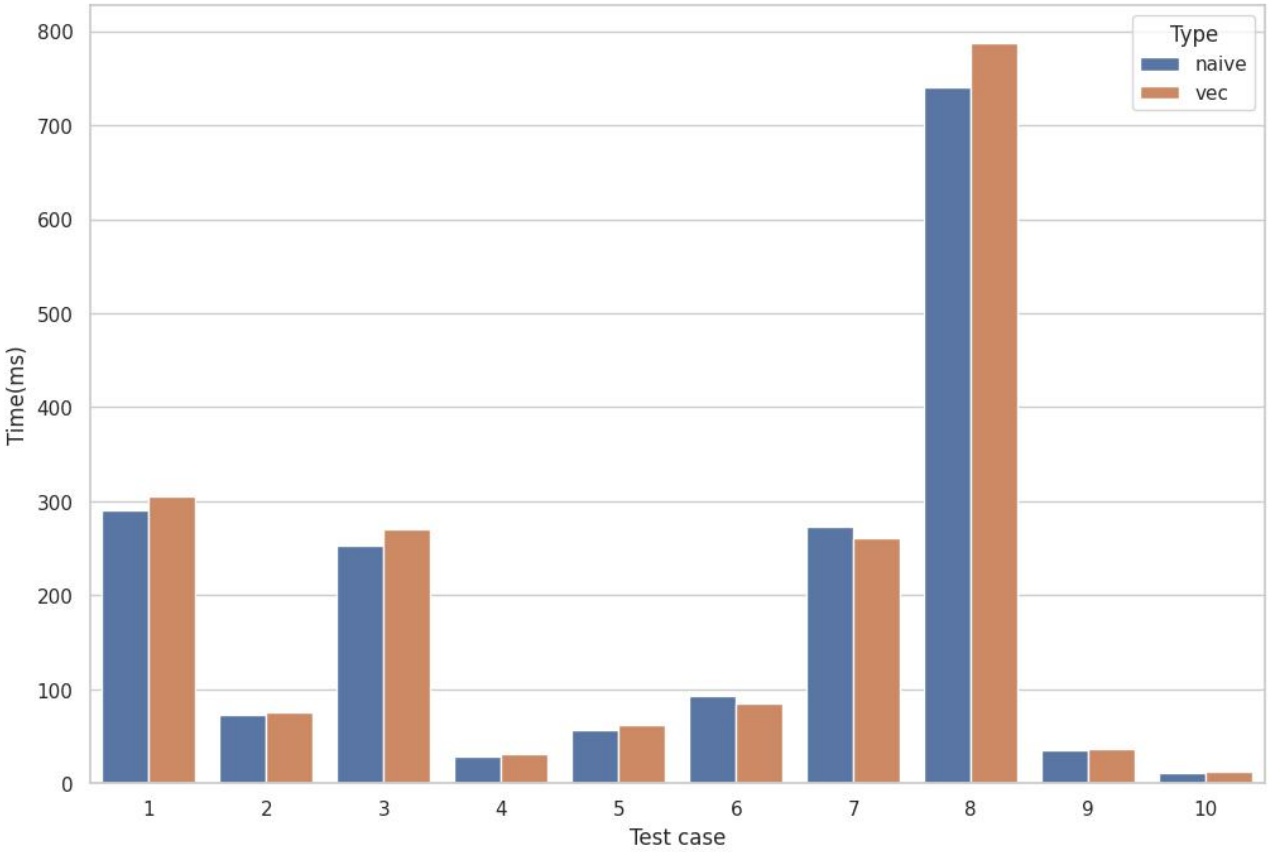
Ho: Output Height
Wo: Output Width
Co: Output Channel
N: Batch size

Hi: Input Height
Wi: Input Width
Ci: Input Channel
N: Batch size

R: Filter Height
S: Filter Width
Co: Filter Channel

Ho: Output Height
Wo: Output Width
Co: Output Channel
N: Batch size

普通卷积(上)和Depthwise卷积(下)的对比



直接卷积(naive)和向量化(vec)方法所用时间对比

本次竞赛尝试的方法：

- 直接卷积计算
- 向量化

THANKS & QA