# Tensor Convolution

Hao Deng

October 2023

## 1 Introduction

In this assignment, we want to investigate two implementations of Tensors, one dimensional (Tensor1D) implementation and four dimensional (Tensor4D) implementation.

The underlying storage of Tensor1D is an one dimensional vector, while that of Tensor4D is four nested vectors. We will implement these two classes and compare their performance in direct convolution under the following compilation optimization arguments:

1. -O0
2. -O2
3. -O3
4. -Ofast -march=native

For each compilation optimization option, we will plot the performance in terms of runtime and FLOPS (or GFLOPS).

## 2 Experiment

To compare the performance under different salutations, we define the following standard for comparison:

For direct convolution:

$$C = A * B$$

1. Fix B's dimension, with $batch = 1$, $channel = 3$, $height = 3$, and $width = 3$.
2. Fix A's batch and channel, with $batch = 1$ and $channel = 3$.
3. Increase A's height and width from 100 to 5000, with $step = 100$.
4. For each iteration, we record runtime and calculate GFLOPS.

The followings are the results under different compilation optimization options:

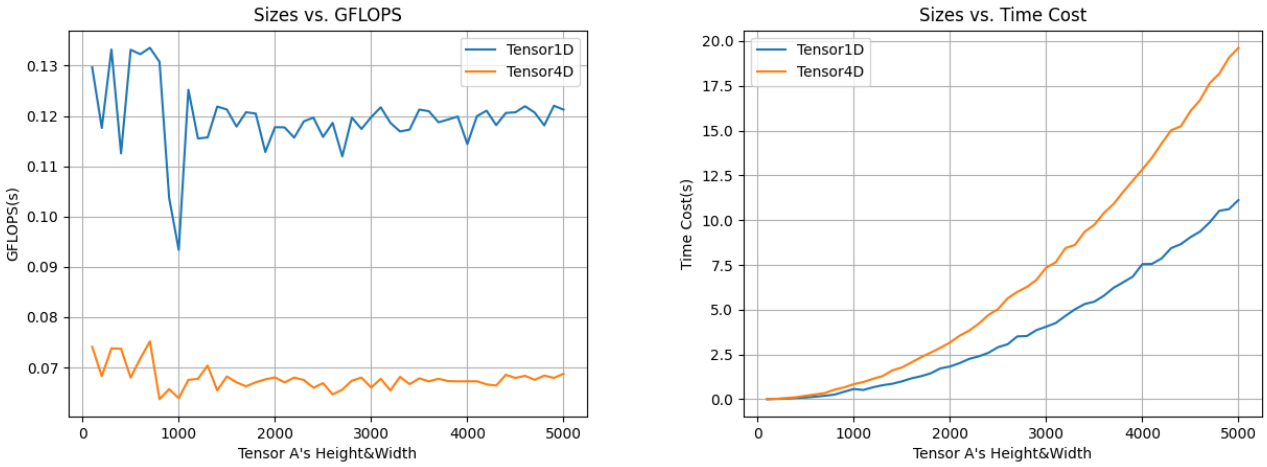


Figure 1: Compilation Optimization option: -O0

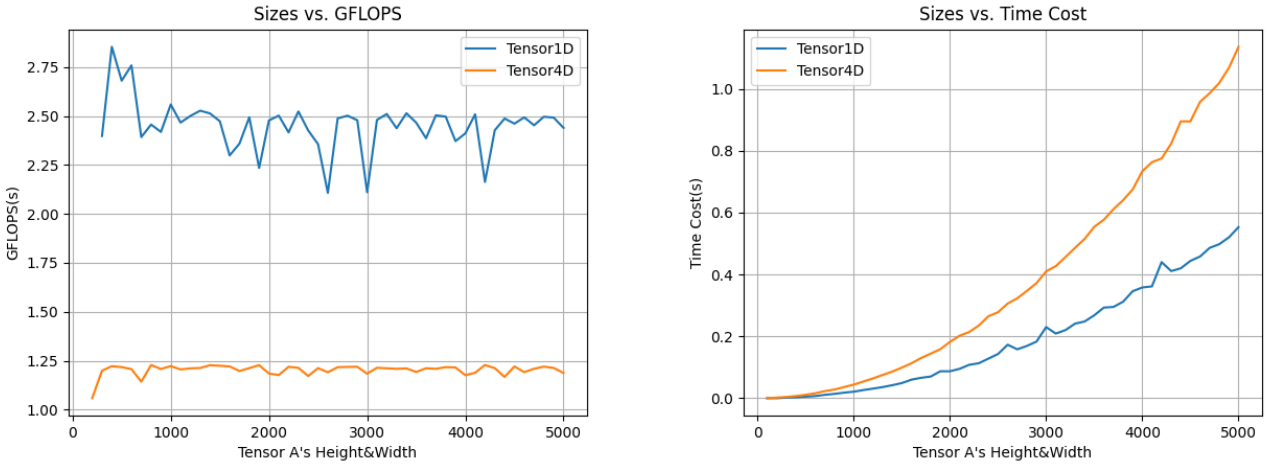


Figure 2: Compilation Optimization option: -O2

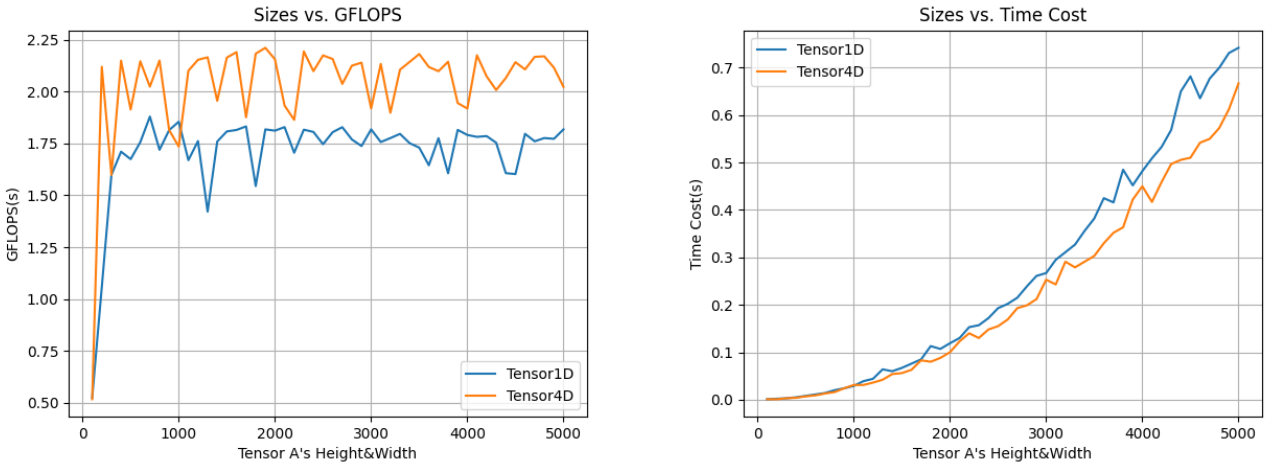


Figure 3: Compilation Optimization option: -O3

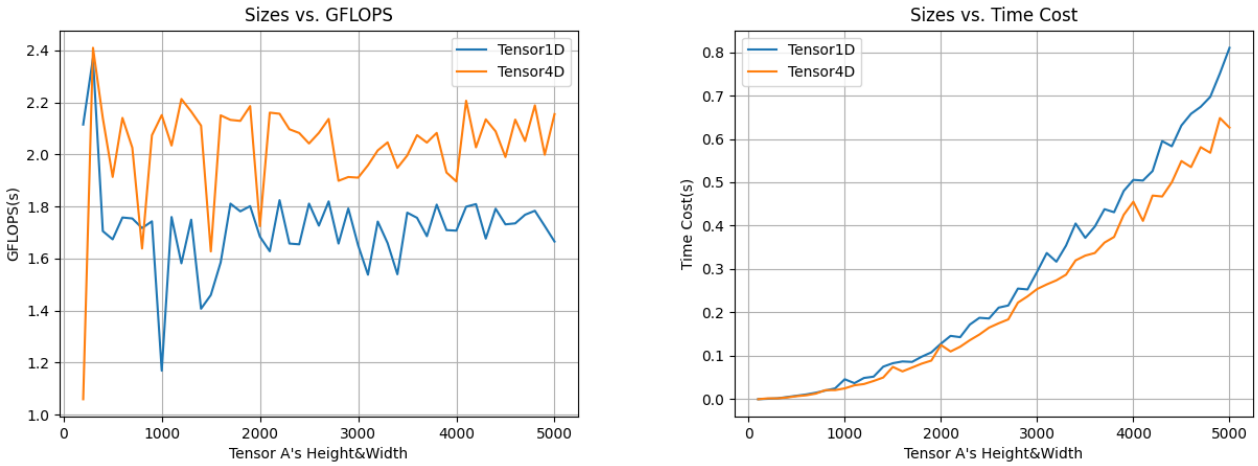


Figure 4: Compilation Optimization option: -Ofast -march=native

# 3 Analysis

Under -O0, we notice that Tensor1D has better performance over Tensor4D.

Let us print out the address for data in both Tensor1D and Tensor4D, both containing int with dimensions $batch = 1$, $channel = 3$, $height = 3$, and $width = 3$.

| Tensor1D | Tensor4D |
|---|---|
| Batch 0 | Batch 0 |
| Channel 0 | Channel 0 |
| 0xe54990 | 0xe64a60 |
| 0xe54994 | 0xe64a64 |
| 0xe54998 | 0xe64a68 |
| | |
| 0xe64aa0 | 0xe5499c |
| 0xe64aa4 | 0xe549a0 |
| 0xe64aa8 | 0xe549a4 |
| | |
| 0xe64ae0 | 0xe549a8 |
| 0xe64ae4 | 0xe549ac |
| 0xe64ae8 | 0xe549b0 |
| | |
| Channel 1 | Channel 1 |
| 0xe64ba0 | 0xe5499c |
| 0xe64ba4 | 0xe549a0 |
| 0xe64ba8 | 0xe549a4 |
| | |
| 0xe64be0 | 0xe549a8 |
| 0xe64be4 | 0xe549ac |
| 0xe64be8 | 0xe549b0 |
| | |
| 0xe64c20 | 0xe549b4 |
| 0xe64c24 | 0xe549b8 |
| 0xe64c28 | 0xe549bc |

| Channel 2 | Channel 2 |
|---|---|
| 0xe64ce0 | 0xe549a8 |
| 0xe64ce4 | 0xe549ac |
| 0xe64ce8 | 0xe549b0 |
| | |
| 0xe64d20 | 0xe549b4 |
| 0xe64d24 | 0xe549b8 |
| 0xe64d28 | 0xe549bc |
| | |
| 0xe64d60 | 0xe549c0 |
| 0xe64d64 | 0xe549c4 |
| 0xe64d68 | 0xe549c8 |

Table 1: Two Consecutive Channel Address

We notice that each address in Tensor1D is separated from each other by 4 bytes. An int contains 32 bits or 4 bytes. Therefore, We reckon the data stored in Tensor1D is consecutive. However, we can not observe the same pattern in Tensor4D. Hence, the data in Tensor4D is not consecutive. We believer this is the reason for better performance of Tensor1D under -O0.

Under -O2, Tensor1D still has better performance compared to Tensor4D. However, under -O3 and -Ofast -march=native, Tensor4D has better performance comparedto Tensor1D.

Notice that from -O2 to -O3 and -Ofast -march=native, the performance of Tensor1D deteriorates while that of Tensor4D gets improved.

# 4 Conclusion

For direct convolution, Tensor1D has better performance under -O0 and -O2; Tensor4D has better performance under -O3 and -Ofast -march=native.