

# AlphaTensor 详解

## 基于强化学习发现更快的矩阵乘法算法

小组：李致远、冯文喆、魏睿、宋泽顷、周启民

同济大学物理科学与工程学院

2025 年 12 月 3 日



- ① 背景介绍
- ② 核心原理
- ③ 实验结果
- ④ 总结与展望



- ① 背景介绍
- ② 核心原理
- ③ 实验结果
- ④ 总结与展望



# 为什么选择这篇文章？

- **核心地位**：矩阵乘法是深度学习（全连接层、Conv 层）和科学计算的基石。
- **优化维度**：
  - *System* 角度：向量化、访存优化（Cache 命中）、并行计算。
  - *Math* 角度：减少数值乘法的计算次数（乘法开销  $\gg$  加法开销）。

## Strassen 算法的启示 (1969)

对于  $2 \times 2$  矩阵乘法：

- **朴素算法**：需要 **8 次乘法** ( $O(N^3)$ )。
- **Strassen 算法**：只需要 **7 次乘法** ( $O(N^{2.807})$ )。

这意味着通过改变算法流程，可以在数学层面实现加速。

# 为什么关注矩阵乘法？

- **核心地位**：矩阵乘法是深度学习（全连接层、Conv 层）和科学计算的基石。
- **优化维度**：
  - *System* 角度：向量化、访存优化（Cache 命中）、并行计算。
  - *Math* 角度：减少数值乘法的计算次数（乘法开销  $\gg$  加法开销）。

## Strassen 算法的启示 (1969)

对于  $2 \times 2$  矩阵乘法：

- **朴素算法**：需要 8 次乘法 ( $O(N^3)$ )。
- **Strassen 算法**：只需要 7 次乘法 ( $O(N^{2.807})$ )。

这意味着通过改变算法流程，可以在数学层面实现加速。

# AlphaTensor 一句话介绍

## 核心理念

利用人工智能技术（强化学习 RL）对 \*\* 非深度学习的传统算法（矩阵乘） \*\* 流程进行自动设计。

- **自动设计**：将算法流程参数化，转化为参数搜索问题。
- **优化目标**：算法的时间复杂度（数值乘法次数）或实际运行时间。
- **方法论**：类比 AutoML，但这是 "**AutoAlgo**" ——自动发现经典算法。

- ① 背景介绍
- ② 核心原理
- ③ 实验结果
- ④ 总结与展望



# 搜索空间的构建：两个对应关系

AlphaTensor 将寻找算法转化为一个 **3D 张量分解** 游戏。

对应关系 ①：矩阵乘法定义  $\leftrightarrow$  表征张量  $T_n$

- 一种尺寸的矩阵乘法定义（如  $2 \times 2$ ）唯一对应一个三维张量  $T_n$ 。
- 张量中的元素为 0 或 1，代表结果矩阵中位置的值由哪些输入元素相乘得到。

对应关系 ②：低秩分解  $\leftrightarrow$  算法流程

- 表征张量的 **秩 (Rank)** = 算法所需的 **乘法次数**。
- 将  $T_n$  分解为  $R$  个秩-1 张量的和：

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



# 强化学习建模 (RL Formulation)

- **游戏目标**: 用尽可能少的步数 (秩-1 张量) 将初始张量  $T_n$  减为零张量。
- **状态 (State)**: 当前剩余的张量  $S_t$  (初始为  $T_n$ )。
- **动作 (Action)**: 选择三个向量  $u, v, w$  构成一个秩-1 张量。
  - 离散化: 系数限制在  $\{-2, -1, 0, 1, 2\}$  中, 剪枝搜索空间。
- **状态更新**:  $S_{t+1} \leftarrow S_t - u \otimes v \otimes w$

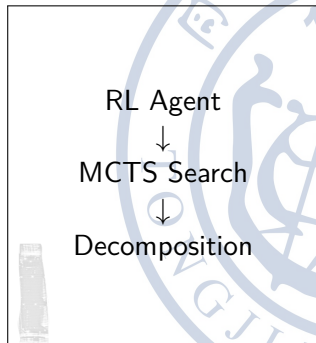
## 奖励函数 (Reward)

- 每走一步,  $\text{Reward} = -1$  (鼓励更少步数)。
- 若步骤超限未归零, 施加额外惩罚 (与剩余张量的秩相关)。
- 可选: 加入硬件运行时延时作为负奖励 (针对特定硬件优化)。

# 网络架构与搜索算法

## AlphaZero 风格的 RL + MCTS

- **MCTS (蒙特卡洛树搜索)**: 用于规划下一步动作, 平衡探索 (Explore) 与利用 (Exploit)。
- **Policy Network**: 基于 Transformer 架构。
  - 输入: 当前张量状态。
  - 输出: 建议的动作分布。
  - 包含 Cross-attention, Causal self-attention 等机制。
- **Value Network**: 预测当前状态归零所需的最小步数。



流程示意图

- ① 背景介绍
- ② 核心原理
- ③ 实验结果
- ④ 总结与展望



# 理论突破：发现更优的秩

表 1: AlphaTensor 在不同矩阵尺寸下的发现

| 矩阵尺寸 $(n, m, k)$ | 现有最佳秩 (Human)               | AlphaTensor |
|------------------|-----------------------------|-------------|
| 4, 4, 4          | 49 (Strassen <sup>2</sup> ) | <b>47</b>   |
| 5, 5, 5          | 98                          | <b>96</b>   |
| 4, 5, 5          | 80                          | <b>76</b>   |

- AlphaTensor 在多种尺寸下发现了比人类已知算法更少的乘法次数。
- 对于较大矩阵，其优势呈递增趋势。
- 注：** $3 \times 3$  的全局最优解仍是数学界的未解之谜，AlphaTensor 也未完全攻克。

# 实际应用：硬件加速

AlphaTensor 不仅理论上减少了计算次数，还能针对特定硬件 (GPU V100, TPU v3) 进行优化。

## Runtime 优化

- 通过将实际运行时间 (Timeit) 加入 Reward。
- 在大尺寸矩阵 ( $> 8192$ ) 上, AlphaTensor 发现的算法在 GPU/TPU 上均有显著加速。
- 超越了标准的 cuBLAS 库性能。
- 左图 (脑补): 在矩阵尺寸增大时, 加速比显著提升。
- 证明了 AI 发现的算法具有实际工程落地价值。

- ① 背景介绍
- ② 核心原理
- ③ 实验结果
- ④ 总结与展望



# 宏观定位与意义

- **AI for Science (Math/CS):**
  - 继 AlphaGo (围棋)、AlphaFold (蛋白质) 之后, DeepMind 在基础数学/理论计算机领域的突破。
- **AutoAlgo (自动算法设计):**
  - 与 AutoML (设计机器学习模型) 对应, AlphaTensor 展示了 AI 自动设计/优化经典算法的潜力。
- **未来展望:**
  - 是否可以推广到其他经典算法? (如数值计算方法)。
  - 难点在于如何为其他算法 (如 Dijkstra, DP) 建立类似的搜索空间 (Formulation)。

# 参考资料 I





# 人员分工

- 李致远：选题，演讲
- 冯文喆：背景调研与总结展望部分
- 魏睿：核心原理与实验部分
- 宋泽顷：做幻灯片
- 周启民：做幻灯片



*Thanks!*

