Сжатие с помощью тензорного поезда и сравнение с SVD/HOSVD

Кирилл Бродт
Сергей Кудашев
Радик Батраев
Сергей Фамуляк
Павел Никишкин

Постановка задачи

- ТТ-формат для таблицы A

$$A(i_1, \ldots, i_d) = \sum_{\alpha_0, \ldots, \alpha_d} G_1(\alpha_0, i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \ldots G_d(\alpha_{d-1}, i_d, \alpha_d)$$

- G: ТТ-ядра
- r_i: ТТ-ранги
- r = max r_i: максимальный ТТ-ранг
- ТТ-формат использует O(dnr^2) памяти для хранения O(n^d) элементов
- Эффективен, если ранг небольшой!

Методы

- SVD
- HOSVD (разложение Таккера)
- Тензорный поезд (TT)

*реализации используют только scipy.linalg.svd и умножения таблиц numpy.tensordot

HOSVD (разложение Таккера)

- For $m = 0, 1, \ldots, M$, do the following:
  1. Construct the mode-*m* flattening $\mathcal{A}_{[m]}$;
  2. Compute the (compact) singular value decomposition $\mathcal{A}_{[m]} = \mathbf{U}_m \mathbf{\Sigma}_m \mathbf{V}_m^T$, and store the left singular vectors $\mathbf{U} \in \mathbb{C}^{I_m \times R_m}$;
- Compute the core tensor $\mathcal{S}$ via the multilinear multiplication $\mathcal{S} = \mathcal{A} \times_0 \mathbf{U}_0^H \times_1 \mathbf{U}_1^H \times_2 \mathbf{U}_2^H \ldots \times_m \mathbf{U}_m^H \ldots \times_M \mathbf{U}_M^H$

https://en.wikipedia.org/wiki/Higher-order_singular_value_decomposition

TT

Algorithm 1. TT-SVD.

Require: d-dimensional tensor $\mathbf{A}$, prescribed accuracy ε.

Ensure: Cores $G_1, \ldots, G_d$ of the TT-approximation $\mathbf{B}$ to $\mathbf{A}$ in the TT-format with TT-ranks $\widehat{r}_k$ equal to the δ-ranks of the unfoldings A_k of $\mathbf{A}$, where $\delta = \frac{\varepsilon}{\sqrt{d-1}}||A||_F$. The computed approximation satisfies

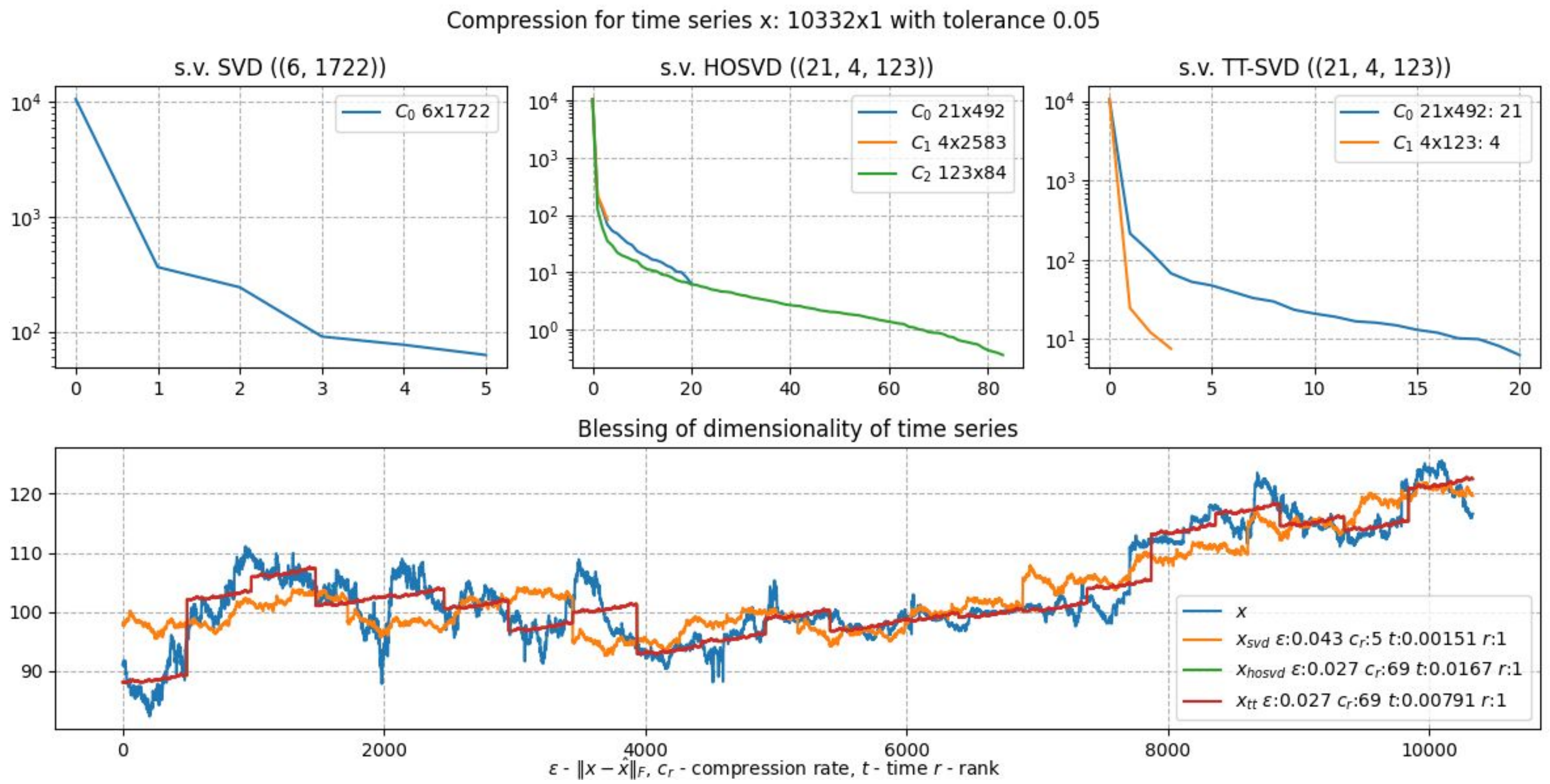
$$||\mathbf{A} - \mathbf{B}||_F \leq \varepsilon ||\mathbf{A}||_F.$$

```
{Initialization}
Compute truncation parameter δ = ε/√(d−1) ||A||_F.
Temporary tensor: C = A, r_0 = 1.
for k = 1 to d − 1 do
    C := reshape(C, [r_{k−1} n_k, numel(C)/(r_{k−1} n_k)]).
    Compute δ-truncated SVD: C = USV + E, ||E||_F ≤ δ, r_k = rank_δ(C).
    New core: G_k := reshape(U, [r_{k−1}, n_k, r_k]).
    C := SV^⊤.
end for
G_d = C.
Return tensor B in TT-format with cores G_1, ..., G_d.
```

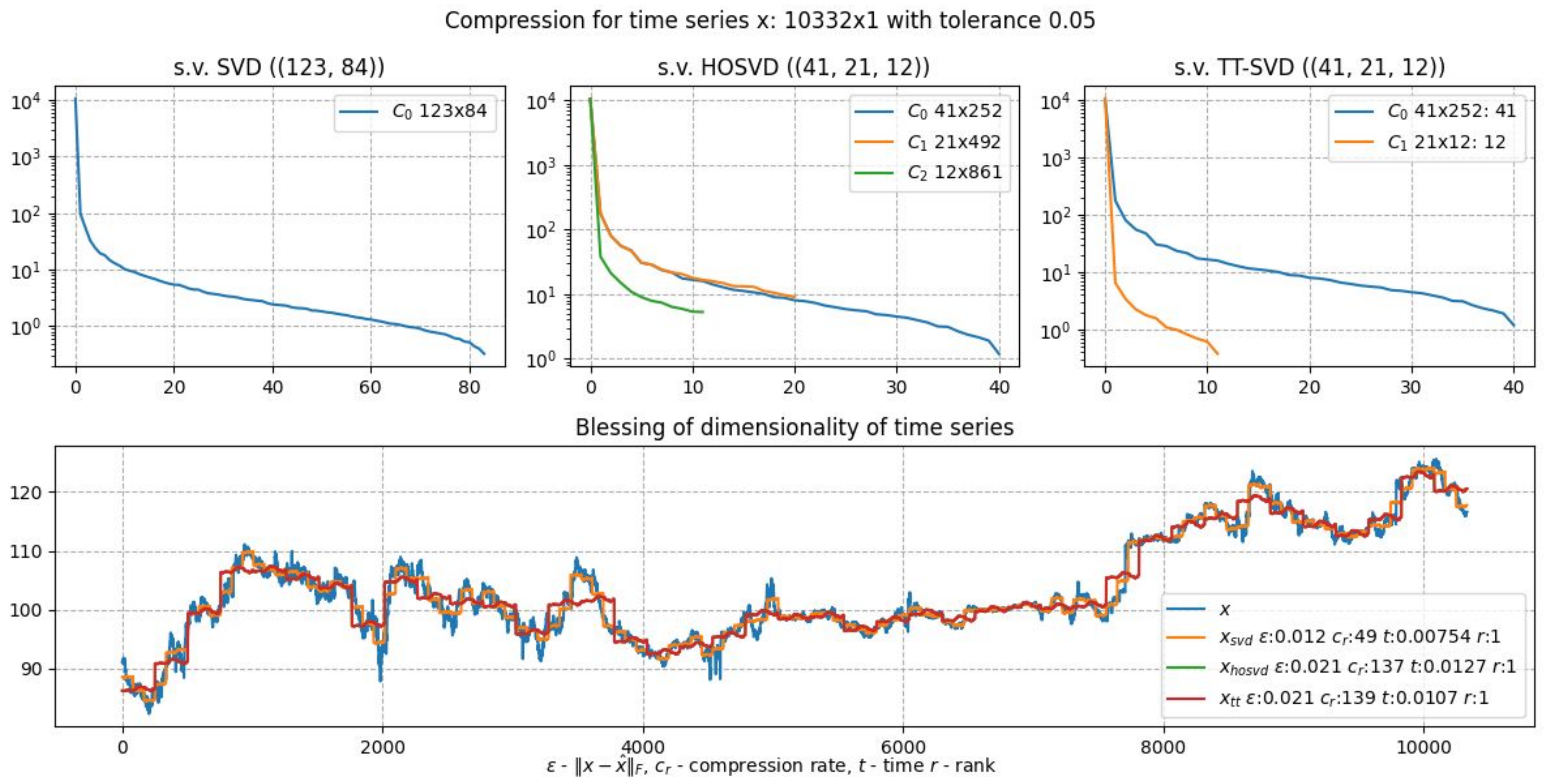
[2011, V. Oseledets, Tensor-Train decomposition]

Приложения: временной ряд

- x # N
- X = x.reshape(n1, n2) # n1 x n2 = N
- X = x.reshape(n1, n2, n3) # n1 x n2 x n3 = N

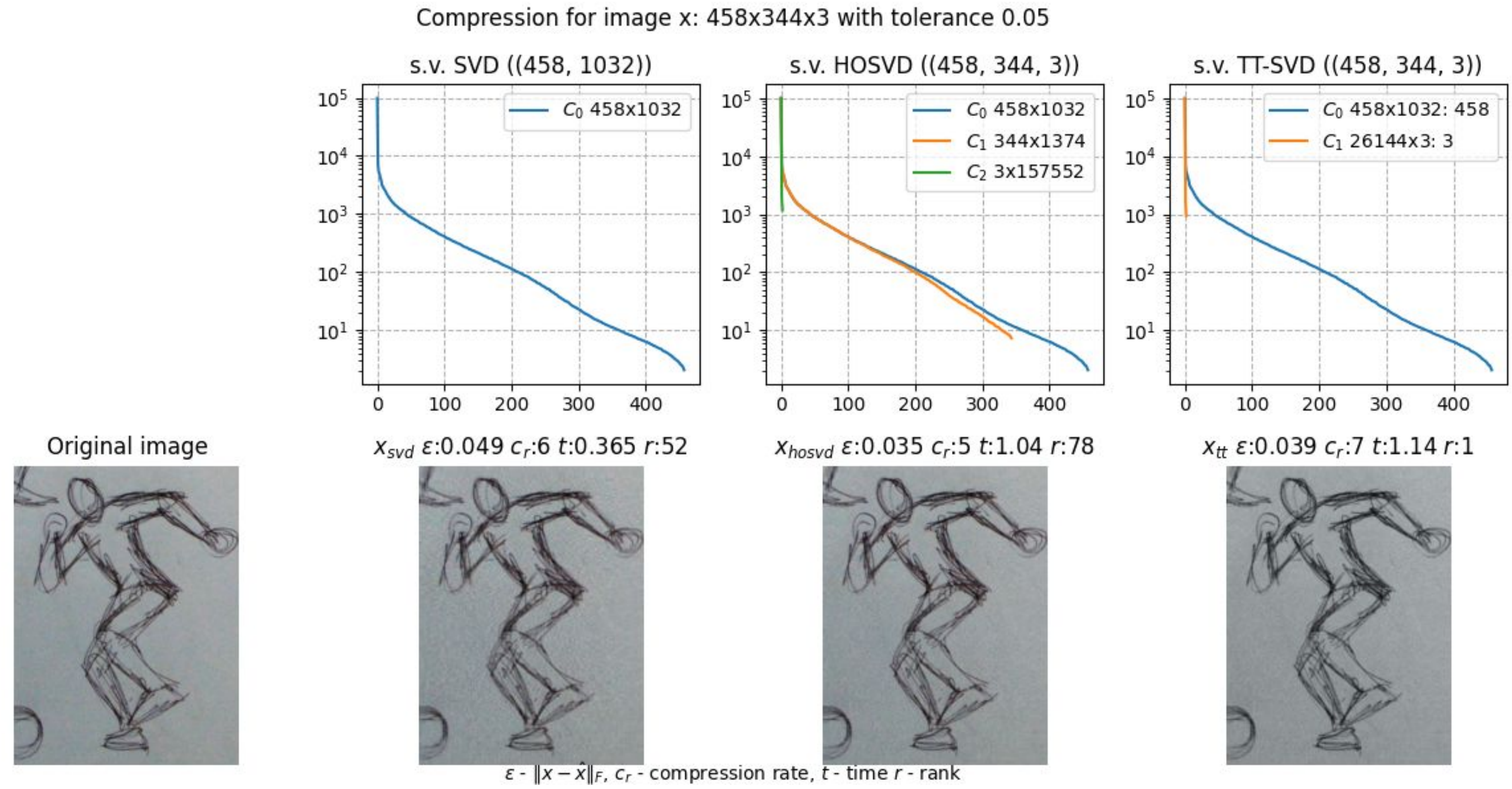


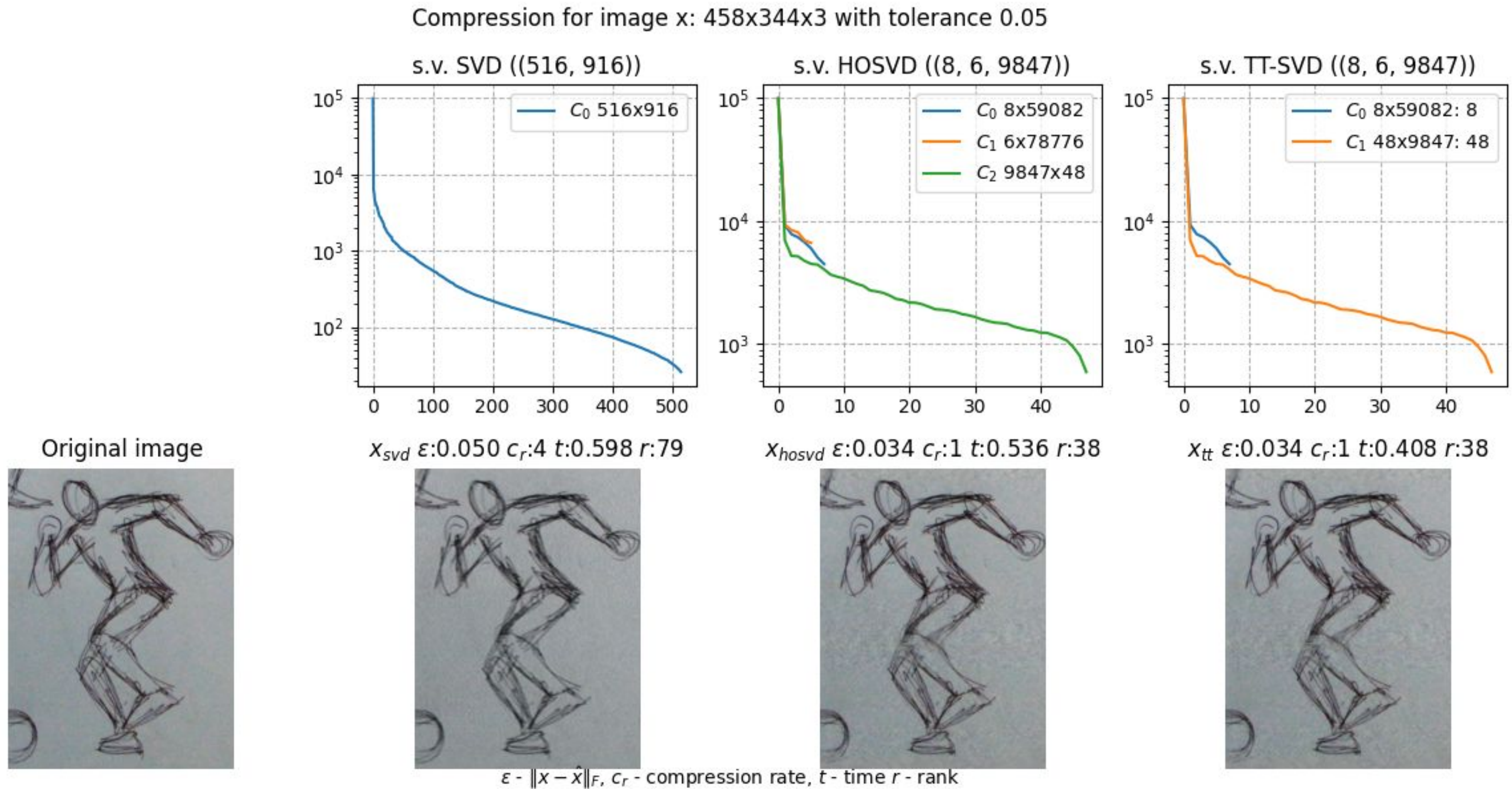
Compression for time series x: 10332x1 with tolerance 0.05
s.v. SVD ((6, 1722))
C_0 6x1722
s.v. HOSVD ((21, 4, 123))
C_0 21x492
C_1 4x2583
C_2 123x84
s.v. TT-SVD ((21, 4, 123))
C_0 21x492: 21
C_1 4x123: 4
Blessing of dimensionality of time series
x
x_{svd} ε:0.043 c_r:5 t:0.00151 r:1
x_{hosvd} ε:0.027 c_r:69 t:0.0167 r:1
x_{tt} ε:0.027 c_r:69 t:0.00791 r:1
ε - $\|x - \hat{x}\|_F$, c_r - compression rate, t - time r - rank

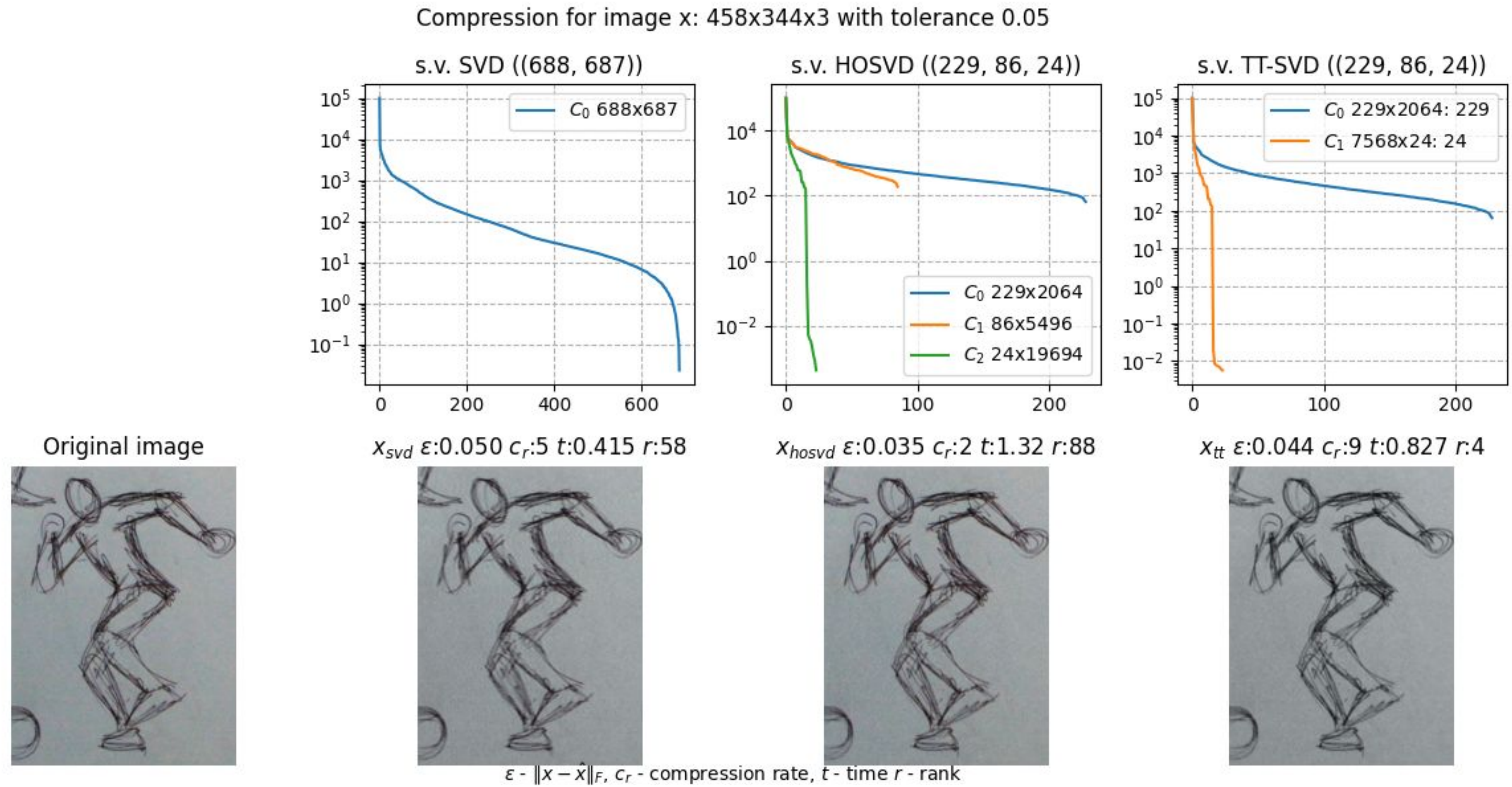

Compression for time series x: 10332x1 with tolerance 0.05
s.v. SVD ((123, 84))
C_0 123x84
s.v. HOSVD ((41, 21, 12))
C_0 41x252
C_1 21x492
C_2 12x861
s.v. TT-SVD ((41, 21, 12))
C_0 41x252: 41
C_1 21x12: 12
Blessing of dimensionality of time series
x
x_{svd} ε:0.012 c_r:49 t:0.00754 r:1
x_{hosvd} ε:0.021 c_r:137 t:0.0127 r:1
x_{tt} ε:0.021 c_r:139 t:0.0107 r:1
ε - $\|x-\hat{x}\|_F$, c_r - compression rate, t - time r - rank

Приложения: изображение

- x # [H, W, 3]
- X = x.reshape(n1, n2) # n1 x n2 = H x W x 3
- X = x.reshape(n1, n2, n3) # n1 x n2 x n3 = = H x W x 3

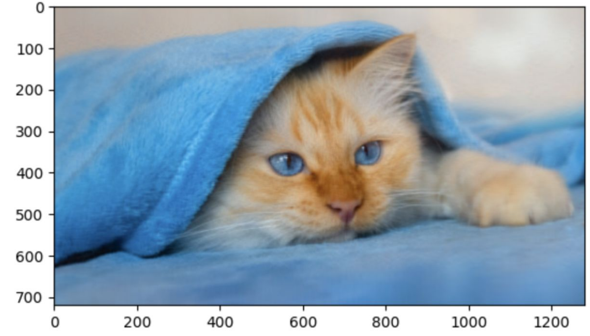

Compression for image x: 458x344x3 with tolerance 0.05
s.v. SVD ((458, 1032))
C_0 458x1032
s.v. HOSVD ((458, 344, 3))
C_0 458x1032
C_1 344x1374
C_2 3x157552
s.v. TT-SVD ((458, 344, 3))
C_0 458x1032: 458
C_1 26144x3: 3
10^5
10^4
10^3
10^2
10^1
0
100
200
300
400
Original image
x_{svd} ε:0.049 c_r:6 t:0.365 r:52
x_{hosvd} ε:0.035 c_r:5 t:1.04 r:78
x_{tt} ε:0.039 c_r:7 t:1.14 r:1
ε - $\|x - \hat{x}\|_F$, c_r - compression rate, t - time r - rank


Compression for image x: 458x344x3 with tolerance 0.05
s.v. SVD ((516, 916))
C_0 516x916
s.v. HOSVD ((8, 6, 9847))
C_0 8x59082
C_1 6x78776
C_2 9847x48
s.v. TT-SVD ((8, 6, 9847))
C_0 8x59082: 8
C_1 48x9847: 48
Original image
x_{svd} ε:0.050 c_r:4 t:0.598 r:79
x_{hosvd} ε:0.034 c_r:1 t:0.536 r:38
x_{tt} ε:0.034 c_r:1 t:0.408 r:38
ε - $\|x-\hat{x}\|_F$, c_r - compression rate, t - time r - rank


Compression for image x: 458x344x3 with tolerance 0.05
s.v. SVD ((688, 687))
C_0 688x687
s.v. HOSVD ((229, 86, 24))
C_0 229x2064
C_1 86x5496
C_2 24x19694
s.v. TT-SVD ((229, 86, 24))
C_0 229x2064: 229
C_1 7568x24: 24
Original image
x_{svd} ε:0.050 c_r:5 t:0.415 r:58
x_{hosvd} ε:0.035 c_r:2 t:1.32 r:88
x_{tt} ε:0.044 c_r:9 t:0.827 r:4
ε - $\|x - \hat{x}\|_F$, c_r - compression rate, t - time r - rank

Использование квантования в сжатии изображений

```
  3  7201280
  |   |   |
 (0) (1) (2)
 / \ / \ / \
1   3   85  1

Compression ratio: 2764800/292409 = 9.45525
Relative error: tensor(0.0233)
RMSE: tensor(0.0144)
R^2: tensor(0.9956)
```



```
CPU times: user 33.9 s, sys: 20.7 s, total: 54.6 s
Wall time: 46.4 s
```

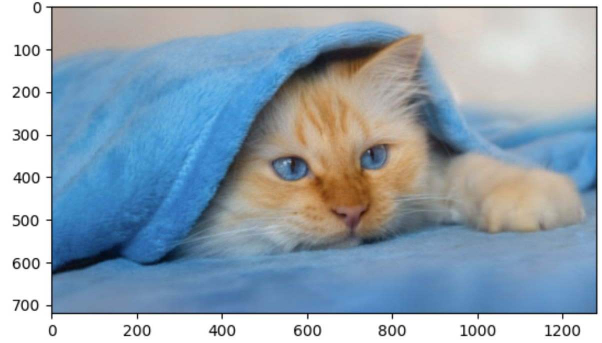
Результат сжатия тензора исходных размеров

```
  3   2   2   2   2   3   3   5   2   2   2   2   2   2   2   2   5
  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 (0) (1) (2) (3) (4) (5) (6) (7) (8) (9) (10)(11)(12)(13)(14)(15)(16)
 / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \
1   3   6   12  24  48  115 115 115 115 115 115 80  40  20  10  5   1

Compression ratio: 2764800/231704 = 11.9325
Relative error: tensor(0.0231)
RMSE: tensor(0.0143)
R^2: tensor(0.9957)
```

```
CPU times: user 2.82 s, sys: 317 ms, total: 3.14 s
Wall time: 1.72 s
```

Результат сжатия тензора после квантования на простые множители

```
 12   4   9   5   4   4  16   5
  |   |   |   |   |   |   |   |
 (0) (1) (2) (3) (4) (5) (6) (7)
 / \ / \ / \ / \ / \ / \ / \ / \
1   12  48  110 110 110 80  5   1

Compression ratio: 2764800/200493 = 13.79
Relative error: tensor(0.0231)
RMSE: tensor(0.0142)
R^2: tensor(0.9957)
```



```
CPU times: user 2.58 s, sys: 281 ms, total: 2.86 s
Wall time: 1.56 s
```

Результат сжатия тензора на оптимальном квантовании

Главный вопрос – поиск оптимального квантования для конкретных размеров изображения. В случае, когда размеры хорошо раскладываются на простые множители, начинать поиск удобно с такого разложения. В противном случае – открытый вопрос.

Выводы

- Реализованы 2 метода разложения многомерных таблиц
- Сжатие возможно, но нужно подбирать размерности с умом!
- ТТ чуть быстрее HOSVD*, так как делает на одно SVD разложение меньше (опять же, зависит от значений размерностей таблиц)

Код

https://github.com/Ulycecc/project_nla/tree/main/project_2

Источники

- TENSOR-TRAIN DECOMPOSITION // V. OSELEDETS
- Тензорные разложения и их применения // YANDEX