Proceedings of the 2023 International Conference on Mathematical Physics and Computational Simulation
DOI: 10.54254/2753-8818/11/20230394

Analysis of faster matrix multiplication

Jie Liu

University of California, Berkeley, Berkeley, CA, USA, 94720

jasperl@berkeley.edu

Abstract: Since the definition of matrices in 1855, matrix multiplication has played a crucial role in a wide range of fields. Over the years, numerous researchers have dedicated their efforts to improving the time complexity of this fundamental operation. This paper aims to delve into the historical development of matrix multiplication algorithms and methodologies employed to achieve these significant advancements in time complexity. By employing various approaches, researchers have been able to improve the time complexity of matrix multiplication, leading to a significant reduction from O (n^3) to O (n^{237188}). Across nearly two centuries, this progress is contributed by a lot of extraordinary scientists and researchers. This paper explores the practical implications of these improvements across various domains, such as computer science, physics, economics, and more. The development of more efficient matrix multiplication algorithms has enabled researchers and practitioners to tackle complex problems and explore new frontiers. In the future, with the rapid growth of machine learning techniques, matrix multiplication will continue to evolve and improve.

Keywords: matrix, multiplication, time complexity, strassen, machine learning.

1. Introduction

Matrix multiplication is an essential operation in various areas of science and engineering, from physics and statistics to computer graphics and machine learning. The concept of matrices and matrix multiplication was first introduced by Arthur Cayley in 1855, laying the foundation for subsequent research in this area. Over the years, researchers have developed increasingly efficient algorithms for matrix multiplication, reducing the time complexity from O (n^3) to the current best-known complexity of O (n^{237188}).

The development of matrix multiplication algorithms has a rich history, starting in the 1960s with the first fast matrix multiplication algorithms, for instance, Strassen's algorithm with a time complexity of O ($n^{2.81}$) or better. In the following decades, many researchers worked to improve the efficiency of matrix multiplication, leading to notable algorithms such as the Coppersmith-Winograd algorithm with a complexity of O (n^{2376}) and Storjohann's algorithm with a complexity of O ($n^{2.373}$). Throughout the 1990s and 2000s, researchers continued to explore different techniques, including hierarchical and recursive methods, as well as those that utilize multiple processors. More recently, François Le Gall discovered an algorithm with a complexity of O ($n^{2.372}$) in 2014 [1], which was the previous best-known complexity for matrix multiplication until Duan, Wu, and Zhou's recent algorithm with a complexity of O ($n^{2.37188}$) in 2022 [2].

© 2023 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2753-8818/11/20230394

This paper aims to review the historical development of matrix multiplication algorithms, discussing the methods used to achieve significant improvements in time complexity. We will explore the practical implications of these results for various fields and identify potential future directions for research. By understanding the evolution of matrix multiplication algorithms, we can better appreciate the impact that these algorithms have had on scientific and engineering applications, and how they continue to shape the future of these fields.

2. Matrix multiplication

To define a matrix, it could be described as a rectangular array of numbers arranged in rows and columns. The number of columns in the first matrix must equal the number of rows in the second matrix in order to multiply the two matrices. This is a fundamental operation in linear algebra. A new matrix is created by multiplying two other matrices. The final matrix shares exactly the same number of rows and columns with the first and second matrices. The elements of the two input matrices are the sum of the products of their respective components of the resultant matrix are calculated. For example, two matrices, A and B, with A having m rows and n columns and B having n rows and p columns, are present. To multiply A and B, we get a resulting matrix C. The sum of the products of the corresponding components of the i-th row of A and the j-th column of B, for each element in the resultant matrix C, where i represents the rows in matrix A from row 1 to row m and j represents the columns in matrix B from column 1 to column p. The resulting matrix C has n rows same as matrix A and p columns same as matrix B. For each element in the resulting matrix C, it can be computed as the sum of all products of corresponding row in matrix A and corresponding column in matrix B.

To compare the efficiency of different algorithms, the concept of time complexity is introduced. The time complexity of an algorithm is a measure of the amount of time it will take to run as a function. In particular, the notation O is a method to represent time complexity called Big-O-Notation. For example, O (n) is used to describe the time complexity with respect to the input size n. The notation O (n) means that the time complexity grows no faster than a constant times n, as n gets very large. Also, the growth rate of O (n) will be linear in the graph. If an algorithm has a time complexity of O (n^2), it means that the time it takes to run grows no faster than a constant times n squared, as n gets very large. The growth rate will be parabola in the graph.

Then, we could use this method to calculate the time complexity of matrix multiplication. The naive approach, the simplest algorithm, is simple nested loops. In other words, we loop m, n, and q to get their products and add them together. The time complexity is O (n^3) because the algorithm needs to compute each element of the resulting matrix by calculating the sum of n products, each of which requires n multiplications and n-1 additions. Consequently, for every component of the resultant matrix, there are n^2 multiplications and n^2- n additions. Since there are n^2 elements in the resulting matrix, the total number of operations required is n to the cube operations.

3. Strassen's algorithm and coppersmith-winograd algorithm

The naive algorithm for two n×n matrix multiplication requires n^3 scalar multiplications and $n^3 - n^2$ scalar additions. That's a lot of calculations, especially when the size of matrix n grows to a great number. In the 1960s, Volker Strassen published a paper about a clever way of two 2*2 matrices multiplication [3]. He divided a matrix into four submatrices of equal size, and recursively compute the products of these submatrices. Here is the idea of Strassen's algorithm. Let A and B be two matrices of size n x n, and C is the result of their products. Let A1, A2, A3, A4, B1, B2, B3, B4, and C1, C2, C3, C4 be their corresponding submatrices of size $\frac{n}{2} \times \frac{n}{2}$.

$$\begin{pmatrix} A1 & A2 \\ A3 & A4 \end{pmatrix}\begin{pmatrix} B1 & B2 \\ B3 & B4 \end{pmatrix} = \begin{pmatrix} C1 & C2 \\ C3 & C4 \end{pmatrix}$$

$$\begin{gathered} P1 = A1 \times (B2 - B4) \quad P2 = (A1 + A2) \times B4 \quad P3 = (A3 + A4) \times B1 \\ P4 = A4 \times (B3 - B1) \qquad P5 = (A1 + A4) \times (B1 + B4) \\ P6 = (A2 - A4) \times (B3 + B4) \qquad P7 = (A1 - A3) \times (B1 + B2) \\ C1 = P5 + P4 - P2 + P6 \qquad C2 = P1 + P2 \end{gathered} \tag{1}$$

DOI: 10.54254/2753-8818/11/20230394

$$C3 = P3 + P4 \qquad C4 = P5 + P1 - P3 - P7$$

Strassen's algorithm used only seven multiplications and 18 additions or subtractions to multiply two matrices.If Strassen's algorithm and the naïve algorithm are applied to two m*m matrices, the total operation count for the naïve algorithm is $m^3 + m^3 - m^2$. However, with Strassen's algorithm, the total operation count is

$$7 \times (2(\frac{m}{2})^3 - (\frac{m}{2})^2) + 18(\frac{m}{2})^2 = \frac{7}{4}m^3 + \frac{11}{4}m^2. \tag{2}$$

The ratio of operation counts for two different algorithms is

$$\frac{\frac{7}{4}m^3 + \frac{11}{4}m^2}{m^3 + m^3 - m^2} = \frac{7m^3 + 11m^2}{8m^3 - 4m^2}. \tag{3}$$

With the size of matrices m increasing, the ratio of the two algorithms increases more. In other words, Strassen's algorithm will become more efficient, with a nearly 12.5% improvement over the naïve algorithm [4]. Strassen's algorithm could be a great start to calculate the matrix multiplication, and he improved the time complexity to O ($n^{2.81}$). Then, in 1981, based on the method of Strassen, the Coppersmith-Winograd algorithm, developed by Coppersmith and Winograd, improved Strassen's algorithm to use 7 multiplications and only 15 adds or subtracts. He used another combinational structure of matrices A, B, and C. [4]

$$\begin{gathered} P1 = A1 \times B1 \qquad P2 = A2 \times B3 \\ P3 = A4 \times (B1 - B2 - B3 + B4) \qquad P4 = (A1 - A3) \times (-B2 + B4) \\ P5 = (A3 + A4) \times (-B1 + B2) \qquad P6 = (A1 + A2 - A3 - A4) \times B4 \\ P7 = (A1 - A3 - A4) \times (B1 - B2 + B4) \\ C1 = P1 + P2 \qquad C2 = P1 + P5 + P6 - P7 \\ C3 = P1 - P3 + P4 - P7 \qquad C4 = P1 + P4 + P5 - P7 \end{gathered} \tag{4}$$

Coppersmith-Winograd's algorithm improves on Strassen's algorithm by reducing the number of required recursive steps, which reduces the number of arithmetic operations needed. The Fast Fourier Transform (FFT) makes use of the fact that matrix multiplication can be represented as a convolutional operation for the purposes of this algorithm. Winograd's algorithm splits the matrices into smaller submatrices, computes their convolutions using the FFT, and then combines these convolutions to obtain the product of the original matrices. This algorithm reduces the number of multiplications needed further from Strassen's algorithm and improves the time complexity to O ($n^{2.376}$).

4. William's algorithm

In 2012, Virginia Vassilevska Williams introduced a paper "Multiplying Matrices Faster than Coppersmith-Winograd" introducing her algorithm named Williams' algorithm [5]. She used a technique named randomization, which is always used for solving computational problems. William's algorithm samples a small number of entries from the matrices and uses these samples to estimate the result of the full matrix multiplication. If William's algorithm could choose the sampling distribution and the number of samples carefully, it is possible to achieve high accuracy with a smaller number of operations than the Coppersmith-Winograd algorithm [6]. Also, William's approach involved two main theorems. The first theorem entails choosing a specific group partitioning for any tensor power (A^n) of a fundamental algorithm A and devising a way to generate formulas for lower bounds on the values of these groups. The second theorem is that they assume they know the values for A^n and could use an efficient method to output a nonlinear constraint program with O (n^2) variables. The solution to this program provides a bound on ω. This approach allows for the analysis of any algorithm, including the algorithm with higher tensor powers through the use of a computer. The tensor of Matrix Multiplication corresponding to the multiplication of an m*n matrix by an n*p matrix is

$$< m, n, p > = \sum_{i=1}^{m} \sum_{j=1}^{p} \sum_{k=1}^{n} a_{ik} \otimes b_{kj} \otimes c_{ij}. \tag{5}$$

William uses these methods for the 2nd, 4th, and 8th tensor powers of the Coppersmith-Winograd algorithm, and with each new tensor power, she gets better bounds [7, 8]. Then she could calculate the matrix multiplication in O ($n^{2.3729}$), which is faster than the Coppersmith-Winograd algorithm. The most recent found algorithm for matrix multiplication will be the Duan-Wu-Zhou algorithm in 2022.

DOI: 10.54254/2753-8818/11/20230394

They used the similar method block structure to maximize the reuse of the intermediate results and minimize the number of multiplications required to compute the final output. They improved the time complexity of this problem to a new level, O ($n^{2.376}$) [8].

Last but not least, to consider each algorithm, a very important point could not be ignored is the memory usage. Various algorithms use different number of operations to process matrices, which lead to different memory usages, such as Strassen's algorithm and Coppersmith-Winograd's algorithm. Strassen's algorithm needs to allocate memory to their submatrices. It requires the allocation of 7 temporary matrices, each with size of $\frac{n}{2} \times \frac{n}{2}$. Coppersmith-Winograd's algorithm could reduce the number of multiplications required, but it requires even more memory than Strassen's algorithm. Since Coppersmith-Winograd's algorithm recursively divides the matrices into 15 submatrices and requires the memory to store them. Specifically, Coppersmith-Winograd's algorithm requires the allocation of 122 temporary matrices, each with the size of $\frac{n}{3} \times \frac{n}{3}$. Sometimes, the better time complexity of an algorithm needs the sacrifice of more memory usage.

5. Matrix multiplication application and future development

Matrix multiplication is a versatile technique can be applied across various fields including physics, engineering, computer science, and economics. Within the realm of computer science, matrices are commonly used, and matrix multiplication enables computers to perform significant precomputed computational tasks. Although creating a matrix that produces valuable computational outcomes can be challenging, the process of performing matrix multiplication itself is straightforward. One specific area where matrix multiplication is applied generally is the graphics. In this context, digital images can be represented as matrices, where the numerical values correspond to the colors of pixels. For example, matrix multiplication is employed in decoding digital videos, and researchers at MIT have successfully developed a chip that implements an advanced video-coding standard for ultra-high-definition televisions. They achieved this by identifying patterns in the matrices utilized. [9] Similarly, matrix multiplication plays an essential role in processing digital sound. By representing a digital audio signal as a sequence of numbers that capture the variations in air pressure over time, matrix multiplication facilitates techniques like filtering and compressing digital audio signals. The Fourier transform, used in such processes, also relies on matrix multiplication.

As we can see, matrix multiplication is very important, not only in computer science, so improving the time complexity of matrix multiplication becomes an essential task. The development of matrix multiplication algorithms, including Strassen's, Coppersmith-Winograd's, and Williams' algorithms mentioned in this paper, has greatly contributed to the matrix multiplication problem. These researchers have dedicated their efforts to improving the time complexity of these problems that led to significant advancements in the computer science field.

In addition to the advancements in matrix multiplication algorithms, the future of this field looks promising with the development of artificial intelligence. With neural networks and artificial intelligence developing so quickly, there will be an increased need for efficient matrix multiplication algorithms. Scientists have previously utilized profound support learning (DRL) to find provably right and effective framework augmentation calculations [10]. By formalizing a rich space of framework increase calculations as low-rank disintegrations of a 3D tensor, the hunt interaction becomes manageable to mechanization. By enabling machines to recognize and generalize patterns in tensors, DRL provides a novel approach, allowing for the prediction of efficient decompositions. Also, an artificial intelligence company called DeepMind has already broken the 50-year mathematic record [11]. Thus, the future of matrix multiplication looks promising with the development of artificial intelligence and the potential for new and more efficient algorithms.

6. Conclusion

Throughout this paper, we have examined the progress of time complexity in matrix multiplication algorithms. Starting with the naive approach, this paper explored significant milestones and breakthroughs that

DOI: 10.54254/2753-8818/11/20230394

have led to improved efficiency and reduced computational costs. From naïve approach with time complexity O (n^3) to the most recent advancements O (n^{237188}), researchers have continuously pushed the boundaries of efficiency in matrix multiplication. The development of matrix multiplication algorithms in history serves as a testament to the power of human innovation and the pursuit of knowledge. The advancements made in this field have led to improvements in various fields and applications, and we can only imagine what possibilities await us in the future. Once again, we express our gratitude to the researchers who have dedicated their time and effort to this field, and we look forward to the future advancements that will undoubtedly come.

References

[1] Le Gall, François (2014), "Algebraic complexity theory and matrix multiplication", in Ka-tsusuke Nabeshima (ed.), Proceedings of the 39th International Symposium on Symbo-lic and Algebraic Computation - ISSAC '14, pp. 296–303, arXiv:1401.7714, Bibcode:2-014arXiv1401.7714L, doi:10.1145/2608628.2627493, ISBN 978-1-4503-2501-1, S2CID 25-97483

[2] Zhang, Y., & Yu, H. (2022). Faster Matrix Multiplication via Partitioned Computing. arXiv preprint arXiv:2210.10173. Retrieved from https://arxiv.org/pdf/2210.10173.pdf

[3] V. Strassen. Gaussian elimination is not optimal. Numer. Math., 13:354-356, 1969.

[4] Steven Huss-Lederman, Elaine M. Jacobson, Anna Tsao, Thomas Turnbull, and Jeremy R. Johnson. 1996. Implementation of Strassen's algorithm for matrix multiplication. In Proceedings of the 1996 ACM/IEEE conference on Supercomputing (Supercomputing '96). IEEE Computer Society, USA, 32–es. https://doi.org/10.1145/369028.369096

[5] Virginia Vassilevska Williams (2012). "Multiplying Matrices Faster than Coppersmith-Winograd". In Howard J. Karloff; Toniann Pitassi (eds.). Proc. 44th Symposium on Theory of Computing (STOC). ACM. pp. 887–898. doi:10.1145/2213977.2214056. S2CID 14350287.

[6] Williams, Virginia Vassilevska. Multiplying matrices in O($n^{2.373}$)time (PDF) (Technical Report). Stanford University.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. J. Symbolic Computation, 9(3):251–280, 1990.

[8] D. Coppersmith and S. Winograd. On the asymptotic complexity of matrix multiplication. In Proc. SFCS, pages 82–90, 1981.

[9] Hardesty, L. (n.d.). Explained: Matrices. MIT News | Massachusetts Institute of Technology. Retrieved April 25, 2023, from https://news.mit.edu/2013/explained-matrices-1206

[10] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. Nature, 610(7930):47–53, 2022. doi:10.1038/s41586-022-05172-4

[11] Benj Edwards - Oct 13, 2022 8:46 pm U. T. C., & Scintillant Seniorius Lurkius et Sub-scriptor jump to post. (2022, October 13). Deepmind breaks 50-year math record usi-ng AI; New record falls a week later. Ars Technica. Retrieved April 20, 2023, fromhttps://arstechnica.com/information-technology/2022/10/deepmind-breaks-50-year-math-record-using-ai-new-record-falls-a-week-later/