

AlphaTensor

Antoni Kowalczyk, prelekcja KNSI Golem
Warszawa, 27.10.2022

Wszystkie screeny
pochodzą z papera
dostępnego pod tym
linkiem:

[https://www.nature.com/a
rticles/s41586-022-05172](https://www.nature.com/articles/s41586-022-05172)

-4



Agenda

- O mnożeniu macierzy
- Algorytmy + setup
- AlphaZero
- Jak to działa?
- Wyniki
- Usprawnienia
- Inne zastosowania
- Losowe insighty z papera
- The FBHHRBNRSSSHK-Algorithm



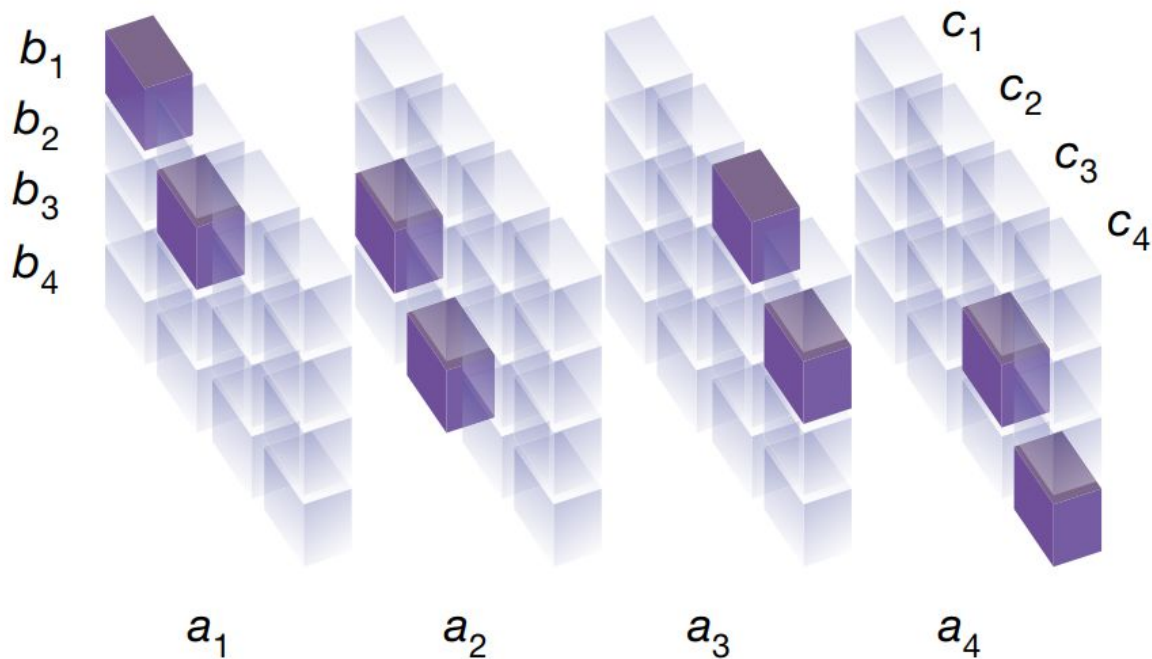
Mnożenie kosztuje
najwięcej

Cel: minimalizacja
liczby mnożeń

Jak zdefiniować mnożenie macierzy?

Tensorom!

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



Moment, ale jak w
ogóle da się
zmniejszyć liczbę
mnożeń???

Przykład: Algorytm Strassena do mnożenia macierzy 2x2 z macierzą 2x2

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

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

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

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

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

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

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

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

$$c_2 = m_3 + m_5$$

$$c_3 = m_2 + m_4$$

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

12.5% mniej!

Ilość wykonanych mnożeń = 7

Zadanie:

dekompozycja tensora

Wynik: algorytm

mnożenia macierzy

Dekompozycja

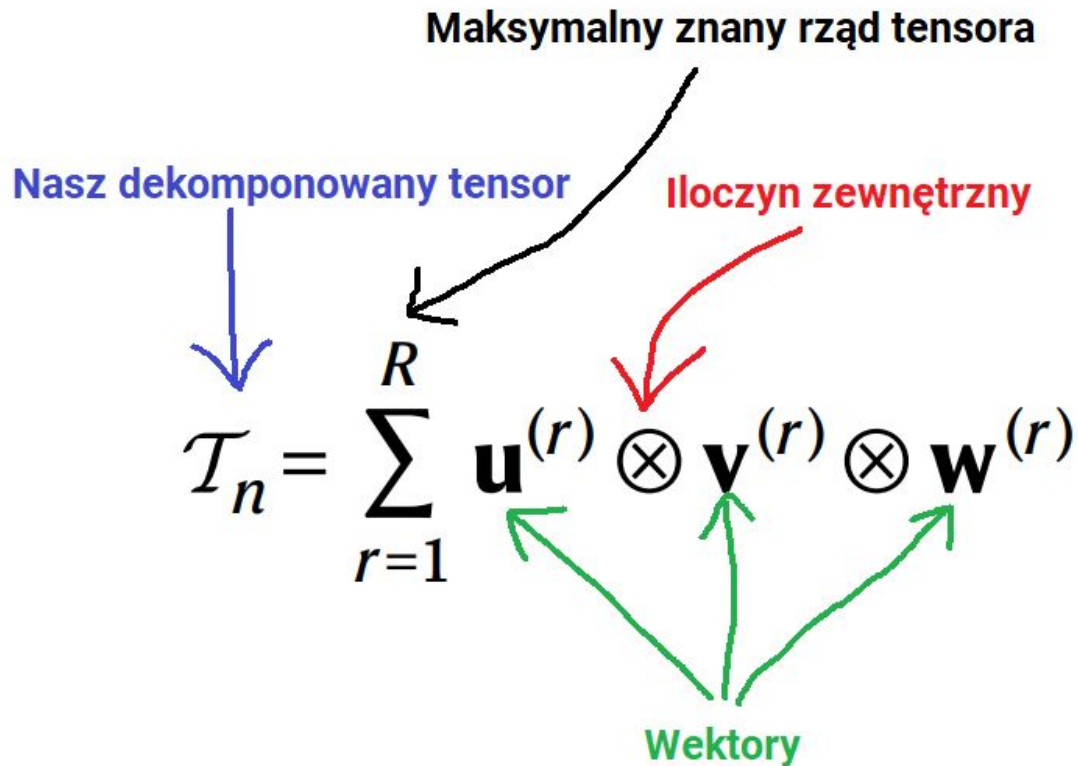
Maksymalny znany rząd tensora

Nasz dekomponowany tensor

Iloczyn zewnętrzny

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

Wektory



Dlaczego i po co?

Algorytm Strassena jako przykład dekompozycji tensora mnożenia macierzy

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

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

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

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

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

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

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

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

$$c_2 = m_3 + m_5$$

$$c_3 = m_2 + m_4$$

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

$$\mathbf{U} = \begin{matrix} & \begin{matrix} m1 & m2 & m3 & m4 & m5 & m6 & m7 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{pmatrix} & \begin{matrix} a1 \\ a2 \\ a3 \\ a4 \end{matrix} \end{matrix}$$

$$\mathbf{V} = \begin{matrix} \begin{pmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{pmatrix} & \begin{matrix} b1 \\ b2 \\ b3 \\ b4 \end{matrix} \end{matrix}$$

$$\mathbf{W} = \begin{matrix} \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} & \begin{matrix} c1 \\ c2 \\ c3 \\ c4 \end{matrix} \end{matrix}$$

Krótki pokaz jak to działa

Meta-algorytm wykorzystujący dekompozycję

Algorithm 1

A meta-algorithm parameterized by $\{\mathbf{u}^{(r)}, \mathbf{v}^{(r)}, \mathbf{w}^{(r)}\}_{r=1}^R$ for computing the matrix product $\mathbf{C}=\mathbf{AB}$. It is noted that R controls the number of multiplications between input matrix entries.

Parameters: $\{\mathbf{u}^{(r)}, \mathbf{v}^{(r)}, \mathbf{w}^{(r)}\}_{r=1}^R$: length- n^2 vectors such that
 $\mathcal{T}_n = \sum_{r=1}^R \mathbf{u}^{(r)} \otimes \mathbf{v}^{(r)} \otimes \mathbf{w}^{(r)}$

Input: \mathbf{A}, \mathbf{B} : matrices of size $n \times n$

Output: $\mathbf{C}=\mathbf{AB}$

(1) **for** $r=1, \dots, R$ **do**

(2) $m_r \leftarrow (u_1^{(r)}a_1 + \dots + u_{n^2}^{(r)}a_{n^2})(v_1^{(r)}b_1 + \dots + v_{n^2}^{(r)}b_{n^2})$

(3) **for** $i=1, \dots, n^2$ **do**

(4) $c_i \leftarrow w_i^{(1)}m_1 + \dots + w_i^{(R)}m_R$

return \mathbf{C}

Cel: dekompozycja
tensora sumą jak
najmniejszej ilości
tensorów o rzędzie 1

Krótko o Reinforcement Learningu

Podstawowe pojęcia

- Krok
- Nr kroku
- Akcja
- Przestrzeń akcji
- Stan
- Stan początkowy
- Stan terminalny
- Przestrzeń stanów
- Policy
- Wartość stanu
- Nagroda



AlphaZero

Co to



- Stworzony również przez DeepMind
- Założenie: bez wcześniejszej wiedzy osiągnąć nadludzkie możliwości w grach
- Używa Reinforcement Learningu do uczenia
- Uczy się za pomocą techniki self-play
- Sieć wypluwa dwie rzeczy
 - ewaluację stanu
 - rozkład prawdopodobieństwa, że dany ruch jest najlepszy dla każdego możliwego ruchu
- Dodatkowy mechanizm: Monte Carlo Tree Search (MCTS)

Osiągnięcia

- Nadludzki performance w grze Go
- Nadludzki performance w szachach
- Nadludzki performance w grze Shogi

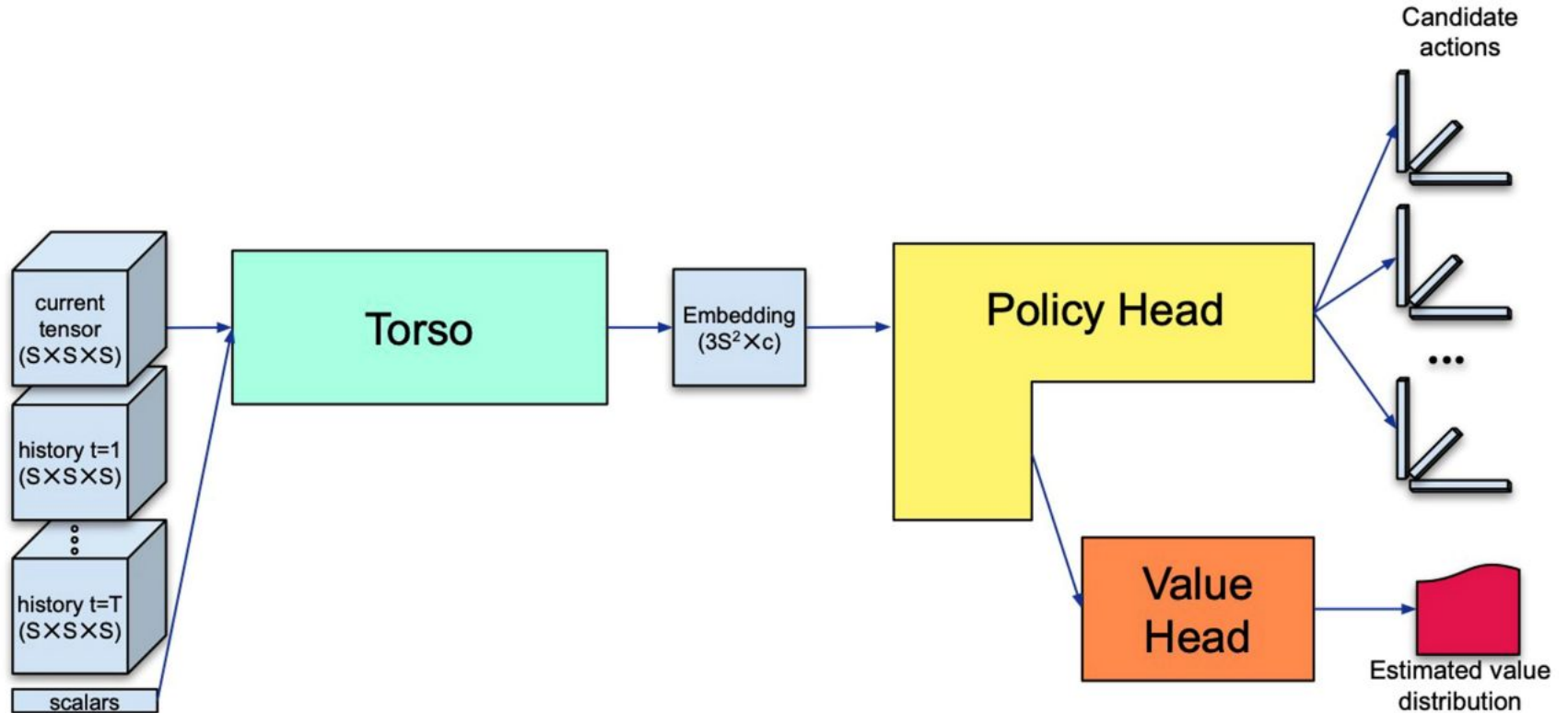


Pomysł: niech
dekompozycja będzie
grą!

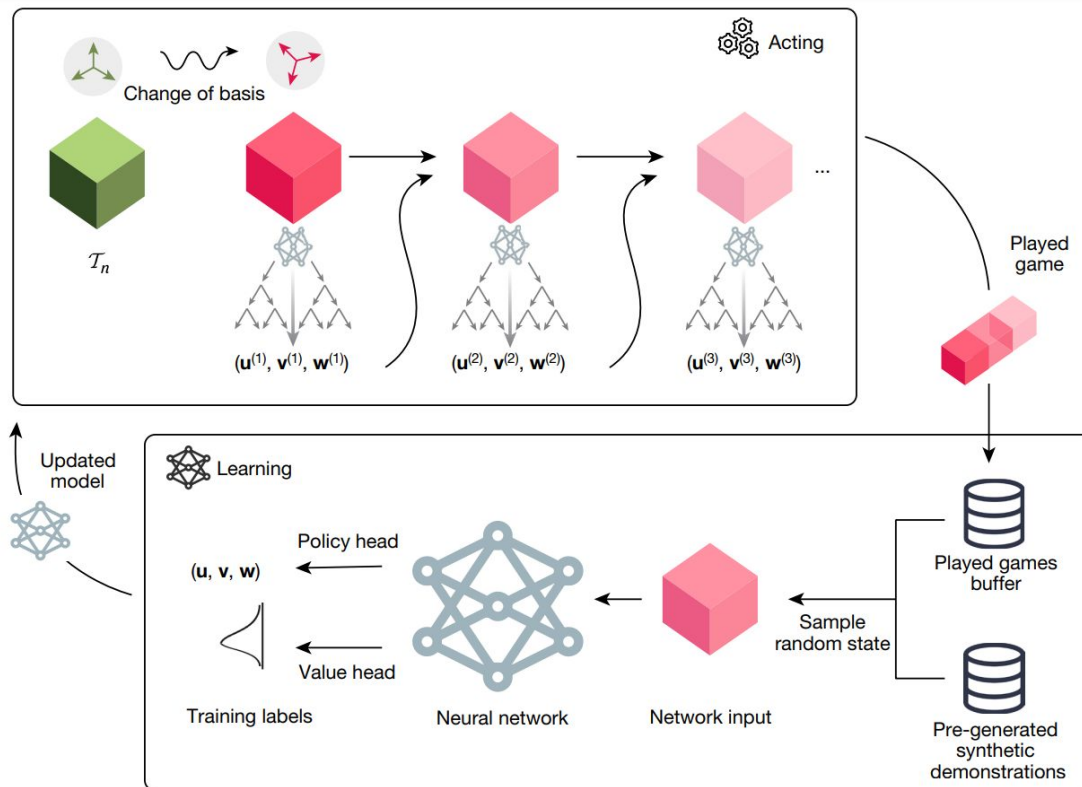
Definicje (każdy je lubi)

- akcja: wektory \mathbf{u} , \mathbf{v} , \mathbf{w} zwracane przez model
- przestrzeń akcji: wszystkie wektory o wielkości n^2
- stan po kroku t : tensor S_t
- stan na początku: $S_0 = T_n$
- krok: $S_t \leftarrow S_{t-1} - \mathbf{u}^{(t)} \otimes \mathbf{v}^{(t)} \otimes \mathbf{w}^{(t)}$
- Stan terminalny: $S_t = \mathbf{0}$ lub t_{limit}
- Nagroda: -1 za każdy krok + kara za osiągnięcie t_{limit} dla $S_t \neq \mathbf{0}$
- Składniki wektorów akcji należą do zbioru $\{-2, -1, 0, 1, 0\}$
- Wartość stanu: zwracana przez model

Zarys architektury



Zarys treningu

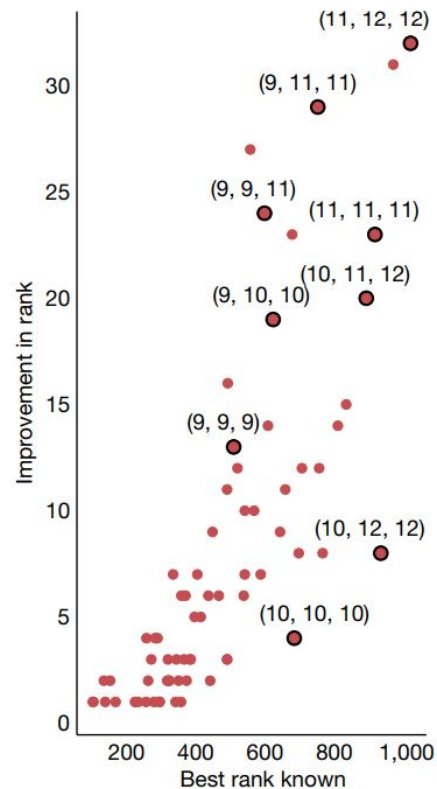


Wyniki

Mamy to! (czyli jest
progress)

Konkretnie

Size (n, m, p)	Best method known	Best rank known	AlphaTensor rank Modular Standard	
(2, 2, 2)	(Strassen, 1969) ²	7	7	7
(3, 3, 3)	(Laderman, 1976) ¹⁵	23	23	23
(4, 4, 4)	(Strassen, 1969) ² (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) ¹⁶	15	15	15
(2, 3, 4)	(Hopcroft and Kerr, 1971) ¹⁶	20	20	20
(2, 3, 5)	(Hopcroft and Kerr, 1971) ¹⁶	25	25	25
(2, 4, 4)	(Hopcroft and Kerr, 1971) ¹⁶	26	26	26
(2, 4, 5)	(Hopcroft and Kerr, 1971) ¹⁶	33	33	33
(2, 5, 5)	(Hopcroft and Kerr, 1971) ¹⁶	40	40	40
(3, 3, 4)	(Smirnov, 2013) ¹⁸	29	29	29
(3, 3, 5)	(Smirnov, 2013) ¹⁸	36	36	36
(3, 4, 4)	(Smirnov, 2013) ¹⁸	38	38	38
(3, 4, 5)	(Smirnov, 2013) ¹⁸	48	47	47
(3, 5, 5)	(Sedoglavic and Smirnov, 2021) ¹⁹	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



Zastosowane usprawnienia

Syntetyczne dane

- Dekompozycja jest NP-trudna, ale jej odwrotność już jest trywialna
- Losujemy R wektorów \mathbf{u} , \mathbf{v} i \mathbf{w} . Z nich tworzymy tensor

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

- Uzyskaliśmy zwykły zbiór treningowy do Supervised Learningu
- Więc dorzucamy do lossa błąd predykcji na tym zbiorze

W efekcie znacznie lepszy performance modelu niż trenując tylko na własnych grach lub tylko na danych syntetycznych

Zmiana bazy

- Przejście z kanonicznej na jakąś wylosowaną
- Losujemy na początku każdej gry
- Co mamy:
 - odpowiadające sobie tensory (rząd, dekompozycja)
- Po co:
 - **wincyj danych**
 - które są przy okazji znacznie bardziej zróżnicowane



Zmiana kolejności operacji

- Motywacja: sumujemy tensory rzędu jeden by uzyskać dekomponowany tensor
- Operacja sumowania jest przemienne
- Pomieszczajmy więc kolejność zestawów wektorów → **wincyj** danych → profit

Jeden model do wszystkich typów mnożenia

- Maksymalne rozmiary: 5×5 mnożone przez 5×5
- Niech model uczy się wszystkich mniejszych zestawów!
- Dla mniejszych macierzy: padding zerami

W efekcie: lepszy performance niż modele trenowane tylko na jednym “typie” mnożenia macieży

Optymalizacja pod runtime

Szybciej. Na
konkretnym sprzęcie

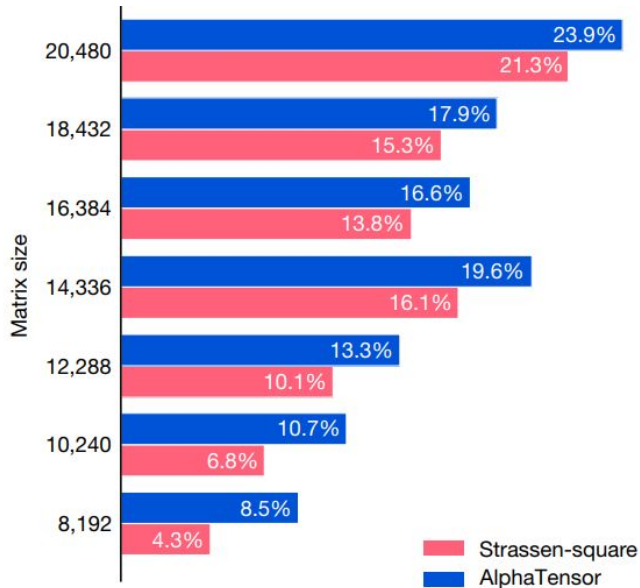
O co chodzi

- Różne sprzęty różnie liczą rzeczy
- Optymalizujemy pod czas obliczeń!

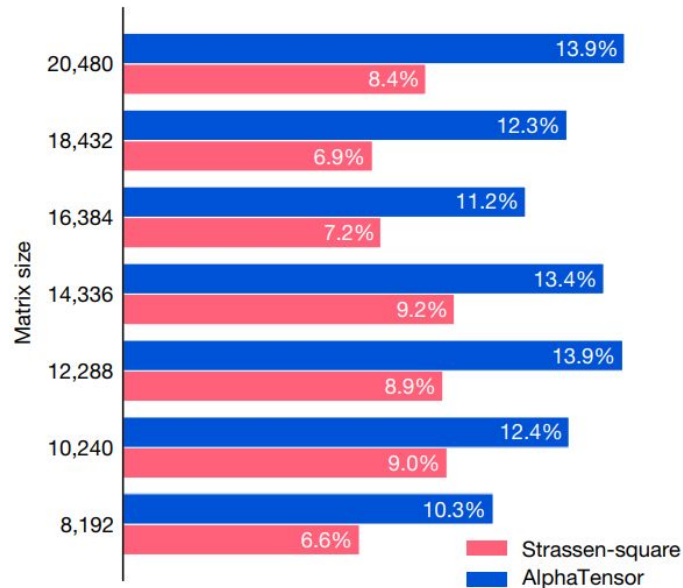
Jak to osiągnąć?

- Dodatkowy czynnik w nagrodzie
- minus czas wykonania na danym sprzęcie (razy jakiś hiperparametr)

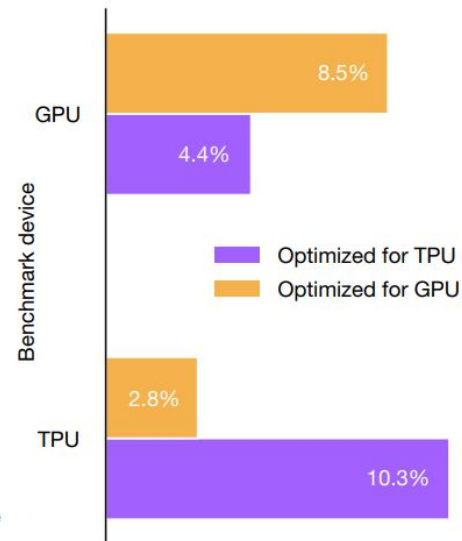
Wyniki



Speed-up on Nvidia V100 GPU



Speed-up on TPU v2



Speed-up of tailored algorithms on both devices

Insights

Istnieje **znacznie**
więcej algorytmów
mnożenia macierzy
(efektywnego)

14000

Tyle algorytmów mnożenia macierzy 4x4 przez 4x4 wypuła AlphaTensor

Każdy o rzędzie 49

Mnożenie macierzy skośno-symetrycznych przez wektor

50%

O tyle nowy algorytm (asymptotycznie) usprawnia taką operację

The FBHHRBNRSSHK-Algorithm

The FBHHRBNRSSHK-Algorithm for Multiplication in $\mathbb{Z}_2^{5 \times 5}$ is still not the end of the story

Manuel Kauers, Jakob Moosbauer

Download PDF

In response to a recent Nature article which announced an algorithm for multiplying 5×5 -matrices over \mathbb{Z}_2 with only 96 multiplications, two fewer than the previous record, we present an algorithm that does the job with only 95 multiplications.

Subjects: **Symbolic Computation (cs.SC)**; Computational Complexity (cs.CC)

Cite as: [arXiv:2210.04045](https://arxiv.org/abs/2210.04045) [cs.SC]

(or [arXiv:2210.04045v3](https://arxiv.org/abs/2210.04045v3) [cs.SC] for this version)

<https://doi.org/10.48550/arXiv.2210.04045> 

<https://arxiv.org/abs/2210.04045>

Dziękuję za uwagę