УДК 004.021-004.8
DOI: 10.25559/SITITO.18.202204.781-789

Оригинальная статья

Реализация (λ, μ)-свернутого произведения многомерных матриц средствами операции tensordot из библиотек для тензорной алгебры

Е. И. Гончаров
ФГБОУ ВО «Смоленский государственный университет», г. Смоленск, Российская Федерация
Адрес: 214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4
drbenvey1996@mail.ru

Аннотация

Алгебра многомерных матриц является удачной моделью данных для задач из самых разных предметных областей. Многие авторы описывали аппаратно-программные комплексы, реализующих алгебру многомерных матриц целиком или параллельные алгоритмы (λ, μ)-свернутого произведения для решения определенной задачи. Их реализации требует значительных трудозатрат, а использование – знание и установку определенный фреймворков и компиляторов. Тем не менее в процессе исследования часто бывает полезно «проверить» предположение на частных случаях, что, возможно, быстро даст контрпример. Статья посещена созданию такого инструмента для прикладных исследований. Python-программы легки в написании, благодаря синтаксису, и имеют обширный выбор разнообразных библиотек. Среди них особое место занимают библиотеки для быстрых вычислений: NumPy, CuPy, PyTorch, TensorFlow. Они могут обеспечить достаточную скорость вычислений как на CPU, так и на GPU, и обладают значительным функционалом для работы с многомерными объектами, хотя и не реализуют (λ, μ)-свернутого произведения. В статье показывается, что операция (0, μ)-свернутого произведения из алгебры многомерных матриц в точности совпадает с операцией tensordot(A, B, μ) из python-библиотек для тензорной алгебры. Статья содержит результаты эксперимента по сравнению скорости вычислений этой операции в разных библиотеках. Автор вводит параллельный алгоритм умножения многомерных матриц, сводящий (λ, μ)-свернутое произведение к последовательности (0, μ)-свернутых произведений, которые могут быть выполнены параллельно. Это фактически сводит сложную задачу к параллельному вызову функции tensordot. Статья содержит подробное описание программы, реализующей алгоритм и результаты ее тестирования в разных средах выполнения.

Ключевые слова: алгебра многомерных матриц, (λ, μ)-свернутое произведение, tensordot, NumPy, CuPy, PyTorch, TensorFlow

Конфликт интересов: автор заявляет об отсутствии конфликта интересов.

Для цитирования: Гончаров Е. И. Реализация (λ, μ)-свернутого произведения многомерных матриц средствами операции tensordot из библиотек для тензорной алгебры // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 4. С. 781-789. doi: https://doi.org/10.25559/SITITO.18.202204.781-789

Original article

Implementation of (λ, μ)-Convolution Product by Means of Tensordot Operation from Libraries for Tensor Algebra

E. I. Goncharov
Smolensk State University, Smolensk, Russian Federation
Address: 4 Przhevalsky St., Smolensk 214000, Russian Federation
drbenvey1996@mail.ru

Abstract

Multidimensional matrix algebra is a successful data model for problems from a variety of subject areas. Many authors have described hardware and software complexes that implement the algebra of multidimensional matrices in its entirety or parallel algorithms of (λ, μ)-convolution product to solve a specific problem. Their implementation requires considerable labor, and the use of knowledge and installation of certain frameworks and compilers. Nevertheless, in the process of research, it is often useful to "test" the assumption on special cases, which may quickly give a counterexample. The article is devoted to the creation of such a tool for applied research. Python-programs are easy to write, thanks to the syntax, and have an extensive selection of various libraries. Among them, a special place is occupied by libraries for fast calculations: NumPy, CuPy, PyTorch, TensorFlow. They can provide sufficient computing speed on both CPU and GPU, and have significant functionality for working with multidimensional objects, although they do not implement a (λ, μ)-convolution product. The article shows that the operation $(0, \mu)$-convolution product from the algebra of multidimensional matrices exactly coincides with the tensordot(A, B, μ) operation from python-libraries for tensor algebra. The article contains the results of an experiment comparing the calculation speed of this operation in different libraries. The author introduces a parallel algorithm for multiplying multidimensional matrices, reducing the (λ, μ)-convolution product to a sequence of $(0, \mu)$-convolution products that can be performed in parallel. This actually reduces a complex task to a parallel call of the tensordot function. The article contains a detailed description of the program implementing the algorithm and the results of its testing in different runtime environments.

Keywords: multidimensional matrix algebra, (λ, μ)-convolution product, tensordot, NumPy, CuPy, PyTorch, TensorFlow

Conflict of interests: The author declares no conflict of interest.

For citation: Goncharov E.I. Implementation of (λ, μ)-Convolution Product by Means of Tensordot Operation from Libraries for Tensor Algebra. *Modern Information Technologies and IT-Education.* 2022;18(4):781-789. doi: https://doi.org/10.25559/SITITO.18.202204.781-789

Введение

Авторы в работах [1] и [2] справедливо отмечают, что для наиболее эффективного решения задачи, программно-аппаратный комплекс должен полностью реализовывать алгебру, в которой описывается её математическая модель[1]. Алгебра многомерных матриц может быть использована как модель данных для решения задачи из самых разных предметных областей. В статье [3] авторы доказывают гомоморфизм между алгеброй многомерных матриц и реляционной алгеброй. Это породило новый подход к распараллеливанию запросов в базах данных. С ее помощью авторы в статье [4] вывели полиномиальные алгоритмы для вывода ассоциативных правил и маршрутизации. Некоторым матричным алгоритмам шифрования (шифр Хилла [5] и алгоритм Диффи-Хеллмана) удалось повысить криптостойкость при их обобщении на многомерный случай. Она удачно подходит для построения цепей Маркова [6] и решения задач на графах [7]. В статье [8] автор получает математическую модель операций свертки на основе алгебры многомерных матриц с операцией (0, μ)-свернутого произведения.

В статьях [2] и [9] авторы описывают различные подходы к построению программно-аппаратного комплекса, реализующего алгебру многомерных матриц. Тем не менее, их использование требует уставки и умения использовать дополнительных пакетов и фреймворков, а сама реализация требует команду высоко квалифицированных инженеров-программистов. Часто же в начале исследования хочется “прощупать” гипотезу на простых примерах, которые, возможно, сразу же дадут контрпример. Например, если в алгебре плоских матриц произведение определителей равно определителю произведения, а алгебра многомерных матриц является естественным обобщением алгебры плоских матриц на многомерных случай, то будет ли это утверждение верно для определителей многомерных матриц и каких-то разбиений? В таких ситуациях скоростные характеристики исследователю не важны, но хочется получать результаты без дополнительных настроек среды.

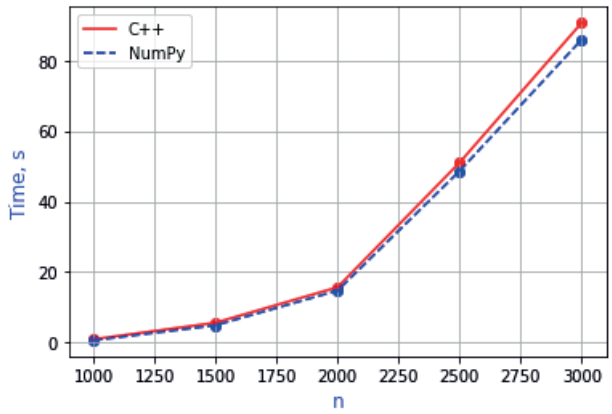
Цель исследования

В работе [10] авторы подробно описывают теоретические отличия тензорной алгебры от алгебры многомерных матриц. Тем не менее, библиотеки, её реализующие, обладают значительным функционалом по работе с многомерными массивами данных. Автор продолжает исследование, начатое в статье [11], и ставит целью создание программного комплекса на базе python-библиотек, реализующей операции тензорной алгебры, для реализации основных операций алгебры многомерных матриц. С его помощью исследователи смогут получать результаты без дополнительных настроек среды выполнения и его использование не будет требовать знания CUDA[2] и других фреймворков.

Скорость python-библиотек в сравнении с C++

Часто можно услышать, что python – интерпретируемый язык программирования[3] [12]. Поэтому программы, написанные на нем, будут работать медленно и для достижения скорости нужно использовать C++ [13]. Это правда только от части. Как правило математические пакеты написаны на C++, CUDA, Fortran и обеспечивают достаточную скорость вычислений, а сам python в таком случае служит лишь «языком интерфейсом» [14]. Проведем эксперимент: сравним скорость умножения плоских матриц для программы[4], написанной на C++ с использованием MPI [15] и алгоритма Кэннона[5] и программы написанной на python, умножающий эти же матрицы с помощью вызова функции $\mathrm{numpy.dot}(A, B)$ [6] [16-18]. Эксперимент производился на компьютере с 16 ГБ ОЗУ и процессором Ryzen™ 7 2700X.

На оси X рисунка 1 указаны размеры квадратных матриц A и B, на оси Y – необходимое время для выполнения операции. Точки, лежащие на штрихованной (основной) ломанной, показывают время в секундах произведения матриц A и B, выполняемого средствами NumPy (C++ с использованием MPI и алгоритма Кэннона). Как видно из рисунка 1, $\mathrm{numpy.dot}(A, B)$ работает быстрее, а код программы в сотни раз короче и приятнее к прочтению.



Р и с. 1. NumPy vs C++ (CPU)

F i g. 1. NumPy vs C++ (CPU)

[1] Захаров В. Н., Мунерман В. И. Параллельная реализация обработки интенсивно используемых данных на основе алгебры многомерных матриц // Аналитика и управление данными в областях с интенсивным использованием данных: XVII Международная конференция DAMDID/RCDL ; под ред. Л. А. Калиниченко, С. О. Старкова. Обнинск : НИЯУ МИФИ, 2015. С. 217-223.

[2] Технология программирования CUDA : учеб. пособие / Д. Н. Тумаков, Д. Е. Чикрин, А. А. Егорчев, С. В. Голоусов. Казань : КФУ, 2017. 112 с. URL: https://kpfu.ru/portal/ias_utils.file_download?p_table_id=4&p_file=F1135783586/Tumakov_D_N_i_dr_Tekhnologiya_programmirovaniya.pdf (дата обращения: 28.09.2022).

[3] O'Connor T. J. Violent Python. Waltham, MA : Syngress, 2013. 262 p. doi: https://doi.org/10.1016/C2011-0-07761-X

[4] Cannon's Matrix Multiplication Algorithm using MPI [Электронный ресурс] // GitHub repository. GitHub, 2018. URL: https://github.com/andadiana/cannon-algorithm-mpi (дата обращения: 28.09.2022).

[5] Cannon L. E. A cellular computer to implement the Kalman filter algorithm : Electronic Theses and Dissertations. Bozeman, MT, USA : Montana State University, 1969. Order Number: AAI7010025. URL: https://scholarworks.montana.edu/xmlui/bitstream/handle/1/4168/31762100054244.pdf (дата обращения: 28.09.2022).

[6] NumPy [Электронный ресурс]. URL: https://numpy.org (дата обращения: 28.09.2022).

Разумеется, из этого не следует, что программы, написанные на NumPy, принципиально предпочтительнее для быстрых вычислений. Целью эксперимента было не сравниваться скорость вычисления C++ и NumPy, а показать, что последний обладает достаточной скоростью выполнения операций из коробки. Операция умножения плоских матриц была выбрана не случайно: она является частным случаем (λ, μ)-свернутого произведения – (0, 1)-свернутое произведение, и частным случаем tensordot(*A*, *B*, μ) – tensordot(*A*, *B*, 1). Связь этих операций из алгебры многомерных матриц и тензоров будет описана ниже и является основой разрабатываемого программного комплекса.

(λ, μ)-свернутое произведение

Пусть матрицы $A=\left\|A_{i_1\ldots i_p}\right\|$ и $B=\left\|B_{i_1\ldots i_q}\right\|$, *p*- и *q*-мерные соответственно. Совокупности индексов этих матриц $i_1,\ldots,i_p$ и $i_1,\ldots,i_q$ разбиваются на четыре группы, содержащие соответственно κ, λ, μ и ν индексов ($\kappa,\lambda,\mu,\nu\geq 0$). Причем $\kappa+\lambda+\mu=p$, а $\lambda+\mu+\nu=q$. Для полученных групп индексов используются обозначения: $l=(l_1,\ldots,l_\kappa)$, $s=(s_1,\ldots,s_\lambda)$, $c=(c_1,\ldots,c_\mu)$ и $m=(m_1,\ldots,m_\nu)$. Тогда матрицы *A* и *B* можно представить в виде $A=\left\|A_{lsc}\right\|$ и $B=\left\|B_{csm}\right\|$. Индексы разбиения *c* называются кэлиевыми, *s* – называются скоттовыми, а индексы разбиения *m* и *l* – свободными. Матрица $C=\left\|C_{lsm}\right\|$, элементы которой вычисляются по формуле $C_{lsm}=\sum_{(c)} A_{lsc}\times B_{csm}$, называется (λ, μ)-свернутым произведением матриц *A* и *B*. Обозначается: ${}^{\lambda,\mu}(A\times B)=C$. (λ, μ)-свернутое произведение многомерных матриц – центральная операция алгебры многомерных матриц[7]. Вычислительная сложность такой операции $O(N^{\lambda+\mu+\kappa+\nu})$ Н. П. Соколов в работе[8] рассматривает только многомерные матрицы, где $i_1,\ldots,i_{(\max(p,q))}=1,\ldots,N$, где N – порядок многомерной матрицы. Тем не менее, для операции (λ, μ)-свернутого произведения достаточно, чтобы попарно совпадали порядки из кэлиевых и скоттовых групп индексов [19].

Выражение (λ, μ)-свернутого произведения через последовательность (0, μ)-свернутых произведений

Пусть $A=\left\|a_{i_{sc}}\right\|$ *p*-мерная матрица. За $S_i=(s_{i_1},\ldots,s_{i_\lambda})$ обозначим зафиксированный набор скоттовых индексов. Всего таких наборов N^λ. Каждому такому набору сопоставим $(p-\lambda)$-мерную матрицу $\widetilde{A}^S_{S_i}=\left\|\left(\widetilde{a}^S_{S_i}\right)_{lc}\right\|=\left\|a_{lS_ic}\right\|$, которая является S_i-тым скоттовым сечением многомерной матрицы *A*. Многомерную матрицу, обладающую скоттовыми индексами, можно представить в виде группы ее скоттовых сечений: $D^S(A)=\left\{\widetilde{A}^S_{S_i}\right\}_{i=1}^{N^\lambda}$. Верно и обратное: по группе скоттовых сечений можно восстановить исходную многомерную матрицу: $A=\left(D^S\right)^{-1}\left(\left\{\widetilde{A}^S_{S_i}\right\}_{i=1}^{N^\lambda}\right)$.

Лемма 1.

$${}^{(\lambda,\mu)}(A\times B)=\left(D^S\right)^{-1}\left(\left\{{}^{(0,\mu)}\left(\widetilde{A}^S_{S_i}\times\widetilde{B}^S_{S_i}\right)\right\}_{i=0}^{N^\lambda-1}\right).$$

Доказательство леммы 1.

$\forall i=1,\ldots,N^\lambda$ верно, что ${}^{(0,\mu)}\left(\widetilde{A}^S_{S_i}\times\widetilde{B}^S_{S_i}\right)=\widetilde{C}^S_{S_i}=\left\|\left(\widetilde{c}^S_{S_i}\right)_{lm}\right\|$, где $\widetilde{c}^S_{S_i}=\sum_{\vec{n}}\widetilde{a}^S_{S_i}\times\widetilde{b}^S_{S_i}=\sum_{\vec{n}}a_{lS_ic}\times b_{cS_im}=c_{lS_im}$.

Тогда $\left(D^S\right)^{-1}\left(\left\{{}^{(0,\mu)}\left(\widetilde{A}^S_{S_i}\times\widetilde{B}^S_{S_i}\right)\right\}_{i=1}^{N^\lambda}A\right)=B\left(D^S\right)^{-1}\left(\left\{{}_{lS_im}\right\}_{i=1}^{N^\lambda}\right)=\left\|{}_{lsm}\right\|={}^{(\lambda,\mu)}(\times)$.

Что и требовалось доказать [11].

Разные варианты идей работы со сотовыми сечениям для выполнения (λ, μ)-свернутого произведения предлагались авторами в статьях [9], [20-22]. Тем не менее, с помощью выше доказанной леммы появляется возможность сводить (λ, μ)-свернутое произведения к параллельной последовательности (0, μ)–свернутых произведений над матрицами меньшей размерности.

Пример применения леммы 1. Выполнить (2, 1)-свернутое произведение матриц *A* и *B*, где

$$A=\left\|\begin{array}{cc|cc}a_{0000}&a_{0001}&a_{0100}&a_{0101}\\a_{0010}&a_{0011}&a_{0110}&a_{0111}\\\hline a_{1000}&a_{1001}&a_{1100}&a_{1101}\\a_{1010}&a_{1011}&a_{1110}&a_{1111}\end{array}\right\|=\left\|\begin{array}{cc|cc}1&2&3&2\\0&-1&4&1\\\hline-2&0&0&1\\1&3&2&1\end{array}\right\|$$

и

$$B=\left\|\begin{array}{cc|cc}b_{000}&b_{001}&b_{100}&b_{101}\\b_{010}&b_{011}&b_{110}&b_{111}\end{array}\right\|=\left\|\begin{array}{cc|cc}1&2&1&3\\-2&0&4&1\end{array}\right\|.$$

${}^{2,1}(A\times B)=C=\left(D^S\right)^{-1}\left(\left\{\widetilde{C}^S_{S_i}\right\}_{i=0}^{3}\right)$, где

$$\widetilde{C}^S_{00}={}^{(0,1)}\left(\widetilde{A}^S_{00}\times\widetilde{B}^S_{00}\right)={}^{(0,1)}\left(\left\|\begin{array}{cc}a_{0000}&a_{0001}\\a_{1000}&a_{1001}\end{array}\right\|\times\left\|\begin{array}{cc}b_{000}\\b_{100}\end{array}\right\|\right)=\begin{pmatrix}1&2\\-2&0\end{pmatrix}\times\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}3\\-2\end{pmatrix}=\begin{pmatrix}c_{000}\\c_{100}\end{pmatrix},$$

$$\widetilde{C}^S_{01}={}^{(0,1)}\left(\widetilde{A}^S_{01}\times\widetilde{B}^S_{01}\right)={}^{(0,1)}\left(\left\|\begin{array}{cc}a_{0010}&a_{0001}\\a_{1010}&a_{1001}\end{array}\right\|\times\left\|\begin{array}{cc}b_{001}\\b_{101}\end{array}\right\|\right)=\begin{pmatrix}0&-1\\1&3\end{pmatrix}\times\begin{pmatrix}2\\3\end{pmatrix}=\begin{pmatrix}-3\\11\end{pmatrix}=\begin{pmatrix}c_{001}\\c_{101}\end{pmatrix},$$

$$\widetilde{C}^S_{10}={}^{(0,1)}\left(\widetilde{A}^S_{2}\times\widetilde{B}^S_{2}\right)={}^{(0,1)}\left(\left\|\begin{array}{cc}a_{0100}&a_{0101}\\a_{1100}&a_{1101}\end{array}\right\|\times\left\|\begin{array}{cc}b_{010}\\b_{110}\end{array}\right\|\right)=\begin{pmatrix}3&2\\0&1\end{pmatrix}\times\begin{pmatrix}-2\\4\end{pmatrix}=\begin{pmatrix}2\\4\end{pmatrix}=\begin{pmatrix}c_{010}\\c_{110}\end{pmatrix},$$

$$\widetilde{C}^S_{11}={}^{(0,1)}\left(\widetilde{A}^S_{11}\times\widetilde{B}^S_{11}\right)={}^{(0,1)}\left(\left\|\begin{array}{cc}a_{0000}&a_{0001}\\a_{1000}&a_{1001}\end{array}\right\|\times\left\|\begin{array}{cc}b_{000}\\b_{100}\end{array}\right\|\right)=\begin{pmatrix}4&1\\2&1\end{pmatrix}\times\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}c_{011}\\c_{111}\end{pmatrix}.$$

Итак, $${}^{2,1}(A\times B)=C=\left\|\begin{array}{cc|cc}c_{000}&c_{001}&c_{200}&c_{101}\\c_{010}&c_{011}&c_{110}&c_{111}\end{array}\right\|=\left\|\begin{array}{cc|cc}3&-3&-2&11\\2&1&4&1\end{array}\right\|$$ [11].

tensordot(A, B, μ) это (0, μ)-свернутое произведение многомерных матриц

Лемма 2.

Функция tensordot(A, B, μ), которая сворачивает многомерные матрицы по μ последним индексам первого аргумента и μ первым индексам второго, реализует (0, μ) свернутое произведение многомерных матриц A и B.

Доказательство леммы 2.

Результатом операции tensordot(*A*, *B*, μ) для двух многомерных объектов $A=\left\|a_{i_1\ldots i_p}\right\|$ и $B=\left\|b_{i_1\ldots i_q}\right\|$, где будет многомерный

[7] Соколов Н. П. Введение в теорию многомерных матриц. Киев : Наукова думка, 1972. 176 с.

[8] Там же.

объектов $A = \left\| a_{i_1 \ldots i_p} \right\|$ и $B = \left\| b_{i_1 \ldots i_q} \right\|$, где будет многомерный объект

$$\tilde{N} = \left\| \tilde{n}_{i_1 \ldots i_{p+q-2\mu}} \right\| = \left\| \sum_{i_{p-\mu+1}, i_{p-\mu+2}, \ldots, i_p} a_{i_1 i_2 \ldots i_{p-\mu} i_{p-\mu+1} \ldots i_p} \times b_{i_{p-\mu+1} \ldots i_p j_1 \ldots j_{q-\mu}} \right\|$$

(*). Разбив индексы многомерных матриц A и B на группы: $A = \left\| a_{i_1 \ldots i_p} \right\| = \left\| a_{i_1 \ldots i_{p-\mu} c_1 \ldots c_\mu} \right\| = \left\| a_{lc} \right\|$ и $B = \left\| b_{i_1 \ldots i_q} \right\| = \left\| b_{c_1 \ldots c_\mu m_1 \ldots m_{q-\mu}} \right\|$.

Тогда выражение (*) можно переписать в виде $\tilde{N} = \left\| \tilde{n}_{lm} \right\| = \left\| \sum_c a_{lc} \times b_{cm} \right\|$, что соответствует определению (0, μ)-свернутого произведения. Что и требовалось доказать.

Метода, реализующего (λ, μ)-свернутое произведение, в библиотеках тензорной алгебры нет.

Сравнение популярных python-библиотек, реализующих tensordot

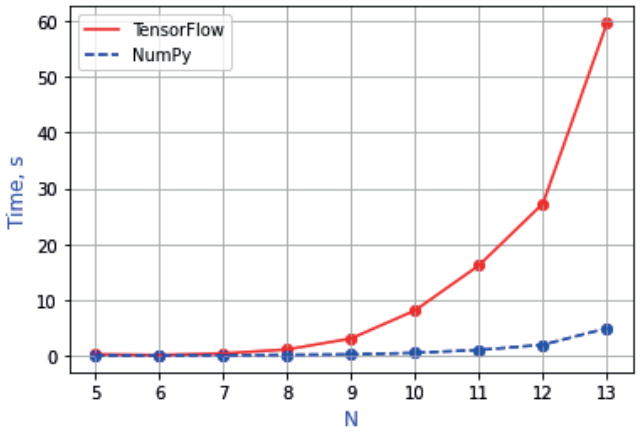
Рассмотрим наиболее популярные библиотеки для работы с тензорной алгеброй, реализующие операцию tensordot: NumPy [18], CuPy[9], PyTorch[10], TensorFlow[11].

Все они являются библиотеками с открытым исходным кодом, в основе которых лежит C++, Fortran CUDA, BLAS и cuBLAS, что обеспечивает их быстродействие. Сравнивать их будем попарно: NumPy vs TensorFlow с вычислениями на CPU, PyTorch vs CuPy с вычислениями на GPU[12]. Эксперимент производился в среде Colab [23, 24].

На оси X рисунка 2 указаны порядки 6-ти и 8-ми матриц A и B, на оси Y – необходимое время для выполнения операции. Точки, лежащие на штрихованной (основной) ломанной, показывают время в секундах $^{0,4}(A \times B)$ свернутого произведения матриц A и B, выполняемого средствами NumPy (TensorFlow) на CPU.

На оси X рисунка 3 указаны порядки 6-ти и 8-ми матриц A и B, на оси Y – необходимое время для выполнения операции. Точки, лежащие на штрихованной (основной) ломанной, показывают время в секундах $^{0,4}(A \times B)$ свернутого произведения матриц A и B, выполняемого средствами CuPy (PyTorch) на GPU.

Вычислительная сложность производимой операции – $O(N^{\lambda+\mu+\kappa+\nu}) = O(N^{0+4+2+4}) = O(N^{10})$, что объясняет экспоненциальный характер графиков. В производимом экспериментах NumPy и CuPy показали себя лучше в вычислениях на CPU и GPU, соответственно. Кроме этого, они имеют практически одинаковый синтаксис в силу того, что CuPy позиционирует себя как NumPy и SciPy для GPU вычислений [25]. Этот факт позволят адаптировать программный код под разные среды выполнений с минимальными изменениями.



Р и с. 2. NumPy vs TensorFlow (CPU)

F i g. 2. NumPy vs TensorFlow (CPU)

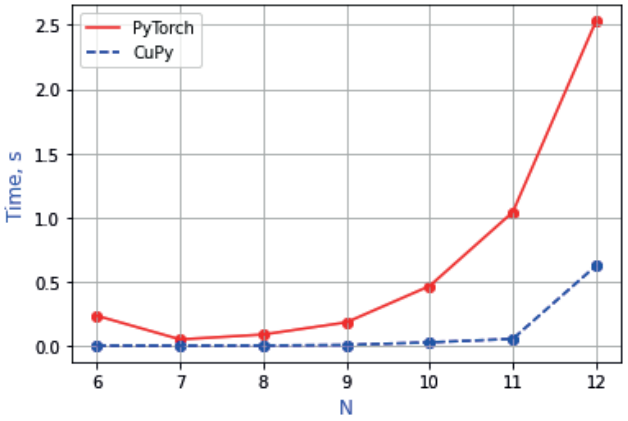


Рис. 3. PyTorch vs CuPy (GPU)

F i g. 3. PyTorch vs CuPy (GPU)

Программная реализация

NumPy имеет много полезных функций, упрощающих работу с многомерными массивами данных:

- инициализации p-мерной матрицы A порядка N:

$$A = A'.reshape(np.full\_like(np.arange(p,\ dtype = int),\ N)) = A'.reshape\left(\left[\underbrace{N\ \ \ldots\ \ N}_{p}\right]\right),$$

где $A' = \left[a'_1 \ \ \ldots \ \ a'_{N^p}\right]$ и $a'_\beta = a_{i_1 i_2 \ldots i_p}$, где $\beta = (\overline{i_1 i_2 \ldots i_p}_N)_{10}$.

[9] CuPy: NumPy & SciPy for GPU : сайт Preferred Networks [Электронный ресурс]. URL: https://cupy.dev (дата обращения: 28.09.2022).

[10] PyTorch : сайт The Linux Foundation [Электронный ресурс]. URL: https://pytorch.org (дата обращения: 28.09.2022).

[11] TensorFlow [Электронный ресурс] // Google. URL: www.tensorflow.org (дата обращения: 28.09.2022).

[12] Ziogas A. N. A Data-Centric Optimization Workflow for the Python Language : Doctoral Thesis. ETH Zurich, 2022. 173 p. doi: https://doi.org/10.3929/ethz-b-000586638

- взятие сечение матрицы:

$\widetilde{A^S_{S_i}} = A[\underbrace{\beta \ \ldots \ \beta}_{\kappa}, s_{i_1},\ldots,s_{i_\lambda}, \underbrace{\beta \ \ldots \ \beta}_{\mu}]$, где

$\beta = slice(None, None, None)]$.

- получение всевозможных наборов индексов:

Выражение $[(a,b)\ for\ a\ in\ \beta\ for\ b\ in\ \beta]$, где $\beta = [1 \ \ldots \ N]$, позволяет получать декартово произведение множеств β. Повторив его нужное количество раз, можно получить всевозможные наборы индексов.

- составление многомерной матрицы из ее сечений.

Операция $A[\underbrace{\beta \ \ldots \ \beta}_{\kappa}, s_{i_1},\ldots,s_{i_\lambda}, \underbrace{\beta \ \ldots \ \beta}_{\mu}] = \widetilde{A'^S_{S_i}}$, где $\beta = slice(None, None, None)]$ заменяет $\widetilde{A^S_{S_i}}$ сечение матрицы A на $\widetilde{A'^S_{S_i}}$. Замена всех сечений нулевой многомерной матрицы на данные дает нужный результат.

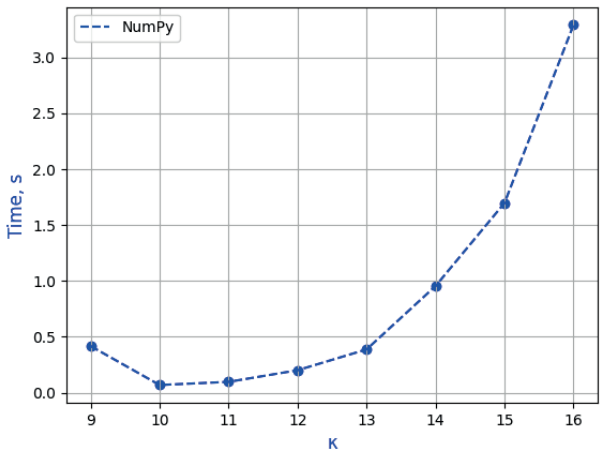
Имея такой вспомогательный функционал, можно реализовать параллельный вариант (λ, μ)-свернутого произведения матриц A и B по следующему алгоритму:

- комбинация методов получения всевозможных наборов индексов и взятия сечений многомерной матрицы реализует операцию $D^S(A) = \left\{\widetilde{A^S_{S_i}}\right\}_{i=1}^{N^\lambda}$. С её помощью представляем матрицы A и B в виде групп их скоттовых сечений: $\left\{\widetilde{A^S_{S_i}}\right\}_{i=1}^{N^\lambda}$ и $\left\{\widetilde{B^S_{S_i}}\right\}_{i=1}^{N^\lambda}$;
- параллельно с помощью библиотеки Joblib[13] выполняем (0, μ)-свернутое произведение для каждой пары $\widetilde{A^S_{S_i}}$ и $\widetilde{B^S_{S_i}}$, получив $\left\{\widetilde{C^S_{S_i}}\right\}_{i=1}^{N^\lambda}$;
- составляем многомерную матрицу C из ее сечений $\left\{\widetilde{C^S_{S_i}}\right\}_{i=1}^{N^\lambda}$.

Для оптимальной работы алгоритма рабочая станция должна иметь число потоков не меньшее, чем число всевозможных наборов скоттовых индексов (N^λ).

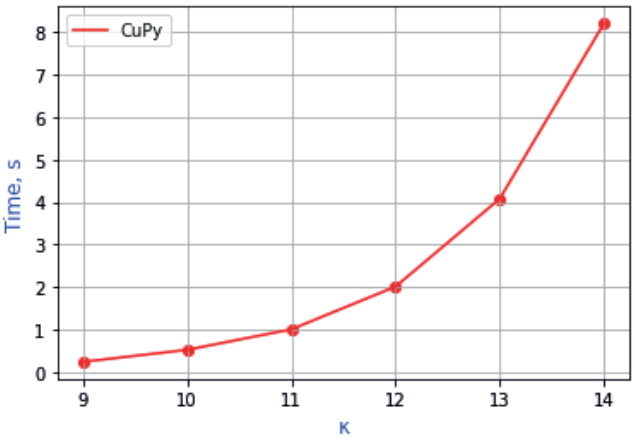
Полученные результаты

По описанию, изложенном в статье, был построен программный комплекс, реализующий (λ, μ)-свернутое произведение матриц A и B. Его тестирование на CPU проводилось на компьютере с 16 ГБ ОЗУ и процессором Ryzen™ 7 2700X. На оси X рисунка 4 указано число свободных индексов матрицы A, на оси Y – необходимое время для выполнения операции. Точки на штрихованной ломанной показывают время в секундах $^{3,5}(A\times B)$ свернутого произведения матриц A и B, где $N = 2$, $\nu = 9$.



Р и с. 4. (3, 5)-свернутое произведение NumPy
F i g. 4. (3, 5)-collapsed NumPy product

Тестирование на GPU проводилось в среде Colab. На оси X рисунка 5 указано число свободных индексов матрицы A, на оси Y – необходимое время для выполнения операции. Точки на основной ломанной показывают время в секундах $^{3,5}(A\times B)$ свернутого произведения матриц A и B, где $N = 2$, $\nu = 9$.



Р и с. 5. (3, 5)-свернутое произведение CuPy
F i g. 5. (3, 5)-collapsed CuPy product

Хотя прямое сравнение полученных результатов не корректно из-за разных сред выполнения, стоит отметить, что в среде Colab CPU обладает лишь 2-мя потоками, что сильно меньше 8-ми скоттовых индексов в матрицах, участвующих в операции.

Заключение

В статье было показано, что популярные python-библиотеки тензорной алгебры, написанные на C++, Fortran CUDA могут

[13] joblib – PyPI [Электронный ресурс] // Python Software Foundation. URL: https://pypi.org/project/joblib (дата обращения: 28.09.2022).

обеспечить достаточную скорость вычислений. Было доказано, что (λ, μ)-свернутое произведение можно представить через последовательность (0, μ)-свернутых произведений, которые можно выполнять параллельно, причем операция (0, μ)-свернутое произведение из алгебры многомерных матриц в точности совпадает с операцией tensordot(A, B, μ) из тензорных библиотек. В статье проведен сравнительный анализ скорости вычислений операции tensordot(A, B, μ) в наиболее популярных python библиотеках, на основании которого была выбрана связка из NumPy и CuPy для разных сред вычислений. В статье подробно описывается программный комплекс, реализующий (λ, μ)-свернутое произведение многомерных матриц и приводятся результаты его тестирования, которые в последствии могут использоваться для сравнения и оценки других параллельных алгоритмов реализации (λ, μ)-свернутого произведения.

Список использованных источников

[1] Мунерман В. И. Построение архитектур программно-аппаратных комплексов для повышения эффективности массовой обработки данных // Системы высокой доступности. 2014. Т. 10, № 4. С. 3-16. URL: https://elibrary.ru/item.asp?id=22831892 (дата обращения: 28.09.2022).

[2] Диев О. Е., Мунерман В. И. Параллельная обработка распределённых баз данных в СУБД Postgresql // Системы компьютерной математики и их приложения. 2017. № 18. С. 68-70. URL: https://elibrary.ru/item.asp?id=30469404 (дата обращения: 28.09.2022).

[3] Мунерман В. И., Мунерман Д. В. Соответствие операций в многомерно-матричной и реляционной моделях данных // Системы компьютерной математики и их приложения. 2019. № 20. С. 209-214. URL: https://elibrary.ru/item.asp?edn=dsutbo (дата обращения: 28.09.2022).

[4] Кириллов Е. В., Мельник К. В., Мунерман В. И. Параллельная реализация вывода ассоциативных правил на основе NUMA-архитектуры // Системы компьютерной математики и их приложения. 2019. № 20. С. 170-176. URL: https://elibrary.ru/item.asp?id=39103178 (дата обращения: 28.09.2022).

[5] Гончаров Е. И., Мунерман В. И., Самойлова Т. А. Выбор параметров многомерных матриц для обобщенного алгоритма шифрования Хилла // Системы компьютерной математики и их приложения. 2019. № 20. С. 111-116. URL: https://elibrary.ru/item.asp?id=39103166 (дата обращения: 28.09.2022).

[6] Корчиц К. С., Муха В. С. Векторные односвязные цепи Маркова // Доклады БГУИР. 2003. Т. 1, № 3. С. 102-105. URL: https://libeldoc.bsuir.by/handle/123456789/30997 (дата обращения: 28.09.2022).

[7] Мунерман В. И., Самойлова Т. А. Алгебраический подход к алгоритмизации задач маршрутизации // Системы высокой доступности. 2018. Т. 14, № 5. С. 50-56. doi: https://doi.org/10.18127/j20729472-201805-08

[8] Гончаров Е. И. Многомерно-матричное определение операции свертки // Современные информационные технологии и ИТ-образование. 2021. Т. 17, № 3. С. 541-549. doi: https://doi.org/1010.25559/SITITO.17.202103.541-549

[9] Goncharov E., Munerman V., Yakovlev G. Software and Hardware Complex for Calculating Convolutions by Methods Multidimensional Matrix Algebra // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia : IEEE Computer Society, 2021. P. 2176-2180. doi: https://doi.org/10.1109/ElConRus51938.2021.9396584

[10] Goncharov E., Iljin P., Munerman V. Multidimensional Matrix Algebra Versus Tensor Algebra or $\mu > 0$ // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). St. Petersburg and Moscow, Russia : IEEE Computer Society, 2020. P. 1949-1954. doi: https://doi.org/10.1109/EIConRus49466.2020.9039478

[11] Гончаров Е. И. Реализация (λ, μ)-свернутого произведения матриц средствами (0, μ)-свернутого произведения // Системы компьютерной математики и их приложения. 2022. № 23. С. 96-100. URL: https://elibrary.ru/item.asp?id=48621275 (дата обращения: 28.09.2022).

[12] Linge S., Langtangen H. P. Programming for Computations – Python // Texts in Computational Science and Engineering. Vol. 15. Cham : Springer, 2020. 332 p. doi: https://doi.org/10.1007/978-3-030-16877-3

[13] Sanner M. F. Python: A Programming Language for Software Integration and Development // Journal of Molecular Graphics and Modelling. 1999. Vol. 17, issue 1. P. 57-61. doi: https://doi.org/10.1016/S1093-3263(99)99999-0

[14] Dabhi S., Parmar M. Eigenvector Component Calculation Speedup Over NumPy for High-Performance Computing // Proceedings of 6th International Conference on Recent Trends in Computing. Lecture Notes in Networks and Systems ; ed. by R. P. Mahapatra, B. K. Panigrahi, B. K. Kaushik, S. Roy S. Vol. 177. Singapore : Springer, 2021. P. 241-249. doi: https://doi.org/10.1007/978-981-33-4501-0_23

[15] Towards Modern C++ Language Support for MPI / S. Ghosh [и др.] // 2021 Workshop on Exascale MPI (ExaMPI). St. Louis, MO, USA : IEEE Computer Society, 2021. P. 27-35. doi: https://doi.org/10.1109/ExaMPI54564.2021.00009

[16] Pellegrini S., Prodan R., Fahringer T. A Lightweight C++ Interface to MPI // 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing. Munich, Germany : IEEE Computer Society, 2012. P. 3-10. doi: https://doi.org/10.1109/PDP.2012.42

[17] Ballman A., Svoboda D. Avoiding Insecure C++ —How to Avoid Common C++ Security Vulnerabilities // 2016 IEEE Cybersecurity Development (SecDev). Boston, MA, USA : IEEE Computer Society, 2016. P. 65-65. doi: https://doi.org/10.1109/SecDev.2016.022

[18] Bisong E. NumPy // Building Machine Learning and Deep Learning Models on Google Cloud Platform. Berkeley, CA : Apress, 2019. P. 91-113. doi: https://doi.org/10.1007/978-1-4842-4470-8_10

[19] Sokolov N. P. Functions of multidimensional matrices and their applications for the solutions of linear systems of partial differential equations // Ukrainian Mathematical Journal. 1970. Vol. 22, issue 6. P. 657-674. doi: https://doi.org/10.1007/BF01086271

[20] Николаев К. С. Применение современных параллельных технологий к решению задачи умножения многомерных матриц методом рекурсивного спуска // Системы компьютерной математики и их приложения. 2020. № 21. С. 183-188. URL: https://elibrary.ru/item.asp?id=44237974 (дата обращения: 28.09.2022).

[21] Гончаров Е. И., Ильин П. Л. Сравнение реализаций блочного алгоритма умножения многомерных матриц // Системы компьютерной математики и их приложения. 2020. № 21. С. 102-109. URL: https://elibrary.ru/item.asp?id=44237961 (дата обращения: 28.09.2022).

[22] Захаров В. Н., Мунерман В. И. Параллельный алгоритм умножения многомерных матриц // Современные информационные технологии и ИТ-образование. 2015. Т. 11, № 2. С. 384-390. URL: https://elibrary.ru/item.asp?id=26167519 (дата обращения: 28.09.2022).

[23] Nelson M. J., Hoover A. K. Notes on Using Google Colaboratory in AI Education // Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). New York, NY, USA : Association for Computing Machinery, 2020. P. 533-534. doi: https://doi.org/10.1145/3341525.3393997

[24] Grout I. Realization of NumPy Tensordot using the Field Programmable Gate Array for Embedded Machine Learning Applications // 2020 8th International Electrical Engineering Congress (iEECON). Chiang Mai, Thailand : IEEE Computer Society, 2020. P. 1-4. doi: https://doi.org/10.1109/iEECON48109.2020.229523

[25] Sarkar V. Automatic Parallelization of Python Programs for Distributed Heterogeneous Computing / J. Shirako [и др.] // Euro-Par 2022: Parallel Processing. Euro-Par 2022. Lecture Notes in Computer Science ; ed. by J. Cano, P. Trinder. Vol. 13440. Cham : Springer, 2022. P. 350-366. doi: https://doi.org/10.1007/978-3-031-12597-3_22

Поступила 28.09.2022; одобрена после рецензирования 17.11.2022; принята к публикации 23.11.2022.

Об авторе:

Гончаров Евгений Игоревич, магистрант физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), **ORCID: https://orcid.org/0000-0002-3393-5772**, drbenvey1996@mail.ru

Автор прочитал и одобрил окончательный вариант рукописи.

References

[1] Munerman V.I. Construction of hardware-software complexes architecture to improve massively data processing. *Highly Available Systems.* 2014;10(4):3-16. Available at: https://elibrary.ru/item.asp?id=22831892 (accessed 28.09.2022). (In Russ., abstract in Eng.)

[2] Diev O.E., Munerman V.I. [Parallel processing of distributed databases in Postgresql DBMS]. *Computer Mathematics Systems and Their Applications.* 2017;(18):68-70. Available at: https://elibrary.ru/item.asp?id=30469404 (accessed 28.09.2022). (In Russ.)

[3] Munerman V.I., Munerman D.V. The compliance of operations of multi- dimensional matrix algebra with operations of the relational data model. *Computer Mathematics Systems and Their Applications.* 2019;(20):209-214. Available at: https://elibrary.ru/item.asp?edn=dsutbo (accessed 28.09.2022). (In Russ., abstract in Eng.)

[4] Kirillov E.V., Melnik K.V., Munerman V.I. Parallel association rules mining based on NUMA-architecture. *Computer Mathematics Systems and Their Applications.* 2019;(20):170-176. Available at: https://elibrary.ru/item.asp?id=39103178 (accessed 28.09.2022). (In Russ., abstract in Eng.)

[5] Goncharov E.I., Munerman V.I., Samoylova T.A. The method of selecting parameters of multidimensional matrix for hill encryption algorithm. *Computer Mathematics Systems and Their Applications.* 2019;(20):111-116. Available at: https://elibrary.ru/item.asp?id=39103166 (accessed 28.09.2022). (In Russ., abstract in Eng.)

[6] Korchits K.S., Mukha V.S. Vectorial 1-Connected Markov Chains. *Doklady BGUIR*. 2003;1(3):102-105. Available at: https://libeldoc.bsuir.by/handle/123456789/30997 (accessed 28.09.2022). (In Russ., abstract in Eng.)

[7] Munerman V.I., Samoylova T.A. Algebraic approach to algorithmization of routing problems. *Highly available systems*. 2018;14(5):50-56. (In Russ., abstract in Eng.) doi: https://doi.org/10.18127/j20729472-201805-08

[8] Goncharov E.I. Multi-Dimensional Definition of Convolution. *Modern Information Technologies and IT-Education.* 2021;17(3):541-549. (In Russ., abstract in Eng.) doi: https://doi.org/10.25559/SITITO.17.202103.541-549

[9] Goncharov E., Munerman V., Yakovlev G. Software and Hardware Complex for Calculating Convolutions by Methods Multidimensional Matrix Algebra. In: 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). St. Petersburg, Moscow, Russia: IEEE Computer Society; 2021. p. 2176-2180. doi: https://doi.org/10.1109/ElConRus51938.2021.9396584

[10] Goncharov E., Iljin P., Munerman V. Multidimensional Matrix Algebra Versus Tensor Algebra or $\mu > 0$. In: 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). St. Petersburg and Moscow, Russia: IEEE Computer Society; 2020. p. 1949-1954. doi: https://doi.org/10.1109/EIConRus49466.2020.9039478

[11] Goncharov E.I. Realization the (λ, μ)-convolution product of matrixes by means of the (0, μ)-convolution product. *Computer Mathematics Systems and Their Applications.* 2022;(23):96-100. Available at: https://elibrary.ru/item.asp?id=48621275 (accessed 28.09.2022). (In Russ., abstract in Eng.)
[12] Linge S., Langtangen H.P. Programming for Computations – Python. Texts in Computational Science and Engineering. Vol. 15. Cham: Springer; 2020. 332 p. doi: https://doi.org/10.1007/978-3-030-16877-3
[13] Sanner M.F. Python: A Programming Language for Software Integration and Development. *Journal of Molecular Graphics and Modelling*. 1999;17(1):57-61. doi: https://doi.org/10.1016/S1093-3263(99)99999-0
[14] Dabhi S., Parmar M. Eigenvector Component Calculation Speedup Over NumPy for High-Performance Computing. In: Mahapatra R.P., Panigrahi B.K., Kaushik B.K., Roy S. (eds.) Proceedings of 6th International Conference on Recent Trends in Computing. Lecture Notes in Networks and Systems. Vol. 177. Singapore: Springer; 2021. p. 241-249. doi: https://doi.org/10.1007/978-981-33-4501-0_23
[15] Ghosh S., Alsobrooks C., Rüfenacht M., Skjellum A., Bangalore P.V., Lumsdaine A. Towards Modern C++ Language Support for MPI. In: 2021 Workshop on Exascale MPI (ExaMPI). St. Louis, MO, USA: IEEE Computer Society; 2021. p. 27-35. doi: https://doi.org/10.1109/ExaMPI54564.2021.00009
[16] Pellegrini S., Prodan R., Fahringer T. A Lightweight C++ Interface to MPI. In: 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing. Munich, Germany: IEEE Computer Society; 2012. p. 3-10. doi: https://doi.org/10.1109/PDP.2012.42
[17] Ballman A., Svoboda D. Avoiding Insecure C++ —How to Avoid Common C++ Security Vulnerabilities. In: 2016 IEEE Cybersecurity Development (SecDev). Boston, MA, USA: IEEE Computer Society; 2016. p. 65-65. doi: https://doi.org/10.1109/SecDev.2016.022
[18] Bisong E. NumPy. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Berkeley, CA: Apress; 2019. p. 91-113. doi: https://doi.org/10.1007/978-1-4842-4470-8_10
[19] Sokolov N.P. Functions of multidimensional matrices and their applications for the solutions of linear systems of partial differential equations. *Ukrainian Mathematical Journal*. 1970;22(6):657-674. doi: https://doi.org/10.1007/BF01086271
[20] Nikolaev K.S. Application of modern parallel technologies to the solution of the problem of multiplaying multidimensional matrices by the method of recursive descent. *Computer Mathematics Systems and Their Applications.* 2020;(21):183-188. Available at: https://elibrary.ru/item.asp?id=44237974 (accessed 28.09.2022). (In Russ., abstract in Eng.)
[21] Goncharov E.I., Iljin P.L. Comparison of realisations of parallel multidimensional matrix multiplication algorithms. *Computer Mathematics Systems and Their Applications.* 2020;(21):102-109. Available at: https://elibrary.ru/item.asp?id=44237961 (accessed 28.09.2022). (In Russ., abstract in Eng.)
[22] Zakharov V.N., Munerman V. I. [Parallel Algorithm for Multidimensional Matrix Multiplication]. *Modern Information Technologies and IT-Education.* 2015;11(2):384-390. Available at: https://elibrary.ru/item.asp?id=26167519 (accessed 28.09.2022). (In Russ.)
[23] Nelson M.J., Hoover A.K. Notes on Using Google Colaboratory in AI Education. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). New York, NY, USA: Association for Computing Machinery; 2020. p. 533-534. doi: https://doi.org/10.1145/3341525.3393997
[24] Grout I. Realization of NumPy Tensordot using the Field Programmable Gate Array for Embedded Machine Learning Applications. In: 2020 8th International Electrical Engineering Congress (iEECON). Chiang Mai, Thailand: IEEE Computer Society; 2020. p. 1-4. doi: https://doi.org/10.1109/iEECON48109.2020.229523
[25] Shirako J., Hayashi A., Paul S.R., Tumanov A., Sarkar V. Automatic Parallelization of Python Programs for Distributed Heterogeneous Computing. In: Cano J., Trinder P. (eds.) Euro-Par 2022: Parallel Processing. Euro-Par 2022. Lecture Notes in Computer Science. Vol. 13440. Cham: Springer; 2022. p. 350-366. doi: https://doi.org/10.1007/978-3-031-12597-3_22

Submitted 28.09.2022; approved after reviewing 17.11.2022; accepted for publication 23.11.2022.

About the author:
Evgeniy I. Goncharov, Master degree student of the Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), **ORCID: https://orcid.org/0000-0002-3393-5772**, drbenvey1996@mail.ru

The author has read and approved the final manuscript.