Slide1: Hello, everyone. Today I'm going to talk about a paper called "Discovering Faster Matrix Multiplication Algorithm with Reinforcement Learning". This paper is also known as "AlphaTensor". There are two parts of this paper. One of the main contributions of this work is to formulate the matrix multiplication to a single-player game. My presentation today will focus on this part. This paper is one of the most exciting works I've read recently. I will try my best to present this work well, and I hope all of you can enjoy this work.

Slide2: There are five parts to today's presentation. The first part is a short introduction to the paper's background. The second part is the motivation for this work and what contribution they made in this work. The third part is about the Mathematics Behind Matrix Multiplication. The fourth part details how to formulate the problem into a single-player game (called the Tensor game in their paper) and some experimental results. The last part is the conclusion and contribution analysis of this work.

Slide3: This article is published in the famous journal Nature. This paper was published online on October 5th, 2022 (Only 6 weeks ago) and featured on the front cover of Nature issued on October 6^{th,} 2022. This article is openly accessible online, and you don't have to subscribe to nature. I attached the link below. If you are interested, please feel free to check it.

Slide4: There are several authors of this work. They are all from DeepMind. There are five authors who contributed equally to this work. To give full credit to them, I attached their photo below.

Slide5:
Matrix multiplication is probably the most important matrix operation. It is used widely in network theory (Use the adjacency matrix to represent a finite graph), the solution of the system of linear equations (Newton's method to solve a system of linear equations), the transformation of coordinate systems, and population modelling.

Let me give you two common examples. All of us have used the machine learning model to predict something. When we input a batch of data into the model, it needs many matrix multiplications to get the prediction. Another example is the autopilot system. We may input some information from the sensors into the model, and the model should output the next action the car should take. Since the driving system should be real-time, we need instantaneous responses. Faster matrix multiplication, in this case, could give us a faster response.

Slide6:
Before we discuss the details of the paper, let us first recap how to do the matrix multiplication. Here I used the matrix multiplication of two 2*2 matrices as an example.

用激光笔对着例子解释一下

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, and I will explain the reason in the following slides.
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 needs R multiplications (because we need to take the inner product of the row of the first matrix and column of the second matrix).

It seems like we cannot do anything to accelerate this process. Actually, from the perspective of mathematics, using this standard process is straightforward enough. However, we still have tricks to accelerate this calculation in computers.

Slide7:
The magic here is that computers have various speeds for calculating additions and multiplications. Let us take the example of 64 bits addition and 64 bits multiplication. The latency of the CPU is counted by cycles. 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. For 64 bits additions, it typically needs only one cycle. In contrast, we will need 20 cycles for 64 bits multiplications.

Slide8:
With this idea in mind, an intuitive idea is that the calculation process can be accelerated if we can use additions to substitute the multiplications.

激光笔解释一下这个例子

As you can see, the two methods above are mathematically equivalent. However, as computer scientists, we prefer the second one since it uses fewer multiplications.

Slide9:
By bringing the similar idea to matrix multiplication, the German mathematician Volker Strassen discovered an algorithm called Strassen’s algorithm to do the matrix multiplication of two 2*2 matrices. Intuitively, Strassen's algorithm looks much more complicated than the standard one. But let us carefully analyze these two algorithms. Here, each m includes one multiplication. The standard algorithm includes 8 multiplications and 4 additions, namely 164 cycles to compute.
On the other hand, Strassen’s algorithm includes 7 multiplications and 18 additions. That is 158 multiplications. Even though Strassen’s algorithm looks much more complex, it can save 6 cycles for computing.

Slide10:
Unfortunately, Even though the trick that uses additions instead of multiplications is a good way to accelerate calculation, it seems hard 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 as small as 3x3.

Given these facts, we would ask a question. 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 we could minimize the number of multiplications used to calculate the inner product, we can achieve our goal!

Slide11:

The main contribution of this work is that in all cases, AlphaTensor discovers algorithms that match or improve over the known state-of-the-art. Meanwhile, AlphaTensor is not limited to certain sizes of the matrices, and it can be extended to explore the algorithm for matrices multiplications with any size.

Slide13:

The first step for reformulating the matrix multiplication is using a 3D tensor to represent a matrix multiplication.

右上角 A tensor has a dimension of three. Here we use the a, b, and c to represent the three axes. The a-axis is the horizontal one. B-axis is the vertical one, and the c-axis is the direction into the plane.

矩阵乘法定义和图 This is the standard process of multiplying two 2*2 matrices. The first matrix has four entries, namely a1 to a4. The second matrix has four entries, namely b1 to b4. The resulting matrix has four entries as well, namely c1 to c4. Therefore, as we can see in the graph of the corresponding tensor, each dimension has 4 entries. Each entry can only be zero or one. We use purple to denote the entries with a value of 1 and semi-transparent to denote the entries with a value of 0.

The last question is why the multiplication of two 2*2 matrices can be transformed to the tensor shown here. Actually, The tensor specifies which entries from the input matrices to read, and where to write the result.

For example, let me take the c1 as an example. C1 is equal to a1 times b1 plus a2 times b3. So we denote the entries at (a1, b1, c1) and (a2, b3, c1) as 1.

If you look at the tensor graph, you should check the last row in the back for c1. You will notice here we have two purple entries. 圈一下 Thatis a1 b1 and a2 b3. To help you better under this

part, I expand the 3D tensor into a planar graph. Again, since c1 is equal to a1 times b1 plus a2 times b3, we denote the entries at (a1, b1, c1) and (a2, b3, c1) as 1 here.

Before we go to the next slide, any questions here?

Prof Liu's comments: We can add a road map here to link each part to the whole architecture of RL. For example, which part is the reward and which part is the state.

Slide14:
As we see in the previous slide, the multiplication of two 2 × 2 matrices can be rewritten to a 3D tensor with a dimension of 4 × 4 × 4.

In general, for multiplication of two n × n matrices, it can be rewritten to a 3D tensor with n square × n square × n square.

More generally, for the multiplication of two matrices with the dimension of n × m and m × p, it can be rewritten to a 3D tensor with (n × m) × (m × p) × (n × p).

For simplicity, we would use the case of multiplication of two n × n matrices. The corresponding 3D tensor with a dimension of n square × n square × n square is denoted as Tn.

Prof Liu's comments: Instead of using bottom up, we'd better to use top down to present this paper. Before recaping the definition of rank and outproduct, we should explain why we need to know rank and outproduct first. Othwise, the audient will get lost. Moreover, we should always stick with the same example we used from the beginning. Otherwise, the audient will get lost. The order of the presentation can be 1. Example 2. Our goal 3. Architecture with RL.

Slide15:
Before we go to the definition of tensor decomposition, let us quickly recap some knowledge from the 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.

Similarly, the rank of a 3D tensor is the dimension of the matrix space generated (or spanned) by its "layers" 圈一下，一个矩阵就是一个 layer

The outer product of u and v is equivalent to a matrix multiplication u times transpose of v.

I illustrate a very naïve example here.

Slide16:
The main argument in this paper is that if we can decompose $\boldsymbol{Tn}$ into a sum of R rank-one terms, there exists an algorithm to calculate the multiplication of two n × n matrices with R multiplications.

In other words, an algorithm to calculate the multiplication of two n × n matrices can be rewritten in the form of a sum of R rank-one terms (a decomposition).

I know that this may not make sense to everybody. I would explain this part step by step. I will first elaborate on the details of this argument. And then, use the following slides to show an example to help you better understand this argument.

The cross with a circle is the operation of the outer product. u, v, w are three matrices. u(r), v(r), and w(r) are the r-th column of the u, v, and w, respectively, actually three vectors. The outer product of u(r), v(r), and w(r) is the rank-one term.

Slide17:
To give an intuitive illustration, I will show an example that translates Strassen's algorithm into a tensor decomposition. That is, we will find the u, v and w.

Slide18:
Each row of the matrices u and v corresponds to the a1 to a4 and b1 to b4. Both u and v have 7 columns because we have 7 m here, from m1 to m7. M1 is equal to (a1 + a4) (b1 + b4). Since a1 + a4 is equal to the inner product of (1, 0, 0, 1) and (a1, a2, a3, a4), the first column of the matrix u is (1, 0, 0, 1). Similarly, since (b1 + b4) is equal to the inner product of (1, 0, 0, 1) and (b1, b2, b3, b4), the first column of the matrix v is (1, 0, 0, 1).

Then when it comes to the c1, we look at the row of w. C1 is equal to m1 + m4 – m5 + m7, that is, the inner product of (1, 0, 0, 1, -1, 0, 1) and (m1, m2, m3, m4, m5, m6, m7). Consequently, we write (1, 0, 0, 1, -1, 0, 1) to the first row of w.

By applying this idea to all, we can write down the whole u, v, w.

Then, let us verify that this u, v and w indeed give us the tensor corresponding to the matrix multiplication of two 2*2 matrices.

Prof Liu's comments: We can explain this part from the backward order. We can say that we first get the u, v, w from the reinforcement learning algorithm. Then, how we can conclude the Strassen's algorithm from the u, v, w. This backward order will be more intuitive for the audients.

Slide19:
Let us see how to calculate the outer product of three vectors. Here I use the u(1), v(1) and w(1) as an example. The outer product of u(1) and v(1) is equal to u(1) multiply the transpose of v(1). 写一下. Then we can have a matrix with dimension 4 * 4. After that, we could calculate the outer product of this matrix and w(1). The result would be a 3D tensor. The first layer of the tensor is equal to the matrix multiply 1, and the second layer is equal to the matrix multiply 0. Similarly, we could get layer 3 and layer 4. The result of the outer product of u(1), v(1) and w(1)

is the 3D tensor shown here. As you may notice, these four layers are linearly dependent, which means this tensor has rank one.

Prof Liu's two questions:

1. How the 7 multiplications corresponds to the 7 additions in the tensor step?
2. How the additions of c1 to c4 corresponds to the tenstor space?

Slide20:

In this slide, I showed the whole process of calculating the outer product of u(i), v(i) and w(i) where i starts from 1 to 7. After that, we sum all of them together. The addition follows the similar rule to the matrix addition, where we add the corresponding entries together.

随便用个什么 entry 来举个例子

The result is shown here.

Slide21:

As you may notice, the sum of these R rank-one terms is exactly the same as the Tensor I introduced before. Through this tiny example, you should be convinced that actually a decomposition of the 3D tensor is an algorithm to calculate the corresponding matrix multiplication.

With this idea, if we can decompose $\boldsymbol{Tn}$ into a sum of R rank-one terms, there exists an algorithm to calculate the multiplication of two n × n matrices with R multiplications.
Thus, our goal is to find the decomposition of the tensor with a minimum R.

Slide23:

If we can formulate the decomposition process into a single-player game, we can take advantage of the state of arts reinforcement algorithm!

We denote the state of the game at timestep t as St.
The initial state S0 is the 3D tensor we want to decompose.

Since from the linear algebra class we know that If x and y are both nonzero vectors, then the outer product matrix $x \cdot y$^T always has matrix rank 1.

The action we could take at each timestep is ut, vt, and wt. 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 to use the current state to subtract the rank-one term generated by taking the outer product of an action. When St =0, we achieve the final state.

解释一下原因，因为 tensor 减 sum 是 0.
0 显然不是数字 0，而是一个 entry 全是 0 的 tensor
The reward of each step is -1. (Thus, to achieve a higher return, we need to minimize the number of steps, i.e., minimize R).

Slide25:
After formulating the single-player game, we could leverage the power of reinforcement learning. In theory, we could apply different reinforcement learning algorithms here. In this paper, the authors exploited similar techniques to AlphaZero (which is also known as AlphaGo 2). They did some little modifications based on AlphaZero, but those modifications are not the main contribution of their work. Therefore, I will not go through the details of this part.

Slide27:
This slide shows the results of the experiment. The first column refers to the size of the matrix multiplication. The tuple (n, m, p) refers to the problem of multiplying n × m with m × p matrices. The complexity is measured by the number of scalar multiplications (that is 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 (mod 2) and standard arithmetic. The red entries denoted where the AlphaTensor finds an algorithm with better performance than existing algorithms.

Slide28:
The main contribution of this work is that in all cases, AlphaTensor discovers algorithms that match or improve over the known state-of-the-art. Meanwhile, AlphaTensor is not limited to certain sizes of the matrices, and it can be extended to explore the algorithm for matrices multiplications with any size.

However, the limitation of this work is also evident. The process of reformulating the problem into a single-player game is not general. For other problems, it might be hard for us to find a way to formulate them into a single-player game.

Slide29:
Additionally, I shared another news in our Saturday group meeting. For two 5 × 5 matrices multiplication in mod 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. This means that the AlphaTensor did not reach the end of the story. Instead, its result can guide the intuition of human mathematicians to improve further. This 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.

Beyond that, this paper also gives us a clue that Even though the reinforcement learning algorithms have not been improved a lot over the past five years, they can still make a huge contribution if you can formulate your problem into a single-player game

Slide30: Here're the references for today's presentation. Since this is a very new paper, there are not many references online. I referred almost to all approachable materials. Please feel free to check them.

Slide31: That's all of today's presentation. If you have any questions about the paper or comments on my presentation, please feel free to discuss them with me.