

Факультет  
компьютерных  
наук

Правительство Российской Федерации  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«Национальный исследовательский университет  
«Высшая школа экономики»

Отчет об учебной практике

## Реализация алгоритма полилинейного сингулярного разложения

Выполнил студент группы БПМИИ 215: *Бугаев Егор Петрович*

Руководитель практики: *Авдеев Роман Сергеевич*

Дата: 03.09.2022

Оценка: 10

# Содержание

|           |   |           |
|-----------|---|-----------|
| <b>1</b>  | <b>Аннотация</b>                            | <b>3</b>  |
| <b>2</b>  | <b>Цели практики</b>                        | <b>3</b>  |
| <b>3</b>  | <b>Задачи практики</b>                      | <b>3</b>  |
| <b>4</b>  | <b>Изученные материалы</b>                  | <b>3</b>  |
| <b>5</b>  | <b>Методы решения задач</b>                 | <b>4</b>  |
| 5.1       | HOSVD . . . . .                             | 4         |
| 5.2       | Multiple SVD . . . . .                      | 5         |
| 5.3       | Детали реализации . . . . .                 | 5         |
| <b>6</b>  | <b>Полученные результаты</b>                | <b>5</b>  |
| 6.1       | Размеры сжатых изображений . . . . .        | 6         |
| 6.2       | Пример сжатия изображения . . . . .         | 7         |
| 6.3       | Норма Фробениуса от потерь сжатия . . . . . | 8         |
| <b>7</b>  | <b>Заключение</b>                           | <b>10</b> |
| <b>8</b>  | <b>Список использованных источников</b>     | <b>11</b> |
| <b>9</b>  | <b>Приложение</b>                           | <b>11</b> |
| <b>10</b> | <b>План-график прохождения практики</b>     | <b>12</b> |

# 1 Аннотация

Практика выполнена студентом образовательной программы Прикладная Математика и Информатика (01.03.02) Факультета Компьютерных Наук *Бугаевым Егором* под руководством доцента департамента больших данных и информационного поиска ФКН ВШЭ *Авдеева Романа Сергеевича*.

В рамках данной практики была прочитана статья [1], реализован описанный там алгоритм нахождения HOSVD (Higher-order singular value decomposition) тензора.

В качестве приложения данного алгоритма выбрано сжатие PNG изображений. Придуман и реализован альтернативный (более простой) способ сжатия, проведены экспериментальные сжатия нескольких изображений. Проанализированы результаты обоих алгоритмов, сделаны выводы об их эффективности.

# 2 Цели практики

Овладение основами теории тензоров и реализация на языке Python алгоритма полилинейного сингулярного разложения из статьи [1]. Применение алгоритма в целях уменьшения расхода памяти на хранение изображений на компьютере.

# 3 Задачи практики

- Освоить базовые понятия тензорных вычислений.
- Реализовать алгоритм полилинейного сингулярного разложения тензоров (далее HOSVD).
- С помощью HOSVD написать сжатие изображение формата RGB.
- Сравнить эффективность сжатия с помощью HOSVD с эффективностью сжатия с помощью многократного применения обычного матричного сингулярного разложения (SVD).

# 4 Изученные материалы

Основным объектом изучения является статья [1] и приведенные там определения из теории тензоров. В качестве вспомогательного материала для ознакомления с операцией Kronecker Product и ее свойствами была изучена статья [2].

Для изучения способов нахождения классического SVD были изучены статьи в Интернете: [3], [4].

Для реализации алгоритма HOSVD на практике были изучены способы работы с тензорами в библиотеке *numpy* для языка программирования Python. Также была использована библиотека *matplotlib* для удобного представления результатов численных экспериментов. Нужные опции обеих библиотек были изучены с помощью официальной документации в Интернете.

## 5 Методы решения задач

### 5.1 HOSVD

Разложение тензора в HOSVD позволяет представить тензор как произведение нового тензора и нескольких ортонормированных матриц. Новый тензор обладает набором свойств, важнейшим из которых для данной работы является некоторое обобщение свойства оптимальности низкорангового приближения у обычного SVD (в статье [1] оно указано как свойство 10: 'approximation'). В частности оно означает, что занулив несколько наименьших 'n-mode singular values' (т.е. откинув часть нового тензора и часть дополняющих матриц), при восстановлении тензора по полученному сокращенному HOSVD получится достаточно хорошее приближение.

Хотя, в отличие от классического SVD, в многомерном случае приближение тензора путем откидывания младших сингулярных значений не является оптимальным (доказательство приведено в статье [1]), оно все еще дает достойные результаты, что хорошо можно заметить на примерах ниже.

Для реализации HOSVD в первую очередь было необходимо представить тензор развернутым в двумерный массив. Для этого были использованы функции библиотеки *numpy*.

Получение HOSVD было реализовано в соответствии с секцией *Computation* статьи [1]. Подсчет HOSVD заданного тензора требует многократного нахождения классических SVD его различных разверток. Переход от первой развертки к остальным осуществляется последовательно, в процессе столбцы предыдущей развертки переходят в подряд идущие части строк новой развертки, что позволяет выполнять переход за оптимальное (линейное от количества значений в тензоре) время.

Классическое SVD было получено с помощью библиотеки *numpy*, так как это позволяет достичь оптимальной производительности. Как альтернативные варианты были изучены базовая и ускоренная версии метода степенных итераций для нахождения SVD.

## 5.2 Multiple SVD

В качестве альтернативного варианта сжатия изображений цветное изображение представляется как 4 одноцветных (матрицы интенсивности красного, зеленого, синего цветов и прозрачности). В дальнейшем каждое из одноцветных изображений представляется в формате SVD, и откидывается часть меньших sigma значений вместе с их векторами (стандартный метод сжатия изображений с помощью SVD).

## 5.3 Детали реализации

Для считывания PNG изображений использовалась библиотека *matplotlib*. В дальнейшем изображение представлялось как трехмерный тензор, к которому и применялся алгоритм нахождения HOSVD.

Оба алгоритма сжатия используют классическую для SVD стратегию сжатия: отбрасывается часть маленьких sigma значений. Чтобы стандартизировать отбрасываемую часть, введен параметр '*sigma compression rate*', означающий, сколько от sigma значений алгоритм оставляет. Например, *sigma compression rate*, равный 2, означает, что отбросится ровно половина sigma значений (и, соответственно, умножаемых на них векторов/матриц). В случае обычного SVD отбрасываются меньшие по величине значения, потому что, в соответствии с теоремой о низкоранговом приближении, они несут меньший вклад в исходную матрицу.

Так как в многомерном случае имеют место только 'n-mode' сингулярные значения, которые определены и отсортированы отдельно для каждого измерения, сжатие HOSVD в данной практике предполагает оставить описанную выше часть вдоль каждого из измерений, кроме последнего. Последнее измерение в случае PNG изображений имеет размер 4, что обусловлено наличием всего 3 цветов и прозрачности, и сжатие вдоль него не представляется возможным.

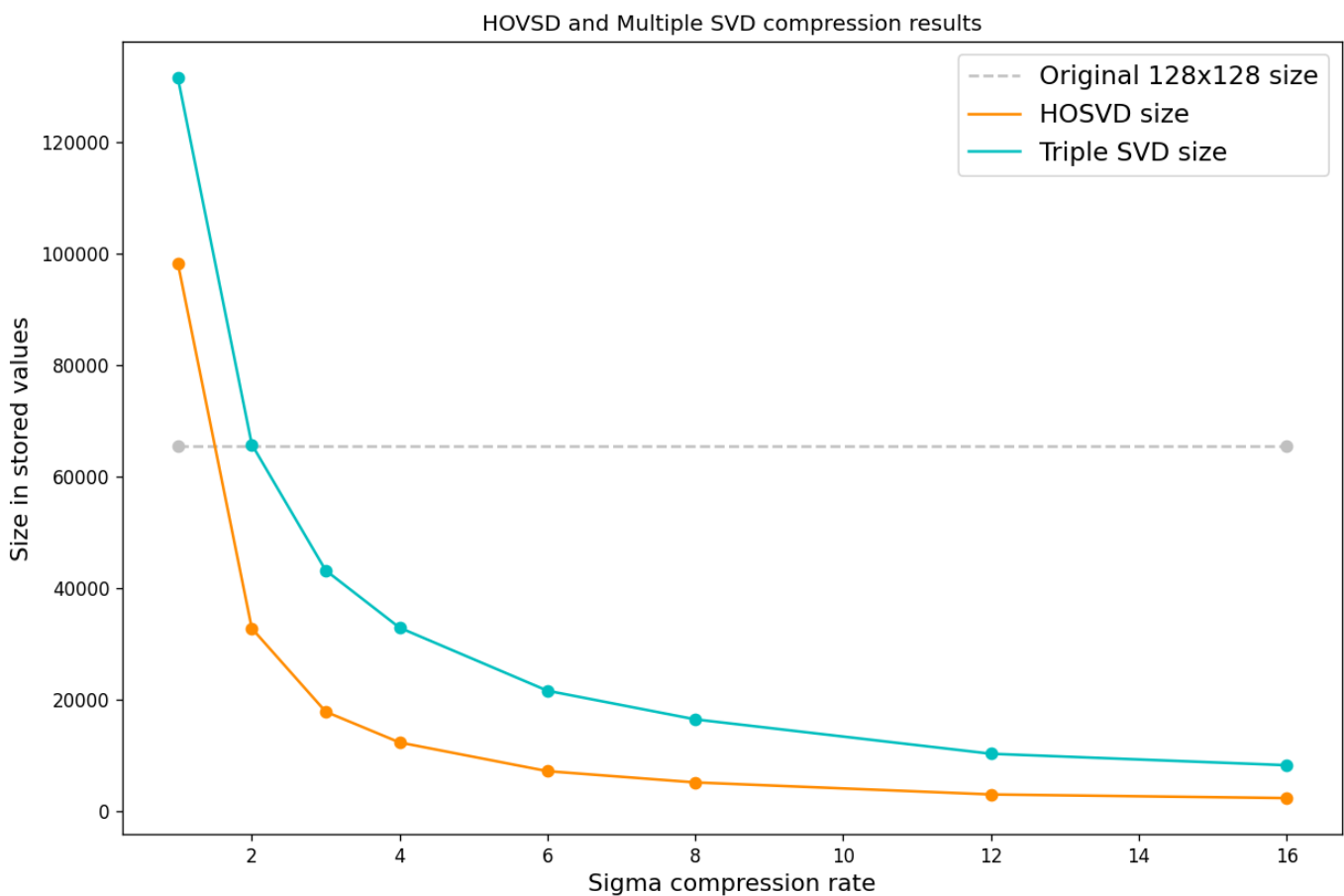
## 6 Полученные результаты

Оба алгоритма работают лишь на небольших изображениях (до 128x128), так как предполагают нахождение SVD, что является затратной в компьютерных ресурсах операцией.

Для сравнения эффективности двух алгоритмов проведено сжатие 5 картинок 64x64 и 7 картинок 128x128 с 7 разными *sigma compression rate* (2, 3, 4, 6, 8, 12, 16). Здесь продемонстрирована только часть полученных значений, оставшиеся результаты можно найти в приложении.

## 6.1 Размеры сжатых изображений

Первым рассмотрим график, демонстрирующий размеры сжатой картинки 128x128 при различных параметрах compression rate. Хотя для HOSVD sigma compression rate означает степень сжатия вдоль каждого из двух измерений, не стоит забывать, что в Multiple SVD происходит сжатие сразу каждой из 4х матриц, соответствующих одноцветным картинкам.



Легко заметить, что при одинаковых параметрах sigma compression rate сжатие с помощью HOSVD значительно эффективнее. При этом сжатие без потерь (т.е. при compression rate равном 1) в обоих случаях требует даже дополнительной памяти, так как необходимо хранить вспомогательные матрицы.

## 6.2 Пример сжатия изображения

Теперь рассмотрим сжатие конкретной картинки 128x128 при заданных значениях sigma compression rate с помощью каждого из алгоритмов.



Рис. 1: Оригинальная картинка



(a) HOSVD, 2



(b) Multiple SVD, 2



(c) HOSVD, 4



(d) Multiple SVD, 4

Рис. 2: Примеры сжатия (алгоритм, sigma compression rate)



(a) HOSVD, 8



(b) Multiple SVD, 8



(c) HOSVD, 16



(d) Multiple SVD, 16

Рис. 3: Примеры сжатия (алгоритм, sigma compression rate)

Визуально разница в качестве сжатия незначительна. Действительно, с параметром sigma compression rate равным 2 и 4 оба алгоритма сжали картинку без сильных потерь, ее все еще можно использовать. При sigma compression rate 8 и 16 оба алгоритма сильно испортили картинку, исходный объект узнать можно, но четкость потеряна. Примеры сжатия других картинок можно найти в приложении.

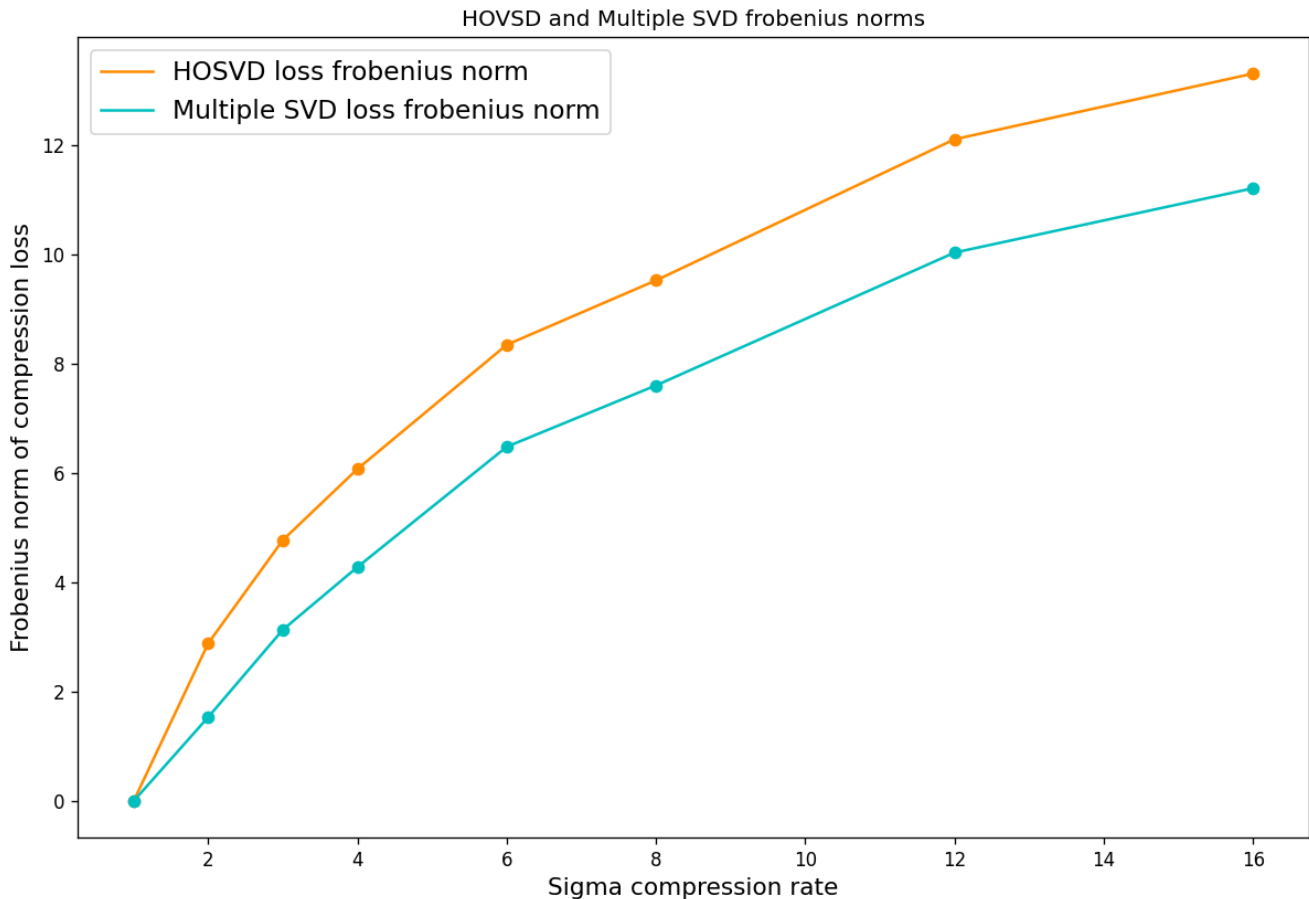
### 6.3 Норма Фробениуса от потерь сжатия

Наконец, рассмотрим, какое в среднем отклонение по норме Фробениуса (определение нормы для многомерного случая дано в статье [1]) получается у новой картинки по сравнению с оригинальной при сжатии каждым из алгоритмов. Это поможет формальнее понять, какой из алгоритмов генерирует меньше помех при сжатии.

Для расчетов взяты 5 картинок 64x64 (ради уменьшения вычислительных затрат в этом случае перешли к меньшему формату), параметры sigma



compression rate такие же, как и выше. Отклонение по норме от оригинала при заданном sigma compression rate берется среднее арифметическое из значений для 5 картинок. Для каждой картинке норма Фробениуса вычисляется от поэлементной разности тензора, полученного после восстановления сжатой картинке, и исходного тензора.



При анализе данного графика важно учитывать, что каждое изображение представляет собой  $64 \times 64 \times 4$  числа от 0 до 1. При таком масштабе норма Фробениуса, равная 10, указывает на существенное отклонение.

Видно, что HOSVD сжимает изображения менее аккуратно. Несмотря на отсутствие явных визуальных различий, при каждом значении sigma compression rate алгоритм Multiple SVD показал лучшие с точки зрения качества результаты.

Заметим, что скорость роста ошибки в обоих случаях примерно одинаковая. Отсюда и появляется малая визуальная разница: при малых  $\sigma$  compression rate ошибку еще не видно, а при больших погрешность уже настолько велика, что оба изображения сильно отличаются от оригинального.

## 7 Заключение

Результаты этой практики показывают, что алгоритм с HOSVD можно успешно применять для сжатия изображений размерами до  $128 \times 128$ . При параметрах  $\sigma$  compression rate до 4 качество картинки падает незначительно, но пользователь получает кратный выигрыш в занимаемой памяти.

Алгоритм Multiple SVD тоже показывает достойные результаты. Обладая выигрышем в качестве при равном показателе  $\sigma$  compression rate, он все же требует существенно больше памяти, чем сжатие с помощью HOSVD. Учитывая, что параметр  $\sigma$  compression rate легко изменяем, меньшую погрешность сжатия можно получить, просто изменив параметр для алгоритма HOSVD (чтобы узнать, какой алгоритм покажет лучшее сжатие при сравнимой точности, требуются дальнейшие эксперименты).

Так как HOSVD подходит для сжатия произвольных тензоров, алгоритмы, примененные в данной работе, могут использоваться для сжатия не только картинок, но и, например, сигналов (что и указано как применение в статье [1]). Ограничение накладывает лишь вычислительная стоимость, так как для нахождения HOSVD (и Multiple SVD) необходимо многократное вычисление классического SVD, что затратно по компьютерным ресурсам.

Дальнейшая работа может подразумевать попытку сократить количество вычислительных мощностей, необходимых для сжатия. Хорошей целью будет сделать алгоритм, способный схожими методами сжимать изображения больших разрешений.

Иным направлением может послужить автоматизации изменения параметра  $\sigma$  compression rate. Как известно из двумерного случая, стоит откидывать те сингулярные значения, которые на порядок меньше предыдущих (тогда потери будут минимальны). В многомерном случае можно попробовать сделать анализирующую функцию, которая будет проводить сжатие постепенно, проверяя, насколько падает качество, и останавливаться перед существенным его снижением.






## 8 Список использованных источников

- [1] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, pp. 1253–1278, Jan. 2000.
- [2] P. A. Regalia and M. K. Sanjit, "Kronecker products, unitary matrices and signal processing applications," *SIAM Review*, vol. 31, pp. 586–613, Dec. 1989.
- [3] Risto Hinno, Towards Data Science, "Simple svd algorithms," 2021. [Онлайн; прочитано 9-Июль-2022].
- [4] Wikipedia contributors, "Power iteration — Wikipedia, the free encyclopedia," 2022. [Онлайн; прочитано 10-Июль-2022].

## 9 Приложение

Больше тестов и полные результаты численных экспериментов, а также код алгоритмов HOSVD и Multiple SVD можно найти по ссылке в репозитории:  
<https://github.com/EgorBugaev/HOSVD-image-compression>

## 10 План-график прохождения практики

| № | Сроки проведения | Планируемые работы   | Подпись руководителя   |
|---|------------------|--|--|
| 1 | 01.07.2022       | Инструктаж по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами трудового распорядка.   |   |
| 2 | 02-05.07.2022    | Ознакомление с основными понятиями, связанными с тензорами в линейной алгебре.   |   |
| 3 | 06.07.2022       | Изучение статьи [1].   |   |
| 4 | 07-13.07.2022    | 1. Реализация функции для работы с тензорами, представленными многомерными данными.<br>2. Реализация алгоритма полилинейного сингулярного разложения и проведение численных экспериментов. |   |
| 5 | 14.07.2022       | Представление результатов численных экспериментов в удобном и читаемом формате, подготовка отчета о прохождении практики.  |  |

3 сентября 2022 г.



Бугаев Егор Петрович



Авдеев Роман Сергеевич