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

Глава 1

Введение

Методы тензорных разложений находят широкое применение в анализе многомерных данных, машинном обучении и сжатии нейронных сетей [1], [2]. Они позволяют эффективно аппроксимировать многомерные структуры путём удаления малозначимой информации и сокращения объёма занимаемой памяти. Однако остаются нерешённые задачи, связанные с выбором оптимальных методов разложения и соответствующих параметров.

Существует множество библиотек, реализующих различные методы тензорных разложений: Tucker [3], Tensor-Train (TT) [4], PARAFAC [1]–[3], [5], [6] и RTPCA [7]. Эффективность каждого из них варьируется в зависимости от структуры и размерности данных, а также от конкретных задач. Одной из ключевых проблем остаётся выбор ранга разложения, напрямую влияющего на точность аппроксимации и степень сжатия.

Целью данной работы является проведение комплексного теоретического и эмпирического анализа методов тензорных разложений на различных типах данных (изображения, видеопоследовательности, ЭЭГ-сигналы), а также исследование их применимости к сжатию глубоких нейронных сетей.

В рамках работы были поставлены следующие задачи:

1. Провести бенчмарк современных Python-библиотек тензорных разложений по времени исполнения, потребляемой памяти и ошибке восстановления на представительных тензорах размерностей 3D, 4D и 6D;

2. Разработать практические рекомендации по выбору методов и инструментов разложения с учётом типов данных, особенностей API, языков программирования и метрик эффективности (ошибка Фробениуса, время, память);

3. Спроектировать алгоритм автоматического подбора рангов для форматов Tucker и Tensor-Train, обеспечивающий заданное пользователем сжатие при минимальной ошибке;

4. Интегрировать предложенный алгоритм ранжирования в расширенный пайплайн для сжатия свёрточных и полносвязных слоёв нейронных сетей с реализацией на Python;

5. Оценить разработанный пайплайн на стандартных архитектурах CNN, проанализировать компромиссы между точностью и сжатием, определить направления для дальнейшей оптимизации.

Вклад авторов распределён следующим образом. Оба автора совместно провели начальный обзор литературы по тензорной алгебре и методам разложения (глава 3).

Григорий Нестеров: систематизировал теоретическую часть обзора, реализовал бенчмарк, разработал алгоритм выбора рангов для Tucker и TT, провёл эмпирическое исследование на разнообразных типах данных (главы 5, 6).

Александр Ващенко: изучил применение тензорных разложений для сжатия нейросетей, расширил существующий метод сжатия слоя с учётом автоматического ранжирования, реализовал конечное решение и провёл его экспериментальную оценку (главы 5, 6).

Оба автора обсуждали полученные результаты и согласовали финальную версию работы.

Структура работы:

- Глава 2: основные понятия тензорной алгебры и форматы представления тензоров.
- Глава 3: обзор классических и современных методов разложения, включая их теоретические основы и практические применения.
- Глава 4: описание типов данных, используемых в бенчмарке, методологии эксперимента и подхода к выбору оптимального ранга.
- Глава 5: технические детали реализации алгоритмов, описания пайплайна и используемых библиотек.
- Глава 6: оценка эффективности решений, обсуждение полученных результатов и выявленных ограничений.
- Глава 7: выводы о наилучших методах и условиях их применения, ограничения и перспективы дальнейших исследований.

Глава 2

Основные термины

В данной главе вводятся ключевые понятия, используемые в работе: тензоры, форматы представления тензоров, тензорное произведение, тензорные сети и тензорные разложения.

Тензор

Тензор — это многомерный массив. Формально, тензор порядка N является элементом тензорного произведения N векторных пространств. Вектор и матрица — это тензоры первого и второго порядка соответственно, а тензоры порядка три и выше называются тензорами высших порядков [3].

Для представления тензоров применяются различные обозначения: компонентная форма, нотация Риччи (с верхними и нижними индексами), а также графическая нотация Пенроуза, в которой тензор изображается как узел, а его размерности — как рёбра.

Пример тензора третьего порядка — цветное изображение, хранящееся как массив NumPy формы $(H, W, 3)$, где два индекса соответствуют пространственным координатам, а третий — цветовому каналу.

Форматы тензоров

Помимо плотного (dense) представления, тензоры могут использовать специализированные форматы: разреженные (sparse), блочно-разреженные (block-sparse), симметричные и суперсимметричные [8]–[10].

Разреженные тензоры хранят только ненулевые элементы, что эффективно при высокой разреженности данных. Блочно-разреженные тензоры группируют ненулевые значения в блоки, обеспечивая структурированное сжатие. Симметричные тензоры обладают симметрией по ряду мод, что снижает избыточность. Суперсимметри тензоры расширяют это понятие на более высокие порядки и применяются в задачах с усиленными симметриями.

Тензорное произведение

Тензорное произведение — это операция, обобщающая понятие произведения матриц на случай тензоров произвольного порядка. Для тензоров $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_m}$ и $\mathcal{B} \in \mathbb{R}^{J_1 \times \cdots \times J_n}$, их тензорное произведение $\mathcal{A} \otimes \mathcal{B}$ задаётся как [9]:

$$(\mathcal{A} \otimes \mathcal{B})_{(i_1,\ldots,i_m,j_1,\ldots,j_n)} = \mathcal{A}_{i_1,\ldots,i_m} \cdot \mathcal{B}_{j_1,\ldots,j_n} \tag{2.1}$$

Результирующий тензор имеет порядок $m + n$ и размерность, равную произведению размерностей исходных тензоров. Эта операция лежит в основе многих методов тензорных разложений.

Тензорные сети

Тензорные сети представляют тензор как граф, узлы которого — тензоры низшего порядка, соединённые рёбрами (индексами). Это позволяет существенно снизить требования к памяти и вычислениям, особенно в задачах высокой размерности [11].

Характерным примером является представление Matrix Product State (MPS), также известное как формат Tensor Train (TT). Тензор $\mathcal{T} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$ раскладывается в TT-форму:

$$\mathcal{T}_{i_1 i_2 \ldots i_N} = \sum_{j_1, \ldots, j_{N-1}} A^{(1)}_{i_1 j_1} A^{(2)}_{j_1 i_2 j_2} \ldots A^{(N)}_{j_{N-1} i_N} \tag{2.2}$$

Здесь $A^{(n)}$ — TT-ядра с размерами $R_{n-1} \times I_n \times R_n$, где R_n — TT-ранги, отражающие степень сжатия. Формат широко используется в квантовой физике и машинном обучении [12], [13].

Тензорные разложения

Тензорное разложение — это процесс представления тензора в виде комбинации более простых компонентов, что облегчает анализ, хранение и обработку данных. Наиболее распространённые методы (например, TT, Tucker, PARAFAC) используют тензорное произведение для сборки исходного тензора из ядер и факторных матриц [3].

Такие разложения применимы ко всем упомянутым форматам: плотным, разреженным, симметричным и др. Это особенно важно при работе с тензорами высокого порядка, где прямые вычисления становятся неэффективными.

Глава 3

Обзор литературы

В данном разделе рассматриваются четыре основных семейства тензорных разложений: Tucker, Tensor-Train (TT), CANDECOMP/PARAFAC и Robust Tensor PCA (RTPCA). Первые три являются классическими методами, лежащими в основе большинства современных приложений, в то время как RTPCA представляет собой недавно разработанные робастные расширения. Основное внимание уделяется математическим основам, ключевым улучшениям и типичным областям применения.

3.1 Методы тензорного разложения

CANDECOMP / PARAFAC

Данный метод известен под разными названиями: Canonical Decomposition (CANDECOMP), Canonical Polyadic Decomposition (CPD), CP и Parallel Factor Analysis (PARAFAC), все они обозначают один и тот же подход. Метод был предложен Ф. Л. Хитчкоком в 1927 году [14] и позднее обобщён Кэрроллом и Чангом в 1970 году в контексте многомерного масштабирования [15].

CANDECOMP/PARAFAC аппроксимирует тензор $\mathcal{X} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$ суммой

ранга-один тензоров:

$$\mathcal{X} \approx \sum_{r=1}^{R} a_r^{(1)} \otimes a_r^{(2)} \otimes \cdots \otimes a_r^{(N)},$$

где R — ранг разложения, $a_r^{(n)} \in \mathbb{C}^{I_n}$ — факторные векторы, а $\otimes$ — тензорное произведение [3], [5], [6]. Метод характеризуется возможной уникальностью решений при выполнении определённых условий, что обеспечивает интерпретируемость факторов, однако подбор ранга и вычислительная эффективность остаются сложными задачами.

Tucker

Tucker-разложение (Higher-Order SVD, HOSVD) — обобщение матричных методов факторизации на многомерные массивы, предложенное Л. Р. Такером в 1966 году [16]. Для тензора третьего порядка $\mathcal{Y} \in \mathbb{C}^{I_1 \times I_2 \times I_3}$ разложение записывается как:

$$\mathcal{Y} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C,$$

где $\mathcal{G} \in \mathbb{C}^{R_1 \times R_2 \times R_3}$ — ядро тензора, а A, B, C — матрицы факторов, соответствующие режимам [16]. Такой подход позволяет адаптировать ранги по каждому режиму и выявлять сложные взаимодействия между компонентами, что полезно для анализа, сжатия и шумоподавления. Недостатком является неуниверсальность решения, затрудняющая интерпретацию факторов.

Робастное Tucker-разложение с функцией потерь Баррона [17]

В работе [17] предложено усовершенствование Tucker-разложения с использование робастной функции потерь Баррона [18] (с параметром $\alpha = 0$) для повышения устойчивости к шумам и выбросам. Это обеспечивает более точную аппроксимацию тензоров на реальных данных, содержащих аномалии.

Tensor-Train (TT)

TT-разложение [4] представляет тензор порядка d в виде цепочки связанных тензоров меньшего порядка (TT-ядра):

$$\mathcal{X}_{i_1,\ldots,i_d} \approx \sum_{r_1=1}^{R_1} \cdots \sum_{r_{d-1}=1}^{R_{d-1}} G^{(1)}_{i_1,r_1} G^{(2)}_{r_1,i_2,r_2} \cdots G^{(d)}_{r_{d-1},i_d},$$

где $G^{(k)}$ — TT-ядра, а R_k — TT-ранги, определяющие компромисс между точностью и размером представления. TT позволяет существенно уменьшить объём параметров, сохраняя многомерную структуру данных, и широко применяется для сжатия нейросетей и научных вычислений.

Robust Tensor Principal Component Analysis (RTPCA)

RTPCA сочетает тензорные разложения с робастным PCA, выделяя низкоранговую структуру данных и одновременно устраняя шумы и выбросы [7]. Формулировка задачи:

$$\min_{\mathcal{L},\mathcal{S}} \sum_n \|\mathcal{L}_{(n)}\|_* + \lambda \|\mathcal{S}\|_1, \quad \text{при условии } \mathcal{M} = \mathcal{L} + \mathcal{S},$$

где $\mathcal{M}$ — наблюдаемый тензор, $\mathcal{L}$ — низкоранговый компонент, $\mathcal{S}$ — разреженный шум. Метод эффективен для обработки реальных шумных данных, включая биомедицинские сигналы и видео [7].

3.2 Цели и приложения тензорных разложений

Цели

Основными задачами тензорных разложений являются:

- Сжатие данных — уменьшение объёма без значительных потерь информации, что критично для изображений и видео [2];
- Эффективное представление — понижение размерности для ускорения вычислений, например, сжатие слоёв нейросетей [13];
- Шумоподавление — выделение релевантной информации за счёт удаления шума [2].

Примеры применения

Стабильное низкоранговое разложение для сжатия сверточных сетей [13] демонстрирует эффективность сочетания CPD и Tucker для компрессии фильтров CNN (VGG-16, ResNet-18, ResNet-50) с улучшением производительности и снижением потерь точности.

Гибридный подход к сжатию нейросетей [19] сочетает TT для сверточных и Hierarchical Tucker для полносвязных слоёв, достигая улучшенных результатов по сравнению с отдельными методами.

Методы Cross Tensor Approximation [20] предлагают альтернативу традиционным разложениям с повышенной вычислительной эффективностью и масштабируемостью при обработке высокоразмерных данных.

—

Данный обзор подчёркивает важность глубокого понимания теоретических основ и актуальных направлений в тензорном анализе для эффективного использования методов в современных научных и инженерных задачах.

Глава 4

Методология

В данной главе описывается структура бенчмарка и представлены количественные критерии оценки методов тензорного разложения, включающие:

- использование памяти различными компонентами вычислений,
- ошибку Фробениуса,
- время вычислений.

Рассматриваются типы тензорных данных, особенности их представления, а также процедуры выбора параметров разложения. Особое внимание уделяется методам оптимизации ранга для разложений Тукера и Тензорного Поезда (Tensor Train, TT). Также описан алгоритм сжатия нейронных сетей и внедрённые улучшения.

4.1 Типы тензорных данных для бенчмарка

В этой секции представлены используемые типы тензорных данных и их конкретные примеры, применяемые для оценки и сравнения методов и библиотек тензорного разложения. Примеры демонстрируют возможности и сферу применения методов разложения.

Изображения

Изображения представлены трёхмерными тензорами с размерами, соответствующи
высоте, ширине и цветовым каналам (RGB). Данные загружаются из распространённых
форматов (jpg, png) с помощью библиотеки OpenCV-Python.

Для бенчмарка выбраны три варианта изображений с разными размерами
и аспектами:

- среднее изображение с размером (564, 564, 3),
- изображение с другим соотношением сторон (412, 620, 3),
- крупное изображение с высоким разрешением (689, 1195, 3).

Такие данные позволяют оценить влияние размера, соотношения сторон
и цветовой палитры на качество и производительность методов тензорного
разложения.

Видео

Видео рассматриваются как четвёртого порядка тензоры $\mathcal{X} \in \mathbb{R}^{T \times H \times W \times 3}$,
где T — число кадров, H и W — высота и ширина кадра соответственно, 3 —
количество цветовых каналов RGB.

Для бенчмарка выбраны три видеоролика с различной длительностью
и пространственным разрешением, загруженные с YouTube и обработанные
через OpenCV-Python. Выбор сделан с учётом ограничения вычислительных
ресурсов, чтобы обеспечить воспроизводимость экспериментов.

ЭЭГ (электроэнцефалография)

Для тестирования методов на тензорах высокого порядка использованы два набора данных ЭЭГ:

- **EEG Motor Movement/Imagery Dataset** [21] — шестимерный тензор с размерностями, отражающими субъектов, запуски экспериментов, типы событий, эпизоды, каналы и временные отсчёты. Использованы данные четырёх субъектов и двенадцати запусков для сохранения вариативности и управляемости вычислений.
- **LIMO dataset** — данные, доступные в библиотеке MNE-Python, содержащие тензор с пятью измерениями, включающими субъектов, запуски, испытания, события и временные отсчёты. Размеры были усечены до общих для всех субъектов значений для обеспечения однородности.

4.2 Оптимальный выбор ранга для разложений Тукера и Tensor Train

Выбор ранга является ключевым в тензорных разложениях, так как определяет компромисс между степенью сжатия и ошибкой аппроксимации. Задача формулируется как ограниченная задача оптимизации, решаемая с помощью локальных и глобальных алгоритмов минимизации из пакета SciPy с использованием разложений из TensorLy.

Ограничения на ранги

Tensor Train. Для каждого внутреннего ранга r_k действуют классические ограничения:

$$1 \leqslant r_k \leqslant \min\left(\prod_{i=1}^{k} I_i,\ \prod_{j=k+1}^{N} I_j\right), \quad k = 1, \ldots, N-1,$$

где $\mathcal{X} \in \mathbb{C}^{I_1 \times \cdots \times I_N}$ — исходный тензор, I_k — размерность k-го моды, r_k — k-й TT-ранг, N — порядок тензора. Внешние ранги фиксированы: $r_0 = r_N = 1$.

Tucker. Для разложения Тукера ранги выбираются для каждого измерения:

$$1 \leqslant r_n \leqslant I_n, \quad n = 1, \ldots, N,$$

где r_n — ранг Тукера вдоль n-й моды.

Функция потерь

Для каждой комбинации рангов строится приближение исходного тензора, после чего оцениваются два параметра:

1. Относительная ошибка Фробениуса — отношение нормы разности исходного и приближённого тензоров к норме исходного.
2. Коэффициент сжатия — отношение объёма памяти, занимаемого факторами разложения, к объёму памяти исходного тензора.

Оптимизация сводится к поиску рангов, минимизирующих комбинацию этих показателей с учётом баланса между точностью и сжатием.

4.3 Сжатие нейронных сетей

Для демонстрации практического применения методов тензорного разложения реализован алгоритм сжатия сверточной нейронной сети. Внедрены улучшения, направленные на повышение эффективности сжатия без существенной потери качества.

—

Таким образом, методология включает разработку комплексного бенчмарка, описывающего разнообразные типы данных, строгие критерии оценки и алгоритмически подходы к оптимальному выбору ранга в задачах тензорного сжатия и аппроксимации.

Глава 5

Реализация

В данной главе описывается реализация ключевых компонентов исследования, включая подготовку данных для бенчмарка, выбор параметров разложения и алгоритмы оптимизации ранга для тензорных разложений. Особое внимание уделено методам оценки качества и эффективности сжатия.

5.1 Типы тензорных данных для бенчмарка

В качестве тестовых данных использовались несколько типов тензоров, отражающих разнообразие реальных задач и проверяющих устойчивость методов разложения.

Изображения

Изображения представлены трехмерными тензорами с размерностями по высоте, ширине и цветовым каналам (RGB). Для экспериментов использованы изображения с различным разрешением и соотношением сторон, что позволяет проверить адаптивность методов к разным визуальным характеристикам.

Видео

Видео рассматриваются как тензоры четвёртого порядка с измерениями: количество кадров, высота, ширина и цветовые каналы. Для бенчмарка были

выбраны короткие видео с разными пространственно-временными параметрами, чтобы балансировать вычислительную нагрузку и разнообразие данных.

Электроэнцефалограммы (ЭЭГ)

Для тестирования методов на высокоразмерных тензорах применялись ЭЭГ-данные из публичных наборов [21]. Эти данные имеют шесть размерностей, отражающих субъектов, прогоны, события, эпохи, каналы и временные сэмплы, что создаёт значительную нагрузку на память и вычисления.

5.2 Выбор оптимального ранга для разложений Tucker и Tensor-Train

Определение ранга разложения критично для баланса между степенью сжатия и точностью аппроксимации. Задача формулируется как ограниченная оптимизация параметров ранга с использованием методов из TensorLy и алгоритмов минимизации из SciPy.

Ограничения на ранги

Для Tensor-Train ранги r_k ограничены классическими условиями:

$$1 \leqslant r_k \leqslant \min\left(\prod_{i=1}^{k} I_i, \prod_{j=k+1}^{N} I_j\right), \quad k = 1, \ldots, N-1,$$

где I_i — размерность i-го моды исходного тензора, что гарантирует корректность вычислений.

Для Tucker разложения ранги могут свободно варьироваться по каждой

моде в пределах:

$$1 \leqslant r_n \leqslant I_n, \quad n = 1, \ldots, N.$$

Функция потерь

Для каждого набора рангов вычисляется приближение исходного тензора, после чего оцениваются две метрики:

1. Относительная ошибка Фробениуса между исходным и восстановленным тензорами.
2. Коэффициент сжатия, измеряющий отношение памяти, занятой факторами разложения, к объему исходного тензора.

Оптимизация направлена на минимизацию функции потерь, учитывающей компромисс между точностью и степенью сжатия.

5.3 Дополнительно: алгоритм сжатия нейросетей

Также реализован алгоритм сжатия нейросетей, основанный на тензорных разложениях, который улучшен с учётом специфики выбранных методов и параметров ранга. Детали алгоритма и его влияние на производительность будут представлены далее.

Глава 6

Анализ результатов

В данном разделе представлен сравнительный анализ эффективности различных методов оптимизации рангов тензорных разложений, а также оценка производительности популярных библиотек для работы с тензорами.

Исследование показало, что локальные градиентные методы, такие как Nelder–Mead и SLSQP, продемонстрировали низкую адаптивность при поиске оптимальных рангов. В частности, они часто сходились к начальному низкорейтинговом решению [1,1,1,1], что указывает на их склонность к застреванию в локальных минимумах без возможности выхода. Это обусловлено многомодальностью и дискретной природой пространства рангов, а также отсутствием механизма глобального поиска. Таким образом, применение указанных алгоритмов требует существенной модификации или гибридизации с эвристическими методами для повышения качества оптимизации.

Скорость работы методов была оценена отдельно: SLSQP оказался наиболее быстрым, однако качество найденных решений было наихудшим. Это подчёркивает известный компромисс между быстродействием и точностью в задачах оптимизации рангов.

Из рассмотренных глобальных методов Grid Search и Random Search проявили себя менее эффективно с точки зрения времени выполнения, однако обеспечивали более устойчивый поиск, что соответствует их полной или частичной переборной природе. Предложенный автором метод Coordinate Descent, дополненный локальной оптимизацией, продемонстрировал наилучшее

сочетание качества решения и приемлемого времени работы.

Сравнение библиотек TensorLy, TensorFlow, и tntorch выявило, что TensorLy обеспечивает наиболее быструю и стабильную работу на тестовых тензорах с малыми и средними размерностями. TensorFlow показал высокую производительность на крупных тензорах благодаря GPU-ускорению, но страдал от повышенного потребления памяти. Библиотека tntorch оказалась самой медленной, что обусловлено спецификой реализации и отсутствием эффективной поддержки некоторых видов разложений.

Анализ логов экспериментов подтвердил критическую важность тщательного выбора стратегии оптимизации рангов для достижения компромисса между точностью восстановления тензора и вычислительными затратами. Результаты указывают на необходимость разработки более адаптивных и гибридных алгоритмов, способных учитывать специфику задачи и структуру данных.

Таким образом, проведённое исследование выявило ключевые ограничения существующих методов и библиотек, а также продемонстрировало перспективность предложенного подхода на основе локального и глобального поиска с комбинированием эвристик. Это открывает возможности для дальнейших разработок в области эффективного сжатия и анализа многомерных данных.

Глава 7

Заключение

В данной работе исследовалась эффективность и применимость методов тензорного разложения для обработки многомерных данных различных типов, включая изображения (3D), видео (4D) и ЭЭГ-сигналы (6D). Кроме того, была проведена оценка воспроизводимости и возможностей улучшения существующего алгоритма сжатия нейронных сетей на основе тензорных разложений.

Для достижения поставленных целей были решены следующие задачи:

- Выполнен сравнительный анализ существующих инструментов для тензорных разложений по качественным критериям. На его основе был сформирован выбор Python-библиотек: TensorLy, T3F, TENPy, scikit-tensor и Tensor Fox. Рассматривались поддерживаемые методы разложения, форматы тензоров, аппаратная совместимость, язык программирования и удобство использования. Для количественной оценки в бенчмарке были выбраны T3F и TensorLy.

- Проведён количественный бенчмарк по времени выполнения, пиковому потреблению памяти и ошибке Фробениуса на репрезентативных 3D, 4D и 6D тензорах, соответствующих изображениям, видео и ЭЭГ-данным. На основании результатов даны практические рекомендации по выбору методов разложения, параметров и библиотек с учётом типа данных, удобства API и производительности.

- Разработан алгоритм автоматического выбора ранга для тензорных разложений

Тукера и Tensor-Train, основанный на методе дифференциальной эволюции из SciPy. Алгоритм оптимизирует ранги с учётом заданного коэффициента сжатия и минимизации ошибки Фробениуса. Полученная реализация интегрирована в процессы автоматической оптимизации и использована в бенчмарке и алгоритме сжатия нейросетей.

- В качестве практического кейса была воспроизведена и улучшена существующая методика сжатия нейросетей с использованием тензорных разложений. Разработана Python-реализация, работающая с Torch-представлениями сетей и поддерживающая конкретные типы слоёв. Метод тестировался на различных архитектурах, показывая смешанные результаты: в ряде случаев точность приближённой модели близка к исходной, в других наблюдается падение, однако потребление памяти и время вывода стабильно уменьшаются.

Работа предоставляет готовую к использованию реализацию алгоритма выбора ранга для разложений Тукера и Tensor-Train на базе TensorLy, воспроизводит и улучшает метод сжатия нейросетей и содержит сравнительный анализ существующих Python-библиотек для тензорных разложений.

Подробности и результаты приведены в главах **??**, **??** и 6. Все методы и эксперименты реализованы на Python и доступны для воспроизводимости и дальнейших исследований.

Предложенные решения имеют определённые ограничения и перспективы развития. Алгоритм выбора ранга решает заявленную задачу, но может быть улучшен за счёт систематической оценки оптимизационных методов и настройки функции потерь. Метод сжатия нейросетей поддерживает ограниченный набор типов слоёв, расширение функционала остаётся задачей будущих исследований.

Кроме того, бенчмаркинг библиотек ограничен по охвату и не включает все рассмотренные на этапе качественного анализа инструменты, что может быть учтено при дальнейшем расширении работы.

Список литературы

[1] G. Ballard and T. G. Kolda, *Tensor Decompositions for Data Science*. Aug. 2024, Preliminary draft copy. Accessed on: October 4, 2024, [Online]. Available: https://www.mathsci.ai/post/tensor-textbook/.

[2] Y. Liu, J. Liu, Z. Long, and C. Zhu, *Tensor Computation for Data Analysis*. Springer Cham, Jan. 2022, ISBN: 978-3-030-74385-7. DOI: 10.1007/978-3-030-74386-4.

[3] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. DOI: 10.1137/07070111X. eprint: https://doi.org/10.1137/07070111X. [Online]. Available: https://doi.org/10.1137/07070111X.

[4] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011. DOI: 10.1137/090752286. eprint: https://doi.org/10.1137/090752286. [Online]. Available: https://doi.org/10.1137/090752286.

[5] G. Tomasi and R. Bro, "Parafac and missing values," *Chemometrics and Intelligent Laboratory Systems*, vol. 75, no. 2, pp. 163–180, 2005, ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2004.07.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743904001741.

[6] R. Bro, "Multiway analysis in the food industry. models, algorithms and applications," *Ph.D. dissertation, University of Amsterdam, Amsterdam*, Aug. 2001.

[7] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, *Tensor robust principal component analysis with a new tensor nuclear norm*, 2019. arXiv: 1804.03728 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1804.03728.

[8] J. Synge and A. Schild, *Tensor Calculus* ((Dover books on Mathematics)). Dover Publications, 1978, ISBN: 9780486636122. [Online]. Available: https://books.google.ru/books?id=8vlGhlxqZjsC.

[9] D. Ahn, J.-G. Jang, and U. Kang, "Time-aware tensor decomposition for sparse tensors," *Machine Learning*, vol. 111, no. 4, pp. 1409–1430, 2022, ISSN: 1573-0565. DOI: 10.1007/s10994-021-06059-7. [Online]. Available: https://doi.org/10.1007/s10994-021-06059-7.

[10] Z. he, J. Li, and L. Liu, "Tensor block-sparsity based representation for spectral-spatial hyperspectral image classification," *Remote Sensing*, vol. 8, p. 636, Aug. 2016. DOI: 10.3390/rs8080636.

[11] R. Bellman, *Dynamic Programming*. Dover Publications, 1957, ISBN: 9780486428093

[12] S. R. White, "Density matrix formulation for quantum renormalization groups," *Phys. Rev. Lett.*, vol. 69, pp. 2863–2866, 19 Nov. 1992. DOI: 10.1103/PhysRevLett.69.2863. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.69.2863.

[13] A. H. Phan, K. Sobolev, K. Sozykin, *et al.*, "Stable low-rank tensor decomposition for compression of convolutional neural network," *CoRR*, vol. abs/2008.05441, 2020. arXiv: 2008.05441. [Online]. Available: https://arxiv.org/abs/2008.05441.

[14] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927. DOI: https://doi.org/10.1002/sapm192761164. eprint: https:

//onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192761164. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164.

[15] J. D. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970. DOI: 10.1007/BF02310791. [Online]. Available: https://doi.org/10.1007/BF02310791.

[16] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966. DOI: 10.1007/BF02289464.

[17] M. Mozaffari and P. P. Markopoulos, "Robust barron-loss tucker tensor decomposition," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, pp. 1651–1655. DOI: 10.1109/IEEECONF53345.2021.9723232.

[18] J. T. Barron, *A general and adaptive robust loss function*, 2019. arXiv: 1701.03077 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1701.03077.

[19] B. Wu, D. Wang, G. Zhao, L. Deng, and G. Li, "Hybrid tensor decomposition in neural network compression," *Neural Networks*, vol. 132, pp. 309–320, Dec. 2020, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.09.006. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2020.09.006.

[20] S. Ahmadi-Asl, C. F. Caiafa, A. Cichocki, *et al.*, "Cross Tensor Approximation Methods for Compression and Dimensionality Reduction," *IEEE Access*, vol. 9, pp. 150 809–150 838, Jan. 2021. DOI: 10.1109/ACCESS.2021.3125069.

[21] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "Bci2000: A general-purpose brain-computer interface (bci) system,"

IEEE Transactions on Biomedical Engineering, vol. 51, no. 6, pp. 1034–1043, Jun. 2004. DOI: 10.1109/TBME.2004.827072.