

**Автономная некоммерческая организация высшего образования
«Университет Иннополис»**

**АННОТАЦИЯ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ)
ПО НАПРАВЛЕНИЮ ПОДГОТОВКИ
09.04.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА**

**НАПРАВЛЕННОСТЬ (ПРОФИЛЬ) ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ
«ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ИНЖЕНЕРИЯ ДАННЫХ»**

Тема

Эффективные методы сжатия тензорных данных

Выполнили

Нестеров Григорий Алексеевич

подпись

Ващенко Александр Александрович

подпись

Иннополис, Innopolis, 2025

Оглавление

1	Введение	3
2	Основные термины	5
3	Обзор литературы	6
4	Методология	8
5	Реализация	10
6	Анализ результатов	13
7	Заключение	16
	Список литературы	18

Глава 1

Введение

Высокоразмерные данные (тензоры) встречаются в различных прикладных областях. В качестве примеров могут выступать: нейронные сети, медико-биологические сигналы (ЭЭГ, МРТ), гиперспектральных изображений. Такие данные становятся все больше, требуя больших вычислительных мощностей для их использования. Классические методы тензорной декомпозиции, такие как Tucker [1], Tensor-Train (TT) [2], CANDECOMP/PARAFAC (CPD/CP) [1], [3]—[6], и их современные расширения позволяют аппроксимировать исходные тензора, уменьшая требуемый объём памяти и ускоряя вычисления на получаемых структурах [5], [6]. Однако эффективность разложения сильно зависит от правильно выбранных параметров метода, самого метода декомпозиции, что остаётся открытой исследовательской проблемой в зависимости от метода декомпозиции.

Актуальность работы. С ростом масштабов нейронных сетей и объёмов научных данных возрастает потребность в универсальных, устойчивых и масштабируемых методах тензорного сжатия, которые предоставляют гарантированное качество аппроксимированных данных и имеют готовые реализации с низкими накладными расходами по времени и памяти.

Цель работы — разработать и экспериментально обосновать эффективные методы выбора ранга и практические рекомендации по применению методов тензорной декомпозиции на примере задачи тензорного сжатия. А также в качестве прикладного практического примера, воспроизвести и улучшить,

на основе существующего исследования, метод сжатия нейронных сетей.

Задачи работы:

1. Провести сравнительный анализ современных реализаций методов тензорной декомпозиции на Python по времени выполнения, пиковому потреблению памяти и ошибки Фробениуса на 3D, 4D и 6D тензорах.
2. Разработать практические рекомендации по выбору методов декомпозиции и реализаций с учетом поддерживаемых форматов тензоров, языков программирования, зрелости API и эффективности по критериям: ошибка Фробениуса, время выполнения и требуемая память, занимаемая различными аппаратными компонентами.
3. Разработать алгоритм автоматического выбора ранга для Tucker и Tensor-Train, который достигает заданного пользователем коэффициента сжатия при минимизации ошибки Фробениуса.
4. Воспроизвести и улучшить метод сжатия нейронных сетей, добавлением алгоритма поиска ранга для Tucker, количественно оценить компромиссы между точностью и сжатием и определить пути для дальнейшей оптимизации.

Работа предлагает адаптивную процедуры выбора ранга, позволяющую автоматически подбирать ранг для Tucker и Tensor-Train на основе заданного процента сжатия. Разработанный программный пакет и методические рекомендации облегчают выбор инструментария при применении методов тензорной декомпозиции в прикладных задачах обработки высокоразмерных данных (тензоров).

Глава 2

Основные термины

Тензор. N -го порядка (или N -мерный) тензор — элемент прямого произведения N векторных пространств. Для $N=1$ это вектор, для $N=2$ — матрица, при $N \geq 3$ говорят о тензорах высшего/высокоразмерного порядка [1].

Структурные форматы тензоров: *плотные* [7] - все элементы хранятся явно (базовый случай); *разреженные* [8]—[10] - сохраняются только ненулевые элементы; эффективно при разреженных данных; *блочно - разреженные* [10] - группирует ненулевые элементы в блоки, ускоряя операции на структурированных данных; *(супер)симметричные* [7] - тензоры используют симметрию по модам, уменьшая избыточность и объём памяти.

Тензорное внешнее произведение расширяет понятие матричного произведения, комбинируя два тензора в новый, порядок которого равен сумме порядков исходных.

Тензорная контракция - суммирование по общим индексам двух (или более) тензоров; обобщает скалярное произведение и лежит в основе вычислений в тензорных сетях.

Тензорные сети факторизуют высокоразмерный тензор в граф низкоразмерных «ядер». Классический пример — *Tensor-Train*, обеспечивающий полиномиальный рост числа параметров вместо экспоненциального.

Методы тензорной декомпозиции. Представляют исходный. Это ключ к сжатию тензоров и слоев нейросетей: выбор ранга и формата напрямую определяет компромисс «точность—степень сжатия».

Глава 3

Обзор литературы

В этой главе проанализированы четыре ключевые семьи тензорных разложений: *CANDECOMP/PARAFAC* (CP), *Tucker* (HOSVD), *Tensor-Train* (TT) и *Robust Tensor PCA* (RTPCA) как представитель устойчивых методов. Рассмотрены их математические основы, недавние усовершенствования и практические сценарии использования.

Классические методы тензорной декомпозиции

- Разложение с названиями Canonical Decomposition (CANDECOMP), Canonical Polyadic Decomposition (CPD/CP) и Parallel Factor Analysis (PARAFAC) [11], [12] аппроксимирует тензор суммой рангов-1, обеспечивая структурную интерпретируемость при условной уникальности представления.
- Tucker/HOSVD [13] разлагает тензор на компактное ядро \mathcal{G} и матрицы факторов $\{A_n\}$ со свободными рангами R_n по модам. Гибкость даёт высокую точность и возможности денойзинга, но порождает неоднозначность факторов и сложность подбора рангов.
- Tensor-Train (TT) [2] хранит данные цепочкой ядер и TT-рангов, переводя экспоненциальную сложность в линейную по порядку d .

Современные расширения классических методов

- Robust Barron-Loss Tucker [14] заменяет норму Фробениуса обобщённой Barron-функцией потерь, уменьшая влияние выбросов при сохранении управляемости гладкостью.
- RTPCA [15] сочетает ядерную норму и ℓ_1 -штраф для разделения низкоранговой структуры и разреженных выбросов. Используется для избавления от шума в тензорах.

Релевантные исследованные работы

- *Stable Low-rank Decomposition* [16] объединяет CP и Tucker методы декомпозиции для аппроксимации сверточных слоев нейронных сетей.
- *Hybrid TT + Hierarchical Tucker* [17] показывает, что разные форматы оптимальны для свёрточных и полносвязных слоёв соответственно.
- Оптимизационные фреймворки [18] объединяют ADDM-регуляризацию, ТТ-разложение и дообучение, позволяя либо повысить точность, либо добиться экстремальных коэффициентов сжатия.
- *Cross Tensor Approximation* (CTA) [19] обходит полную SVD, используя выборку срезов/фибр и малое ядро; обеспечивает скорость и малый объём памяти для гиперспектральных и EEG-тензоров.
- *Time-aware tensor decomposition for sparse tensors* [20] вводят динамический штраф по временной моде, сочетая аналитическое и итеративное обновления матриц факторов для эволюционирующих данных.

Глава 4

Методология

В данной главе представлена методология, охватывающая разработку: (1) бенчмарка реализаций методов тензорной декомпозиции, (2) алгоритма автоматического выбора ранга для декомпозиций Tucker и Tensor Train, а также (3) алгоритма аппроксимации нейронных сетей.

Корпус тензоров для бенчмарка

Изображения — три RGB-тензора формата $(H, W, 3)$: $564 \times 564 \times 3$, $412 \times 620 \times 3$ и $689 \times 1195 \times 3$; *Видео* — три 4-х мерных тензора $(T, H, W, 3)$: $220 \times 256 \times 144 \times 3$, $100 \times 144 \times 192 \times 3$ и $237 \times 144 \times 256 \times 3$; *ЭЭГ* — два 6-ти мерных тензора, включающие факторы «субъект», «сеанс», «событие», «эпоха», «канал», «время»: $4 \times 12 \times 2 \times 15 \times 64 \times 1281$ и $3 \times 1 \times 1050 \times 2 \times 132 \times 201$.

Алгоритм выбора ранга для Tucker и Tensor-Train

Ранг (\mathbf{r}) ищется как минимум функции $\mathcal{L}(\mathbf{r}) = \alpha \varepsilon_F(\mathbf{r}) + \beta (\rho_{\text{target}} - \rho_{\text{actual}}(\mathbf{r}))^2$, где ε_F — нормированная ошибка Фробениуса, ρ — доля памяти (цель — 50% от исходного объёма). Ограничения на ТТ-ранги r_k и Tucker-ранги R_n задаются классическими верхними/нижними границами. Поиск ведётся пакетами SciPy (Nelder–Mead, Powell, SLSQP, differential_evolution) плюс кастомный локальный оптимизатор; выбирается наименьшая \mathcal{L} .

Сравнение по качественным критериям

Производится ручное сравнение представленных в работе [21] реализаций, по критериям: поддерживаемые форматы тензоров, аппаратная поддержка, поддерживаемые языки, доступность.

Дизайн бенчмарка методов тензорной декомпозиции

Для каждого тензора вызывается процедура выбора ранга (для Tucker или TT) под целевое сжатие 0.5; Запускается расчет аппроксимации и реконструкции на основе реализаций методов декомпозиции представленных в TensorLy и T3F; Регистрируются время, пиковая память, ошибка Фробениуса.

Сравнение по количественным критериям

На основе зарегистрированных логов бенчмарка проводится сравнение реализаций методов тензорной декомпозиции по следующим критериям: пиковое потребление VRAM и RAM, затраченное время и ошибка Фробениуса.

Аппроксимации слоев нейронных сетей

На основе работы [16] воспроизводится и улучшается метод аппроксимации нейронных сетей в PyTorch: автоматический выбор слоёв (conv и transposed-conv); применение CP, Tucker или гибрид CP+Tucker к фильтрам; интеграция алгоритма ранга для соблюдения заданного ρ_{target} ; дообучение сети.

Глава 5

Реализация

Вся кодовая база доступна по ссылке: <https://github.com/Innopolis-tensor-compression/tensor-compression-methods>. Проект организован с учетом возможности воспроизводимости: фиксация версий библиотек в `pyproject.toml`, установленные сиды (`np/torch/tf.random.seed(42)`), отключённые нефрагментированные эвристики `cuDNN` и `TF_DETERMINISTIC_OPS=1`.

Использованные среды для экспериментов:

- *Среда А* — WSL 2 (Ubuntu 22.04) на Intel i7-6700K, 32 GB DDR3, NVIDIA GTX 1080 Ti 11 GB; Python 3.11, CUDA-PyTorch 2.0.1, TensorFlow 2.17.
- *Среда В* — Ubuntu 25.04 на Intel i7-13700HK, 32 GB DDR5, NVIDIA RTX 4070 8 GB; Python 3.12, PyTorch 2.5.1 + CUDA.

Алгоритм подбора ранга для Tucker и Tensor-Train

Эксперименты реализованы на TensorLy+PyTorch (Tucker, TT) и SciPy оптимизаторах (Nelder–Mead, Powell, SLSQP, differential evolution) с единой функцией потерь $\mathcal{L}(\mathbf{r}) = \alpha \varepsilon_F(\mathbf{r}) + \beta (\rho_{\text{target}} - \rho_{\text{actual}}(\mathbf{r}))^2$, ($\alpha=1$, $\beta=10$). Для быстрых тестов доступен детерминированный алгоритм покоординатного поиска.

Сравнение реализаций методов декомпозиции

Бенчмарк реализаций методов тензорной декомпозиции

Загрузка тензора (изображения, видео, ЭЭГ); выбор ранга с помощью разработанного алгоритма под целевое сжатие 50%; запуск разложения с реализацией на TensorLy или T3F; логирование: время (`perf_counter`), пик RAM/VRAM (`memory_profiler`, `torch.cuda.max_memory_allocated`), ошибка Фробениуса, фактическое сжатие; экспорт в JSON для последующего анализа.

Сравнение по количественным критериям

Количественную оценку результатов бенчмарка мы проводили посредством совокупности взаимодополняющих методов. Сырые журналы измерений подвергались предварительной статистической обработке (агрегирование медиан и доверительных интервалов), после чего к ним последовательно применялись: (1) корреляционный анализ, визуализированный в виде тепловых карт, что позволило выявить устойчивые связи между гиперпараметрами и метриками производительности; (2) понижение размерности методом главных компонент (PCA), обеспечивавшее проекцию многомерных данных в двумерное и трёхмерное пространство без существенной потери информативности; (3) парные диаграммы рассеяния (*pairplot*), использовавшиеся для детального изучения локальных зависимостей и проверки гипотез о наличии кластеров. Совместное применение указанных техник обеспечило надёжное, репрезентативное и интерпретируемое сравнение реализаций по времени исполнения, пиковому потреблению памяти и ошибке Фробениуса.

Сжатие слоев нейронных сетей

На базе PyTorch создан метод, который: сканирует модель, выделяет свёрточные и транспонированные свёрточные слои; применяет CP, Tucker или гибриды CP+Tucker с автоматически подобранным рангом, исходя из заданного процента сжатия; подменяет исходный слой компактной каскадой; выполняет дообучение для восстановления точности.

На VGG-16 и ResNet-18/50 достигнуто 4–8-кратное сокращение параметров при снижении топ-1-точности не более чем на 1 %.

Косвенная работа и трудности

Ограничения GPU-памяти вынудили исключить CP-разложение для 4D и 6D тензоров; переход TensorLy на новую мажорную версию потребовал адаптации к изменённому API; ряд C/Matlab-библиотек отклонён по отсутствию Python-обёрток и невозможности конвертации форматов тензоров.

Глава 6

Анализ результатов

В текущей главе представлен анализ результатов по исследованным темам (1) точность алгоритма автоматического выбора ранга, (2) производительность популярных библиотек тензорных разложений и (3) эффективность предложенного метода сжатия нейронных сетей.

Алгоритм выбора ранга для Tucker и Tensor-Train

Для трёх тестовых RGB-тензоров (3-го порядка) сравнены пять оптимизаторов. *Дифференциальная эволюция* оказалась наилучшей по совокупности критериев: при целевом сжатии 50 % она обеспечила: среднюю относительную ошибку Фробениуса $< 1\%$ (0.01 % для наиболее «цветового» изображения); время подбора ранга 39–51 с против 94 с у координатного поиска и 28–58 с у Powell при худшей точности; надёжную сходимость без «залипаний» в начальном ранге, наблюдавшихся у Nelder–Mead и SLSQP.

Таким образом, сформулированная задача минимизации комбинированного функционала (ошибка Фробениуса + штраф за отклонение от заданного коэффициента сжатия) успешно решена; итоговый метод поиска ранга для форматов Tucker и ТТ опубликован в репозитории.

Бенчмарк реализаций методов тензорной декомпозиции

Комплексный бенчмарк (10 158 уникальных запусков; тензоры 3D–6D) охватывал TensorLy 0.9.0, T3F 1.2.0. Ключевые выводы:

- **TensorLy + Tensor-Train** показал наилучший баланс «точность – память – время»: ошибка Фробениуса 0.0002–0.48 %, ускорение до 5× по сравнению с CP/Tucker, память ≤ 3 ГБ для 3D и ≤ 5.5 ГБ для 6D.
- **CP/PARAFAC** превосходил по ошибке (до 0.3 %) лишь на мелких 3D-тензорах, но требовал $\sim 2\times$ больше памяти и $5\times$ больше времени.
- Основные *ошибки по памяти* (50 % неудачных прогонов) вызваны комбинациями `init=svd` у TensorLy CP, `svd=symeig_svd` у TensorLy TT или `init=svd` и `svd=symeig_svd` у TensorLy Tucker; переход к `init=random` и `svd=truncated_svd` устраняет проблему без потери качества.

Итоговые рекомендуемые параметры: *TT* (`truncated_svd`), *Tucker* (`init=random`, `svd=randomized_svd`) и *CP* (`init=random`, `cvg_criterion=rec_error`, `l2-reg.=0.5`).

Сжатие слоев нейронных сетей

Метод на основе PyTorch применён к шести моделям ImageNet (ResNet-18/34/50, VGG-11/16/19); все свёрточные и обратные свёрточные слои заменены Tucker-ядрами с автоматическим подбором ранга.

- **Сжатие параметров:** 4–8× (30 % от исходного веса).
- **Скорость инференса:** ускорение 14–22 %.

- **Точность:** после 1 эпохи дообучения потери ≤ 1 pp Top-1; для ResNet-18/34 точность даже выросла на $\approx 2\%$ благодаря лёгкому регуляризующему эффекту разложения.

Ключевые выводы

1. Дифференциальная эволюция предпочтительна для оптимизации рангов для Tucker и Tensor-Train.
2. TensorLy-TT — практический стандарт в рамках рассмотренных реализаций для разнотипных тензоров; CP оправдан лишь при строгих требованиях к интерпретируемости на малых данных.
3. Интеграция автоматического ранга в метод сжатия слоев нейронных сетей позволяет получать компактные модели без ручного подбора гиперпараметров и без существенного ухудшения метрик.

Перспективы. Для улучшения есть возможность изучить гибриды «глобальный поиск + локальное доулучшение», GPU-ускоренные версии оптимизаторов ранга и дополнительные функции потерь, учитывающие структурные свойства данных.

Все графики, логи и ноутбуки доступны по ссылке: <https://github.com/Innopolis-tensor-compression/tensor-compression-methods>.

Глава 7

Заключение

Работа обобщает результаты систематического исследования тензорных разложений для трёх классов тензоров (изображения, видео и ЭЭГ) и их применения к сжатию слоев нейронных сетей.

Основные достижения.

1. *Бенчмарк 10 000+ уникальных прогонов* на 3D–6D тензорах: измерены время, пиковая RAM/VRAM и ошибка Фробениуса. Выявлено, что TensorLy-ТТ обеспечивает оптимальный баланс точности и ресурсов; даны практические настройки гиперпараметров для CP, Tucker и ТТ.
2. Разработан *алгоритм выбора ранга* для Tucker и Tensor-Train. Алгоритм на базе дифференциальной эволюции гарантирует заданное сжатие при минимальной ошибке Фробениуса.
3. Воссоздан и улучшен *конвейер сжатия отдельных слоев нейронных сетей*. Для VGG и ResNet получено 4–8x уменьшение параметров и 14–22 % ускорение инференса при падении Top-1 не более 1 pp (часто — рост после дообучения).

Практическая ценность.

- Открытый репозиторий содержит итоговый код реализаций, логи и нутбуки с экспериментами (<https://github.com/Innopolis-tensor-compr>

ession/tensor-compression-methods), обеспечивая воспроизводимость и возможность дальнейшего расширения.

- Рекомендации по выбору библиотек, форматов и параметров пригодны для прикладных задач связанных с декомпозицией тензоров.

Ограничения и направления будущих работ.

- Улучшить функцию потерь и исследовать гибридные «глобальный + локальный» оптимизаторы ранга методов декомпозиции.
- Расширить конвейер на другие типы слоёв.
- Дополнить бенчмарк новыми наборами данных (аудио, гиперспектр) и библиотеками (EXATN, Scikit-TT).

Таким образом, работа вносит вклад в практику эффективного сжатия тензорных данных и моделей, предлагая как новые алгоритмы, так и документированную инфраструктуру для их оценки и применения.

Список литературы

- [1] T. G. Kolda и B. W. Bader, «Tensor Decompositions and Applications,» *SIAM Review*, т. 51, № 3, с. 455—500, 2009. DOI: 10.1137/07070111X. eprint: <https://doi.org/10.1137/07070111X>. url: <https://doi.org/10.1137/07070111X>.
- [2] I. V. Oseledets, «Tensor-Train Decomposition,» *SIAM Journal on Scientific Computing*, т. 33, № 5, с. 2295—2317, 2011. DOI: 10.1137/090752286. eprint: <https://doi.org/10.1137/090752286>. url: <https://doi.org/10.1137/090752286>.
- [3] G. Tomasi и R. Bro, «PARAFAC and missing values,» *Chemometrics and Intelligent Laboratory Systems*, т. 75, № 2, с. 163—180, 2005, ISSN: 0169-7439. DOI: <https://doi.org/10.1016/j.chemolab.2004.07.003>. url: <https://www.sciencedirect.com/science/article/pii/S0169743904001741>.
- [4] R. Bro, «Multiway analysis in the food industry. Models, algorithms and applications,» *Ph.D. dissertation, University of Amsterdam, Amsterdam*, авг. 2001.
- [5] G. Ballard и T. G. Kolda, *Tensor Decompositions for Data Science*. авг. 2024, Preliminary draft copy. Accessed on: October 4, 2024, url: <https://www.mathsci.ai/post/tensor-textbook/>.
- [6] Y. Liu, J. Liu, Z. Long и C. Zhu, *Tensor Computation for Data Analysis*. Springer Cham, янв. 2022, ISBN: 978-3-030-74385-7. DOI: 10.1007/978-3-030-74386-4.

- [7] J. Synge и A. Schild, *Tensor Calculus* ((Dover books on Mathematics)). Dover Publications, 1978, ISBN: 9780486636122. url: <https://books.google.ru/books?id=8vlGhlxqZjsC>.
- [8] I. Duff, A. Erisman и J. Reid, *Direct Methods for Sparse Matrices*. Oxford University Press, 2017, ISBN: 9780198508380. url: <https://books.google.ru/books?id=JhlLDgAAQBAJ>.
- [9] D. Ahn, J.-G. Jang и U. Kang, «Time-aware tensor decomposition for sparse tensors,» *Machine Learning*, т. 111, № 4, с. 1409—1430, 2022, ISSN: 1573-0565. DOI: 10.1007/s10994-021-06059-7. url: <https://doi.org/10.1007/s10994-021-06059-7>.
- [10] Z. he, J. Li и L. Liu, «Tensor Block-Sparsity Based Representation for Spectral-Spatial Hyperspectral Image Classification,» *Remote Sensing*, т. 8, с. 636, авг. 2016. DOI: 10.3390/rs8080636.
- [11] F. L. Hitchcock, «The Expression of a Tensor or a Polyadic as a Sum of Products,» *Journal of Mathematics and Physics*, т. 6, № 1-4, с. 164—189, 1927. DOI: <https://doi.org/10.1002/sapm192761164>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192761164>. url: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164>.
- [12] J. D. Carroll и J.-J. Chang, «Analysis of individual differences in multidimensional scaling via an n-way generalization of “Eckart-Young” decomposition,» *Psychometrika*, т. 35, № 3, с. 283—319, 1970. DOI: 10.1007/BF02310791. url: <https://doi.org/10.1007/BF02310791>.
- [13] L. R. Tucker, «Some Mathematical Notes on Three-Mode Factor Analysis,» *Psychometrika*, т. 31, № 3, с. 279—311, 1966. DOI: 10.1007/BF02289464.

- [14] М. Mozaffari и Р. Р. Markopoulos, «Robust Barron-Loss Tucker Tensor Decomposition,» в *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, с. 1651—1655. DOI: 10.1109/IEEECONF53345.2021.9723232.
- [15] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin и S. Yan, *Tensor Robust Principal Component Analysis with A New Tensor Nuclear Norm*, 2019. arXiv: 1804.03728 [stat.ML]. url: <https://arxiv.org/abs/1804.03728>.
- [16] А. Н. Phan, К. Sobolev, К. Sozykin и др., «Stable Low-rank Tensor Decomposition for Compression of Convolutional Neural Network,» *CoRR*, т. abs/2008.05441, 2020. arXiv: 2008.05441. url: <https://arxiv.org/abs/2008.05441>.
- [17] B. Wu, D. Wang, G. Zhao, L. Deng и G. Li, «Hybrid tensor decomposition in neural network compression,» *Neural Networks*, т. 132, с. 309—320, дек. 2020, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.09.006. url: <http://dx.doi.org/10.1016/j.neunet.2020.09.006>.
- [18] M. Yin, Y. Sui, S. Liao и B. Yuan, *Towards Efficient Tensor Decomposition-Based DNN Model Compression with Optimization Framework*, 2021. arXiv: 2107.12422 [cs.CV]. url: <https://arxiv.org/abs/2107.12422>.
- [19] S. Ahmadi-Asl, C. F. Caiafa, A. Cichocki и др., «Cross Tensor Approximation Methods for Compression and Dimensionality Reduction,» *IEEE Access*, т. 9, с. 150 809—150 838, янв. 2021. DOI: 10.1109/ACCESS.2021.3125069.
- [20] D. Ahn, J.-G. Jang и U. Kang, «Time-aware tensor decomposition for sparse tensors,» *Machine Learning*, т. 111, № 4, с. 1409—1430, 1 апр. 2022, ISSN: 1573-0565. DOI: 10.1007/s10994-021-06059-7. url: <https://doi.org/10.1007/s10994-021-06059-7>.

- [21] С. Psarras, L. Karlsson и P. Bientinesi, «The landscape of software for tensor computations,» *CoRR*, т. abs/2103.13756, 2021. arXiv: 2103.13756.
url: <https://arxiv.org/abs/2103.13756>.