A Comparison of Different Algorithms for Einsum

BACHELOR THESIS

to be Awarded the Academic Degree

of

Bachelor of Science (B.Sc.)

in Informatics

FRIEDRICH-SCHILLER-UNIVERSITY JENA

Faculty for Mathematics and Informatics

Submitted by Leon Manthey

born on 31.10.1999 in Berlin

Supervisor: Mark Blacher

Jena, 15.05.2024

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce eleifend orci et venenatis cursus. Nullam eget ornare lacus. Donec non dolor non tellus eleifend vehicula. Sed et lorem lectus. Vestibulum sagittis sed nisi ac interdum. Duis nec accumsan velit, hendrerit malesuada magna. Aliquam erat volutpat. Cras eu ante nec est malesuada volutpat. Proin quis posuere quam. Etiam aliquam eros quis dui sagittis, a fermentum lectus rutrum. Nunc tempor mauris vel tellus facilisis rhoncus. Aliquam ut leo eget metus volutpat vestibulum. Nam non consequat ante. In rutrum felis in enim fringilla lacinia. Phasellus ut imperdiet risus. Curabitur tincidunt libero sed urna dignissim, eget rutrum felis scelerisque. Nunc ut convallis neque, non tincidunt nulla. Curabitur quis condimentum leo. Phasellus laoreet ligula vel mi commodo, id accumsan diam tristique. Maecenas euismod lorem in tempor iaculis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Ut eget purus sem. Suspendisse venenatis aliquet dignissim. Integer turpis lorem, tempus non turpis et, gravida aliquet erat. Sed vel neque non ex ultrices vestibulum. Aliquam purus quam, rhoncus non ante at, convallis sagittis erat. Sed justo elit, vulputate vel accumsan non, porta eget turpis. Proin eget ultrices sem. Nunc eu velit.

Contents

1 Introduction

Einstein summation notation is a powerful and compact notation used in mathematics and physics to represent sums over tensor indices. It was introduced by Albert Einstein in the early 20th century as a means to simplify expressions in the theory of relativity. [2] The notation is both elegant and efficient, making it a valuable tool in various fields such as theoretical physics, computational mathematics, and data science.

The fundamental operation in Einstein summation notation, often referred to simply as "Einsum," is the Einstein summation. This operation allows for the concise expression of various tensor operations, including element-wise multiplication, dot products, outer products, and matrix multiplications. The computational efficiency and expressiveness of Einsum have led to its adoption in numerous applications, ranging from machine learning to scientific computing.

In many practical applications, especially in machine learning and scientific computing, the data involved is often sparse. Sparse matrices are matrices in which most of the elements are zero. Handling sparse matrices efficiently requires specialized algorithms and data structures to avoid unnecessary computations and to save memory. Traditional libraries like NumPy [3] and other major artificial intelligence frameworks [4, 5] typically support Einstein summation for dense matrices, but not for sparse matrices. The only known library that aims to support Einsum operations on sparse tensors is Sparse [6]. However, like NumPy, Sparse only allows for the use of capital and small letters of the Latin alphabet as indices, which limits the number of dimensions tensors can have. Furthermore, real Einstein summation problems often inculde expressions with hundreds or even thousands of higher order tensors. In order to express the aforementioned operations we require a large set of unique symbols. Thus, our approaches are capable of handling all symbols in the UTF-8 encoding.

This thesis explores the implementation and performance of Einstein summation across different computing paradigms, with a particular focus on sparse tensors. Specifically, it focuses on comparing two distinct implementations to multiple libraries:

- SQL-based Implementation: This implementation is based on the algorithm presented in "Efficient and Portable Einstein Summation in SQL" by Blacher et al [1]. It constructs SQL queries dynamically using Python. While SQL is traditionally used for database operations, this approach demonstrates the versatility of SQL in performing tensor operations.

- C++ Implementation: The second implementation is written in C++, with multiple versions ranging from naive to optimized approaches. The C++ implementations aim to explore the performance trade-offs between simplicity and optimization, offering insights into how different coding strategies affect computational efficiency.

By comparing these implementations, this thesis aims to provide a comprehensive analysis of the performance and scalability of sparse Einstein summation in various computing environments. The SQL-based implementation serves as a baseline, showcasing the potential of database query languages for tensor operations. Furthermore, the C++ implementations demonstrate the impact of low-level optimizations on computational performance. Comparing these against the sparse library Sparse and highly performance-tuned dense libraries like PyTorch provides insights into different use cases and helps identify the optimal tool for various tasks. The code for the tools can be found on Github https://github.com/Lethey2552/Sparse-Einsum.

Through this comparative study, we seek to identify the strengths and weaknesses of each approach, providing guidelines for selecting the appropriate method based on specific use cases and computational requirements. This work contributes to the broader understanding of tensor operations and their efficient implementation, offering practical insights for researchers and practitioners in fields that rely heavily on tensor computations.

2 Background

2.1 Tensors

Tensors are algebraic objects and a fundamental concept in mathematics, physics and computer science. They extend the idea of scalars, vectors and matrices to higher dimensions. In essence, a tensor is a multi-dimensional array with an abitrary number of dimensions.
Each dimension of a tensor is represented by an index with its own range. The number of indices is commonly referred to as the tensor's "rank" or "order." The size of a tensor is determined by the product of the maximum values of each index's range.
For example, consider a tensor T with indices i, j, k and corresponding ranges $i \in \{1,2\},\ j \in \{1,2,3,4,5,6\}$ and $k \in \{1,2,3,4\}$. The size of tensor T is calculated as follows: $2 \cdot 6 \cdot 4 = 48$. This means tensor T has a total of 48 elements.

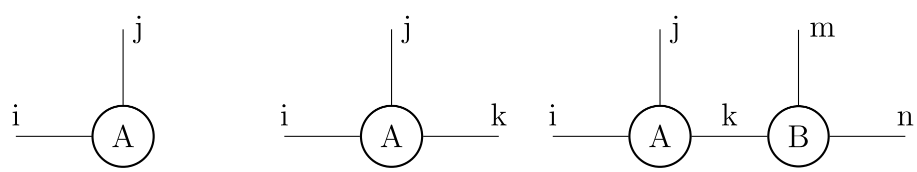

Figure 2.1: A matrix, a tensor and a tensor network visualized as a graph. Each index is represented by an edge, shared indices in a tensor network are edges between nodes.

2.2 Einstein Notation

In 1916, Albert Einstein introduced the so called Einstein notation, also known as Einstein summation convention or Einstein summation notation, for the sake of representing tensor expressions in a concise manner. As an example, the contraction of tensors $A \in \mathbb{R}^{I \times J \times K}$ and $B \in \mathbb{R}^{K \times M \times N}$ in figure 2.1,

$$C_{ijmn} = \sum_k A_{ijk} \cdot B_{kmn}$$

can be simplified by making the assumption that pairs of repeated indices in the expression are to be summed over.

Consequently, the contraction can be rewritten as:

$$C_{ijmn} = A_{ijk} \cdot B_{kmn}$$

To expant upon the expressive power of the original Einstein notation, modern Einstein notation was introduced. This notation is used by most linear algebra and machine learning libraries that provide Einstein summation notation. Modern Einstein notation explicitly states the indices for the output tensor, enabling further operations like transpositions and traces.
In modern Einstein notation, the expression from our previous example would be written as:

$$A_{ijk} B_{kmn} \to C_{ijmn}$$

When using common Einstein summation APIs, tensor operations are encoded by using the indices of the tensors in a format string and the data itself. The format string for the above operation would come down to:

$$ijk, kmn \to ijmn$$

In Modern Einstein notation, indices that are not mentioned in the output are to be summed over. For the sake of simplicity, we will from now on refer to Einstein summation as Einsum, and we will use the original, the modern notation or just the format string, depending on the context.

Bibliography

[1] Mark Blacher et al. "Efficient and Portable Einstein Summation in SQL". In: *Proc. ACM Manag. Data* 1.2 (2023).

[2] Albert Einstein. "Die Grundlage der allgemeinen Relativitätstheorie". In: *Annalen der Physik*. Vierte Folge Band 49 (1916). pp. 781-782, pp. 769–822.

[3] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[4] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: `https://www.tensorflow.org/`.

[5] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *CoRR* abs/1912.01703 (2019).

[6] Hameer Rocklin Matthew Abbasi. *Sparse 0.15.4*. `https://github.com/pydata/sparse`. 2024.