

# Discovering faster matrix multiplication algorithms with reinforcement learning (AlphaTensor)

---

Presenter: Ye Yuan

2022.12.06

# Catalog

- Paper background
- Motivation, Problem, and Contribution
- **Mathematics Behind Matrix Multiplication**
- **Tensor Game**
- Conclusion and Thoughts



# Paper Background

---

- Received: 2 October 2021
- Accepted: 2 August 2022
- Published online: 5 October 2022
- Featured front cover of Nature on Volume 610 Issue 7930, 6 October 2022
- This article is openly accessible; you don't have to subscribe to Nature journal.
- [Link](#)

# Authors of Paper

---

- Alhussein Fawzi<sup>1,2</sup> ✉, Matej Balog<sup>1,2</sup>, Aja Huang<sup>1,2</sup>, Thomas Hubert<sup>1,2</sup>, Bernardino Romera-Paredes<sup>1,2</sup>, Mohammadamin Barekatain<sup>1</sup>, Alexander Novikov<sup>1</sup>, Francisco J. R. Ruiz<sup>1</sup>, Julian Schrittwieser<sup>1</sup>, Grzegorz Swirszcz<sup>1</sup>, David Silver<sup>1</sup>, Demis Hassabis<sup>1</sup> & Pushmeet Kohli<sup>1</sup>



Alhussein Fawzi



Matej Balog



Aja Huang



Thomas Hubert



Bernardino Romera-Paredes

- <sup>1</sup>DeepMind, <sup>2</sup>These authors contributed equally

# Motivation

---

- Matrix multiplication is probably the most important matrix operation. It is used widely in network theory, the solution of the system of linear equations, the transformation of coordinate systems, and population modelling, to name a few. [1]
- For example:
  - When using a machine learning model for prediction, faster matrix multiplication results in faster prediction.
  - When using a machine learning model for autopilot, faster matrix multiplication can enhance the response time.

# Matrix Multiplication

---

- Take two  $2 \times 2$  square matrices as an example.

- $$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} a_1 \cdot b_1 + a_2 \cdot b_3 & a_1 \cdot b_2 + a_2 \cdot b_4 \\ a_3 \cdot b_1 + a_4 \cdot b_3 & a_3 \cdot b_2 + a_4 \cdot b_4 \end{pmatrix}$$

- This is the standard process to multiply these two matrices, which takes 8 multiplications and 4 additions. In fact, we do not care about the number of additions.
- In the general case, when multiplying two matrices with dimensions  $Q \times R$  and  $R \times P$ , we need  $Q \times R \times P$  multiplications since the result matrix has the dimension of  $Q \times P$ , and each entry need  $R$  multiplications (take the inner product).
- It seems like we can do nothing to accelerate this calculation.
- From the perspective of mathematics, we may not have a method to accelerate. However, we can accelerate this in computers.

# Addition and Multiplication

---

- These statistics are from the note of Prof Srinivas Devadas from MIT. [2]
- A “cycle” is technically a pulse synchronized by an internal oscillator, but for our purposes, they're a basic unit that helps understand a CPU's speed.

---

Name of Operation	Latency for CPU
-------------------	-----------------

64 Bits Integer Addition	1 cycle
--------------------------	---------

64 Bits Integer Multiplication	20 cycles
--------------------------------	-----------

# Accelerate Calculation

---

- As shown in the previous slides, the calculation process can be accelerated if we can use additions to substitute the multiplications.
- For example:
  - $a^2 - b^2 = a \times a - b \times b$  (2 multiplications and 1 addition, i.e., 41 cycles)
  - $a^2 - b^2 = (a + b) \times (a - b)$  (1 multiplication and 2 addition, i.e., 22 cycles)
- Actually, the two methods above are mathematically equivalent. However, as computer scientists, we prefer the second one since it use less multiplications.



# Bring This Idea to Matrix Multiplication

---

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$$

Standard Algorithm:

- $m_1 = a_1 \cdot b_1$
- $m_2 = a_1 \cdot b_2$
- $m_3 = a_2 \cdot b_3$
- $m_4 = a_2 \cdot b_4$
- $m_5 = a_3 \cdot b_1$
- $m_6 = a_3 \cdot b_2$
- $m_7 = a_4 \cdot b_3$
- $m_8 = a_4 \cdot b_4$
- $c_1 = m_1 + m_3$
- $c_2 = m_2 + m_4$
- $c_3 = m_5 + m_7$
- $c_4 = m_6 + m_8$

Strassen's Algorithm:

- $m_1 = (a_1 + a_4) \cdot (b_1 + b_4)$
- $m_2 = (a_3 + a_4) \cdot b_1$
- $m_3 = a_1 \cdot (b_2 - b_4)$
- $m_4 = a_4 \cdot (-b_1 + b_3)$
- $m_5 = (a_1 + a_2) \cdot b_4$
- $m_6 = (-a_1 + a_3) \cdot (b_1 + b_2)$
- $m_7 = (a_2 - a_4) \cdot (b_3 + b_4)$
- $c_1 = m_1 + m_4 - m_5 + m_7$
- $c_2 = m_3 + m_5$
- $c_3 = m_2 + m_4$
- $c_4 = m_1 - m_2 + m_3 + m_6$


- Each m include 1 multiplication
- Standard Algorithm includes 8 multiplications and 4 additions. That is 164 cycles.
- Strassen's Algorithm includes 7 multiplications and 18 additions. That is 158 cycles.
- We can save 6 cycles.

# Problem

---

- Even though the trick that uses additions instead of multiplications is a good way to accelerate calculation, it seems hard for us to discover a new one.
- Despite decades of research following Strassen's breakthrough, larger versions of this problem have remained unsolved – to the extent that it's not known how efficiently it's possible to multiply two matrices that are as small as  $3 \times 3$ . [3]
- **Is there a way to reformulate the problem of matrices multiplication algorithm discovery such that existing tools are helpful?**
- As we mentioned, in the general case, when multiplying two matrices with dimensions  $Q \times R$  and  $R \times P$ , we need  $Q \times R \times P$  multiplications.  $Q \times P$  is the dimension of the result matrix. We can do nothing about it.
- Therefore, if the number of multiplications used to calculate the inner product can be minimized (it might be smaller than the value of  $R$ ), we can achieve our goal!

# Contribution of This Work



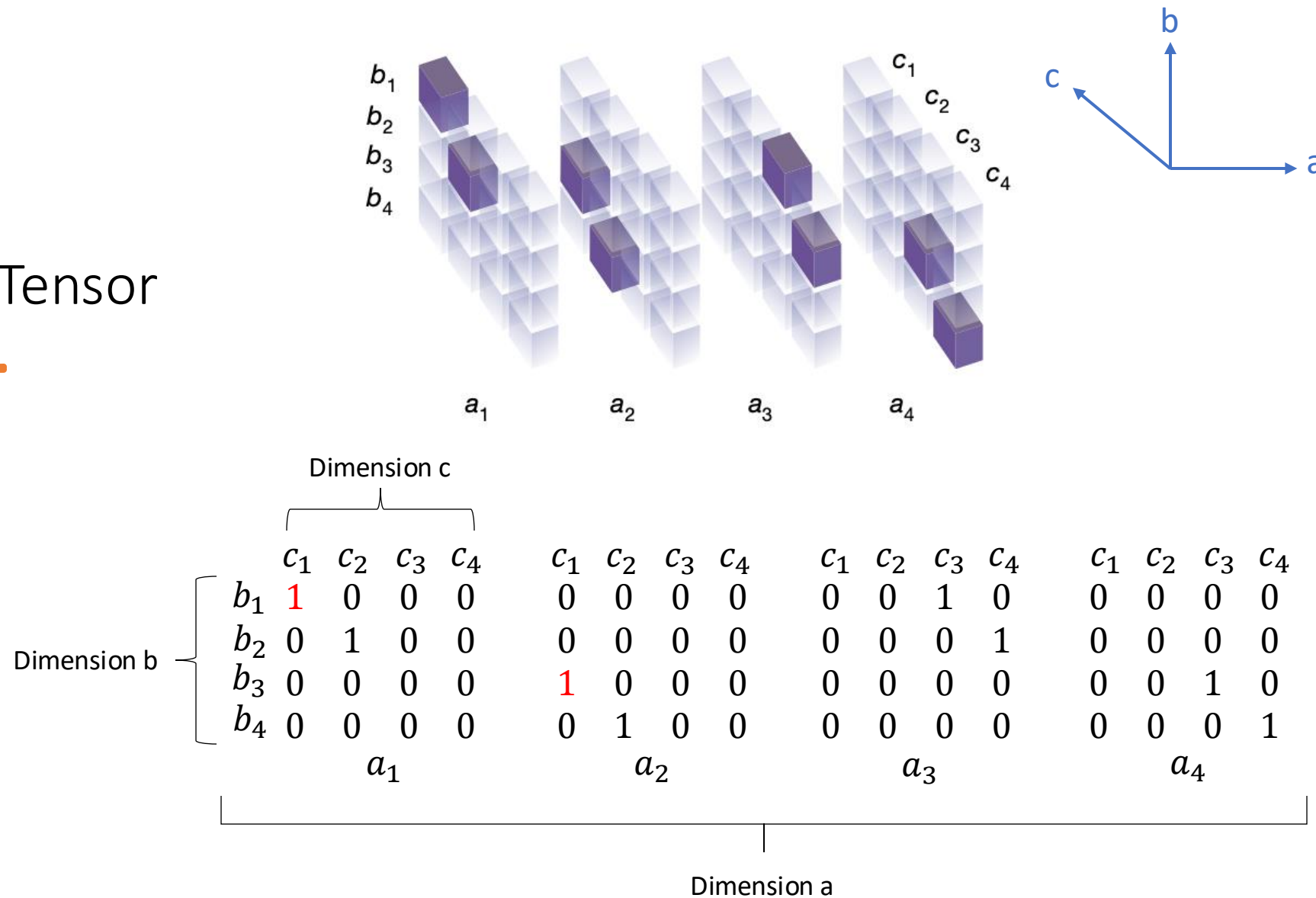
- In all cases, AlphaTensor discovers algorithms that match or improve over the known state-of-the-art algorithm.

# Mathematics Behind Matrix Multiplication

---

# Transform Matrices Multiplication to 3D Tensor

- $T_2 \doteq \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$
- Tensor  $T_2$  representing the multiplication of two  $2 \times 2$  matrices. Tensor entries equal to 1 are depicted in purple, and 0 entries are semi-transparent.
- The tensor specifies which entries from the input matrices to read, and where to write the result. For example, as  $c_1 = a_1b_1 + a_2b_3$ , tensor entries located at  $(a_1, b_1, c_1)$  and  $(a_2, b_3, c_1)$  are set to 1.



# Transform Matrices Multiplication to 3D Tensor

---

- As we see in the previous slide, the multiplication of two  $2 \times 2$  matrices can be rewritten to a 3D tensor with  $4 \times 4 \times 4$ .
- In general, for multiplication of two  $n \times n$  matrices, it can be rewritten to a 3D tensor with  $n^2 \times n^2 \times n^2$ .
- More generally, for the multiplication of two matrices with the dimension of  $n \times m$  and  $m \times p$ , it can be rewritten to a 3D tensor with  $(n \times m) \times (m \times p) \times (n \times p)$ .
- For simplicity, we would use the case of multiplication of two  $n \times n$  matrices. The corresponding 3D tensor with  $n^2 \times n^2 \times n^2$  is denoted as  $T_n$ .

# Recap Knowledge from Linear Algebra Course

- In linear algebra, the rank of a matrix  $A$  is the dimension of the vector space generated (or spanned) by its columns. This corresponds to the maximal number of linearly independent columns of  $A$ . [4]
- Similarly, the rank of a 3D tensor is the dimension of the matrix space generated (or spanned) by its "layers".

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

Rank 1

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Rank 1

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Rank 2

- The outer product  $u \otimes v$  is equivalent to a matrix multiplication  $u \cdot v^T$ , provided that  $u$  is a  $m \times 1$  and  $v$  is a  $n \times 1$  vector. [4]

$$\bullet \text{ If } u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \text{ and } v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}, \text{ we have } u \otimes v = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 \end{bmatrix}$$

# Decomposition of the 3D Tensor

---

- **THEOREM:** If we can decompose  $T_n$  into a sum of  $R$  rank-one terms, there exists an algorithm to calculate the multiplication of two  $n \times n$  matrices with  $R$  multiplications.
- In other words, an algorithm to calculate the multiplication of two  $n \times n$  matrices can be rewritten in the form of a sum of  $R$  rank-one terms (a decomposition).
- By decomposition of  $T_n$  into a sum of  $R$  rank-one terms, we mean

$$T_n = \sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$$

- $\otimes$  is the operation of the outer product.
- $u, v, w$  are three matrices
- $u^{(r)}, v^{(r)}, w^{(r)}$  are the  $r$ -th column of the  $u, v, w$  respectively, actually three vectors.
- $u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$  is a rank-one term, that is the outer product of  $u^{(r)}, v^{(r)}, w^{(r)}$
- I will give an example to illustrate this part in the following three slides.



# How to translate Strassen's algorithm to a Tensor Decomposition

---

- $m_1 = (a_1 + a_4) \cdot (b_1 + b_4)$
- $m_2 = (a_3 + a_4) \cdot b_1$
- $m_3 = a_1 \cdot (b_2 - b_4)$
- $m_4 = a_4 \cdot (-b_1 + b_3)$
- $m_5 = (a_1 + a_2) \cdot b_4$
- $m_6 = (-a_1 + a_3) \cdot (b_1 + b_2)$
- $m_7 = (a_2 - a_4) \cdot (b_3 + b_4)$
- $c_1 = m_1 + m_4 - m_5 + m_7$
- $c_2 = m_3 + m_5$
- $c_3 = m_2 + m_4$
- $c_4 = m_1 - m_2 + m_3 + m_6$



$$\sum_{r=1}^7 u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$$

That is to show there exists such  $u$ ,  $v$ , and  $w$ !

# How?

Strassen's Algorithm:

$$m_1 = (a_1 + a_4)(b_1 + b_4)$$

$$m_2 = (a_3 + a_4)b_1$$

$$m_3 = a_1(b_2 - b_4)$$

$$m_4 = a_4(b_3 - b_1)$$

$$m_5 = (a_1 + a_2)b_4$$

$$m_6 = (a_3 - a_1)(b_1 + b_2)$$

$$m_7 = (a_2 - a_4)(b_3 + b_4)$$

$$c_1 = m_1 + m_4 - m_5 + m_7$$

$$c_2 = m_3 + m_5$$

$$c_3 = m_2 + m_4$$

$$c_4 = m_1 - m_2 + m_3 + m_6$$

$$\mathbf{U} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix}$$

$$\mathbf{V} = \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix}$$

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix}$$

$m_1 \quad m_2 \quad m_3 \quad m_4 \quad m_5 \quad m_6 \quad m_7$

- How to comprehend this?
- $m_1 = (a_1 + a_4)(b_1 + b_4)$
- $m_1 = ([1, 0, 0, 1] \cdot [a_1, a_2, a_3, a_4])([1, 0, 0, 1] \cdot [b_1, b_2, b_3, b_4])$
- $m_1 = (u^{(1)} \cdot [a_1, a_2, a_3, a_4])(v^{(1)} \cdot [b_1, b_2, b_3, b_4])$
- $c_1 = [1, 0, 0, 1, -1, 0, 1] \cdot [m_1, m_2, m_3, m_4, m_5, m_6, m_7]$
- $c_1 = m_1 + m_4 - m_5 + m_7$
- Let us verify this u, v, and w work well in the following two slides.

# How to calculate $u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$

$$\begin{aligned}
 \mathbf{u}^{(1)} &= \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} \begin{matrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{matrix} \\
 \mathbf{v}^{(1)} &= \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{matrix} \\
 \mathbf{w}^{(1)} &= \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix} \\
 &\quad m_1 \quad m_2 \quad m_3 \quad m_4 \quad m_5 \quad m_6 \quad m_7
 \end{aligned}$$

$$\bullet \quad u^{(1)} \otimes v^{(1)} = [1, 0, 0, 1] \otimes [1, 0, 0, 1]$$

$$\bullet \quad u^{(1)} \otimes v^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\bullet \quad \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \otimes w^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \otimes [1, 0, 0, 1]$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$$

$$\mathbf{U} = \begin{pmatrix} u^{(1)} & u^{(2)} & u^{(3)} & u^{(4)} & u^{(5)} & u^{(6)} & u^{(7)} \\ 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}$$

+

+

+

+

+

+

$$\sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)} =$$

[illegible]

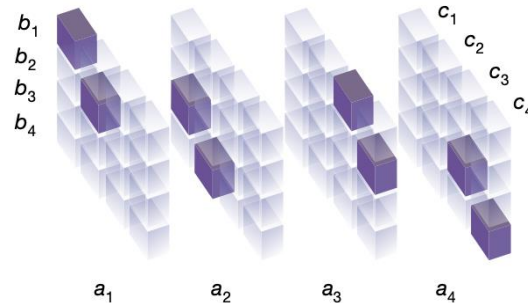
# Decomposition of the 3D Tensor

$$\bullet \sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$


- This example shows that  $T_n = \sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$
- This example also shows that an algorithm can be translated to a tensor decomposition.
- **If we can decompose  $T_n$  into a sum of  $R$  rank-one terms, there exists an algorithm to calculate the multiplication of two  $n \times n$  matrices with  $R$  multiplications.**
- **Finding a new tensor decomposition is equivalent to finding a new matrix multiplication algorithm**
- **Thus, our goal is to find the decomposition with minimum  $R$ .**

# Tensor Game

---

# Tensor Game

---

- If we can formulate the decomposition process to a single-player game, we can take advantage of the state of arts reinforcement algorithm!
- How to change the process of decomposition to a game?
- The state at timestep  $t$  is  $S_t$ .
- The initial state is  $S_0 = T_n$ .
- If  $x$  and  $y$  are both nonzero vectors, then the outer product matrix  $x \cdot y^T$  always has matrix rank 1.
- An action at timestep  $t$  is  $a_t = (u^{(t)}, v^{(t)}, w^{(t)})$ . We can get a rank-one term by taking the outer product of an action.  $u^{(t)}, v^{(t)}, w^{(t)}$  are randomly generated and should be non-zero.
- An update of the state of the game is  $S_t = S_{t-1} - u^{(t)} \otimes v^{(t)} \otimes w^{(t)}$
- When  $S_t = 0$ , we achieve the final state.  $0 = T_n - \sum_{r=1}^R u^{(r)} \otimes v^{(r)} \otimes w^{(r)}$
- Obviously, the 0 here is a 3D tensor where all entries are zero.
- The reward of each step is -1. (Thus, to achieve higher return, we need to minimize the number of step, i.e., minimize  $R$ ).

# Experiment



- After formulating the single-player game, we could leverage the power of reinforcement learning.
- In this paper, the authors used similar techniques to AlphaZero (a.k.a. AlphaGo 2)[5]



# Experiment Results

- Column  $(n, m, p)$  refers to the problem of multiplying  $n \times m$  with  $m \times p$  matrices.
- The complexity is measured by the number of scalar multiplications (or equivalently, the number of terms in the decomposition of the tensor).
- ‘Best rank known’ refers to the best known upper bound on the tensor rank (before this paper), whereas ‘AlphaTensor rank’ reports the rank upper bounds obtained with AlphaTensor, in modular arithmetic ( $\mathbb{Z}_2$ ) and standard arithmetic.

Size $(n, m, p)$	Best method known	Best rank known	AlphaTensor rank Modular	AlphaTensor rank Standard
(2, 2, 2)	(Strassen, 1969) <sup>2</sup>	7	7	7
(3, 3, 3)	(Laderman, 1976) <sup>15</sup>	23	23	23
(4, 4, 4)	(Strassen, 1969) <sup>2</sup> (2, 2, 2) $\otimes$ (2, 2, 2)	49	47	49
(5, 5, 5)	(3, 5, 5) + (2, 5, 5)	98	96	98
(2, 2, 3)	(2, 2, 2) + (2, 2, 1)	11	11	11
(2, 2, 4)	(2, 2, 2) + (2, 2, 2)	14	14	14
(2, 2, 5)	(2, 2, 2) + (2, 2, 3)	18	18	18
(2, 3, 3)	(Hopcroft and Kerr, 1971) <sup>16</sup>	15	15	15
(2, 3, 4)	(Hopcroft and Kerr, 1971) <sup>16</sup>	20	20	20
(2, 3, 5)	(Hopcroft and Kerr, 1971) <sup>16</sup>	25	25	25
(2, 4, 4)	(Hopcroft and Kerr, 1971) <sup>16</sup>	26	26	26
(2, 4, 5)	(Hopcroft and Kerr, 1971) <sup>16</sup>	33	33	33
(2, 5, 5)	(Hopcroft and Kerr, 1971) <sup>16</sup>	40	40	40
(3, 3, 4)	(Smirnov, 2013) <sup>18</sup>	29	29	29
(3, 3, 5)	(Smirnov, 2013) <sup>18</sup>	36	36	36
(3, 4, 4)	(Smirnov, 2013) <sup>18</sup>	38	38	38
(3, 4, 5)	(Smirnov, 2013) <sup>18</sup>	48	47	47
(3, 5, 5)	(Sedoglavic and Smirnov, 2021) <sup>19</sup>	58	58	58
(4, 4, 5)	(4, 4, 2) + (4, 4, 3)	64	63	63
(4, 5, 5)	(2, 5, 5) $\otimes$ (2, 1, 1)	80	76	76

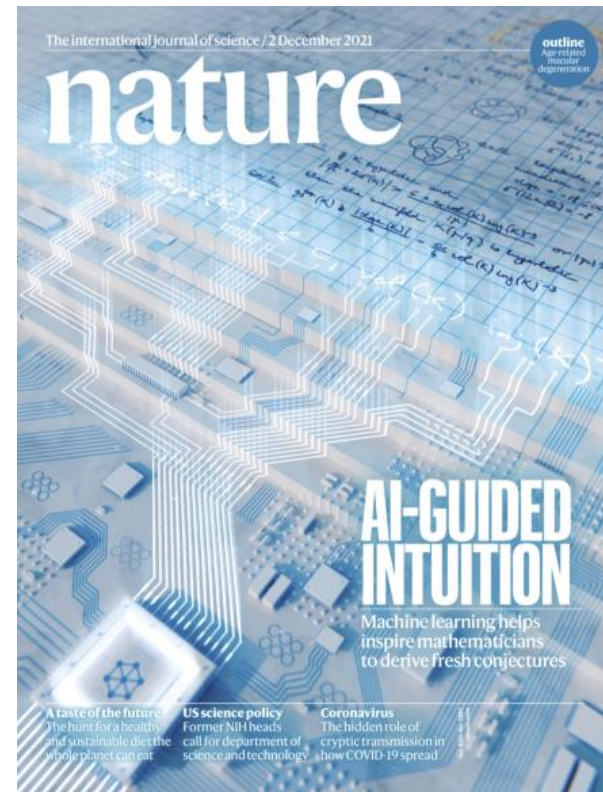
# Conclusion

---

- Contribution:
  - In all cases, AlphaTensor discovers algorithms that match or improve over the known state-of-the-art.
- Questions on This Work:
  - Reformulating a problem to a single-player game is not general.

# Discussion and My Thoughts

- For two  $5 \times 5$  matrices multiplication in  $\mathbb{Z}_2$ , Manuel Kauers and Jakob Moosbauer of Johannes Kepler University published a paper claiming they have reduced that count by one, down to 95 multiplications. [6]
- This means that AlphaTensor can guide the intuition of mathematicians, which coincides with the idea of another paper “Advancing mathematics by guiding human intuition with AI” I presented last time. This paper is also published by DeepMind on Nature.
- Even though the algorithms of reinforcement learning have not been improved a lot over the past five years, they can still make huge contribution if you can formulate your problem to a single-player game.



# Reference

---

- [1] Brian D. Hahn, Daniel T. Valentine, in Essential MATLAB for Engineers and Scientists (Seventh Edition), 2019
- [2] Devadas, Srini. “Latencies for Typical Modern Processor.” 18 Nov. 2022, Cambridge, Massachusetts Institute of Technology.
- [3] Fawzi, Alhussein, et al. “Discovering Novel Algorithms with AlphasTensor.” *DeepMind*, 5 Oct. 2022, <https://www.deepmind.com/blog/discovering-novel-algorithms-with-alphasensor>.
- [4] Axler, Sheldon (2015). Linear Algebra Done Right. Undergraduate Texts in Mathematics (3rd ed.). Springer. ISBN 978-3-319-11079-0.
- [5] Silver, David, et al. “A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play.” *Science*, vol. 362, no. 6419, 2018, pp. 1140–44, doi:10.1126/science.aar6404.
- [6] Kauers, Manuel, and Jakob Moosbauer. The FBHHRBNRSSHK-Algorithm for Multiplication in  $\mathbb{Z}_2^{5 \times 5}$  Is Still Not the End of the Story. arXiv, 2022, doi:10.48550/ARXIV.2210.04045.
- [7] Fawzi, A., Balog, M., Huang, A. et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* 610, 47–53 (2022). <https://doi.org/10.1038/s41586-022-05172-4>
- [8] Liang, Depeng. “An Introduction to AlphaTensor.” Zhihu, 16 Oct. 2022, [zhuanlan.zhihu.com/p/574170693](https://www.zhihu.com/question/574170693).
- [9] Kilcher, Yannic. “This Is a Game Changer! (AlphaTensor by DeepMind Explained).” YouTube, 7 Oct. 2022, [www.youtube.com/watch?v=3N3BI5AA5QU](https://www.youtube.com/watch?v=3N3BI5AA5QU).
- [10] Chen, Emma. “AlphaTensor by DeepMind Explained – Talk by Emma Chen at Harvard Medical AI Lab.” YouTube, 13 Oct. 2022, [www.youtube.com/watch?v=gpYnDIs4PdQ](https://www.youtube.com/watch?v=gpYnDIs4PdQ).

# Thanks



ANY QUESTIONS ABOUT THE  
PAPER?



ANY COMMENTS AND FEEDBACK  
FOR MY PRESENTATION?