А. А. Курилович[1], *С. А. Матвеев*[2],

О ВОЗМОЖНОСТЯХ СЖАТИЯ ВИДЕОДАННЫХ С ПОМОЩЬЮ ТЕНЗОРНЫХ ПОЕЗДОВ И РАЗЛОЖЕНИЯ ТАККЕРА*

Введение

Разработка эффективных методов для сжатия видео остается актуальной задачей в различных мультимедийных приложениях из-за экспоненциального увеличения количества видеоданных, генерируемых современными устройствами [1]. Очевидной целью данного направления исследований является минимизация объема данных, необходимого для представления видео в сжатом формате, который можно эффективно хранить, передавать и декодировать. Сжатие видео с потерями является одним из основных подходов к решению этой задачи. Оно направлено на достижение высокой степени сжатия при минимизации искажений, вносимых в видео, что часто формулируется как задача оптимизация функционала скорость-искажение (rate-distortion). Ее решение позволяет найти оптимальный компромисс между коэффициентом сжатия (CR) и искажением привносимым в видео. Последнее обычно измеряется с использованием различных количественных метрик визуального качества, таких как PSNR [2], SSIM [3] и VMAF [4].

Выбор метода сжатия видео зависит от конкретных требований и ограничений, налагаемых областью применения через целевые CR, визуальное качество и быстродействие. Ведутся активные исследования, направленные на изучение и разрабатотку новых методов для вышеописанной задачи. В течение последних десятилетий были разработаны различные стандарты сжатия видео. Наиболее широко используемые из них основаны на кодировании с преобразованием, например H.264/AVC [5], H.265/HEVC [6] и AV1 [2]. Эти методы основаны

[1]научный сотрудник центра CEST Сколковского института науки и технологий, (текущий статус): научный сотрудник Израильского технологического института Технион, Израиль e-mail: a.a.kurilovich@yandex.ru .

[2]доцент факультета ВМК МГУ имени М. В. Ломоносова; научный сотрудник ИВМ РАН, e-mail: matseralex@cs.msu.ru .

*С. А. Матвеев был поддержан проектом РНФ 21-71-10072, А. А. Курилович выражает благодарность академии Больших Данных компании Vk за возможность подготовить данное исследование в качестве выпускного проекта.

[2]Спецификация доступна по ссылке

на блочных преобразованиях, квантовании, энтропийном кодировании и прогнозировании для достижения высоких коэффициентов сжатия.

Методы глубокого обучения так же недавно показали многообещающие результаты в сжатии видео с потерями [1]. Эти методы используют нейронные сети для обучения процессу сжатия непосредственно из данных, что позволяет избежать сложного конвейера используемого в традиционных методах. В качестве примеров можно привести стандарт VVC (Versatile Video Coding) [7] и фреймворк Neural Video Compression (NVC) [8]. VVC использует нейронные сети для внутреннего прогнозирования и циклической фильтрации. NVC основывается на применении генеративной модели для сжатия видеокадров. Методы, основанные на глубоком обучении, являются относительно новыми для области сжатия видео, и их ограничения для промышленного использования еще находятся в процессе изучения. Например, они могут работать хуже при использовании “атак” (adversarial attacks), а также из-за слабой интерпретируемости процесса обучения. Одним из возможных путей решения этой проблемы является применение различных методов визуализации [9] и анализа чувствительности [10] к предварительно обученным нейронным сетям и даже к фиксированным нейронам в рамках изучаемых архитектур. В целом, эти относительно новые методы пригодны для сжатия видео с потерями, но у них также есть нерешенные проблемы, которые только предстоит устранить. Это так же делает актуальным поиск альтернативных и/или комплементарных подходов к сжатию видео.

Один из таких подходов основывается на применении тензорных разложений, продемонстрировавших перспективные результаты в задачах компрессии различных многомерных данных (например, гиперспектральных снимков [12], [13], [11], параметров нейросетей [15], [14] и т. д.). Тензорные разложения позволяют использовать естественную структуру многомерных данных для их представления в сжатой форме при сохранении основных характеристик. Алгоритмы их построения в настоящее время достаточно хорошо изучены. Для этих алгоритмов известны также оценки на точность и устойчивость искомых приближений. Наиболее распространенными форматами тензорных разложений являются канонический полилинейный формат(см., например, обзор [16]), разложение Таккера [17, [18] и разложение тензорного поезда (TT) [19], [20]. Последние два формата особенно интересны для практических приложений благодаря наличию вычислительно эффективных алгоритмов, позволяющих аналитически контролировать выбор точности аппроксимации, а именно st-HOSVD [21] и TTSVD [22] соответственно. Эти алгоритмы основаны на последовательном применении сингулярного разложения (SVD) к используемым данным.

Относительная новизна этого подхода в области сжатия видео в сочетании с многообещающими результатами, которые он показал при сжатии других данных высокой размерности, а также наличие эффективных алгоритмов с теоретическими оценками, мотивируют провести исследовательскую работу по применению форматов Таккера и TT для сжатия видео с потерями. Так же в данной работе оценивается результативность данного подхода при помощи сравнительного анализа в совокупности с результатами стандартных методов сжатия видеоданных.

Постановка задачи

Основной целью данной работы является оценка результатов, которые можно получить с использованием тензорных форматов Таккера и TT при решении задачи сжатия видео с потерями. Эффективность предложенного подхода исследуется путем сравнения с существующими реализациями отраслевых стандартов. Для этого был разработан фреймворк, основанный на алгоритмах TTSVD и st-HOSVD. В качестве референсного решения используется реализация `x264` стандарта h264 с открытым исходным кодом. Построенный фреймворк обладает следующими аспектами, учитывающими цели нашего тематического исследования:

(1) Минимальная предварительная обработка видеоданных. Видео преобразуется в формат `RGB24` и разбивается на фрагменты фиксированной длины (по 30 кадров в каждом). Это позволяет обеспечить сжатие видео произвольной длины. Также была применена квантизация для каждого видеофрагмента [20] при использовании TTSVD, чтобы проверить, позволяет ли она улучшить общее качество сжатия в силу того, что TT часто показывает лучшие результаты для сжатия многомерных данных [19].

(2) Аналитический подбор гиперпараметров (рангов) st-HOSVD и TTSVD, обеспечивающих требуемое качество сжатого видео. Это важно для их дальнейшей интеграции в существующие стандарты, так как позволяет исключить вычислительно затраные процедуры оптимизации, обеспечивающие требуемое визуальное качество сжатого видео. Подробное описание приведено в соответствующем разделе этой статьи.

(3) Отсутвие акцента на оптимизацию быстродейсвия. Его улучшение выходит за рамки этой работы, которая в основном сосредоточена на выяснении максимального соотношения качества/сжатия, которое могут обеспечить тензорные форматы. Тем не менее, существуют большие возможности для оптимизации скорости вычислений путем использования более быстрых алгоритмов (например, рандомизированного st-HOSVD [23], параллельных алгоритмов [24] и т. д.) и/или более эффективной программной реализации.

Подробная схема разработанного фреймворка представлена на рис.1.

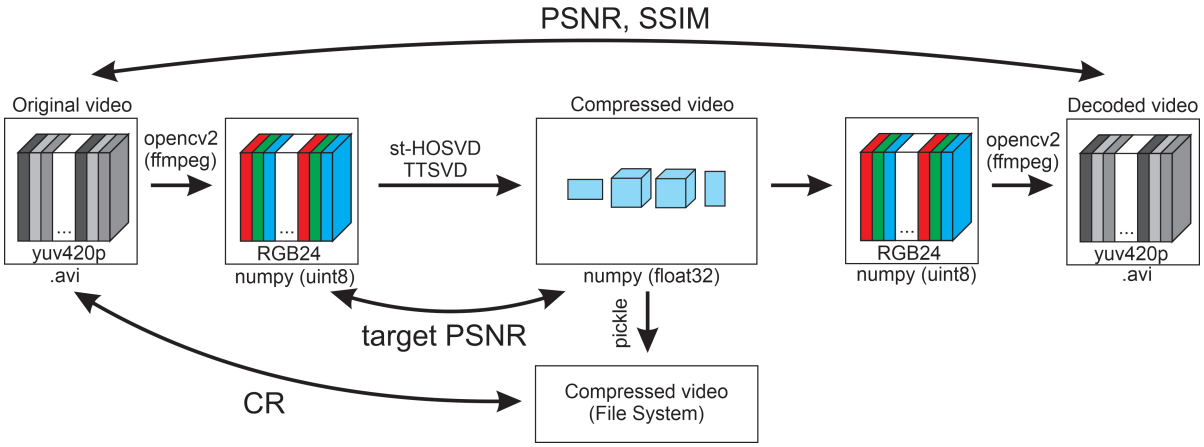

Рис. 1. Схема разработанной фреймворка.

Исходное видео хранится в файловой системе в формате `yuv420p`. Далее оно преобразуется в `RGB24` с помощью библиотеки `opencv-python`, использующей `ffmpeg`. Затем видео сжимается путем применения st-HOSVD или TTSVD с аналитическим подбором рангов, обеспечивающим целевое значение PSNR для видео в формате `RGB24`. Мы используем тип данных `float32` для уменьшения размера матриц/тензоров без заметного увеличения ошибок сжатия по сравнению с `float64`. Сжатое видео сериализуется модулем Python `pickle` и сохраняется в файловой системе. Значение CR вычисляется как отношение размеров сжатого и исходного видео в байтах. Метрики PSNR и SSIM рассчитываются с помощью `ffmpeg` между исходным видео и сжатым видео, которое декодируется в `RGB24` и преобразуется обратно в `yuv420p` с помощью `ffmpeg`. Можно заметить, что такой подход искусственно занижает наши результаты при сравнении со стандартными реализациями, работающими непосредственно с `yuv` и не вносящими при этом дополнительных ошибок при преобразовании между `yuv420p` и `RGB24`. Эту проблему можно преодолеть конвертацией видео в `yuv444`. В этом случае, оно может быть представлено как единый тензор, для работы с которым относительно просто адаптировать разработанный фреймворк. Тем не менее, эти расхождения незначительны и остаются предметом дальнейших исследований.

Сравнение проводится для видео из открытой коллекции с сайта xiph.org[1]. По аналогии с [25], мы используем три видеоролика разной временной и пространственной сложности: *Crowd Run*, *Red Kayak* и *Speed Bag*). Это - короткие видеоролики длинной 500-570 кадров и fps = 30-50 fps, что существенно упрощает инженерную часть без изменения общих выводов, сделанных из сравнительного анализа. Они используются для

[1] `https://media.xiph.org/video/derf/`

оценки результатов сжатия тензорными форматами, сравнения их стандартными кодеками и выявления типичных артефактов сжатия. Сравнение производится с помощью кривых «скорость-искажение» (RD): PSNR vs. коэффициент сжатия (CR); и SSIM vs. CR. Значение CR определяется как отношение размера сжатого (сериализованного библиотекой pickle Python) и необработанного видео (контейнер .avi) в байтах, хранящихся в файловой системе.

Следующие три раздела подробно описывают алгоритмы st-HOSVD, TTSVD и процедуру оценки рангов. Для наглядности детали алгоритмов st-HOSVD и TTSVD сформулированы для трехмерных массивов.

Алгоритм st-HOSVD

Рассмотрим трехмерный массив как тензор:

$$A(i,j,k) \in \mathbb{R}^{N\times N\times N}.$$

Для его хранения требуется N^3 ячеек памяти. Иногда данные имеют структуру, позволяющую параметризовать их в некотором формате, требующем меньшего количества параметров.

Формат Таккера заключается в следующем представлении:

$$A(i,j,k) = \sum_{\alpha_1=1}^{r_1}\sum_{\alpha_2=1}^{r_2}\sum_{\alpha_3=1}^{r_3} G(\alpha_1,\alpha_2,\alpha_3)\; U_1(i,\alpha_1)U_2(j,\alpha_2)U_3(k,\alpha_3). \quad (1)$$

В этом случае следует прокомментировать способ определения полилинейных таккеровских рангов (r_1, r_2, r_3), которые существуют и соответствуют минимальным значениям r_1, r_2, r_3, позволяющим получить поэлементное равенство между разложением и начальным тензором A.

Таким образом, мы должны хранить так называемое ядро Таккера $G(\alpha_1,\alpha_2,\alpha_3) \in \mathbb{R}^{r_1\times r_2\times r_3}$ и факторы Таккера $U_1(i,\alpha_1) \in \mathbb{R}^{N\times r_1}$, $U_2(i,\alpha_2) \in \mathbb{R}^{N\times r_2}$, $U_3(i,\alpha_3) \in \mathbb{R}^{N\times r_3}$. Таким образом, нужно хранить $r_1r_2r_3 + N(r_1 + r_2 + r_3)$ элементов вместо исходных N^3. Большим преимуществом разложения Таккера являются процедуры для его вычисления: SVD высокого порядка (HOSVD) и эквивалентные последовательно усеченные процедуры HOSVD (st-HOSVD). Для обеих процедур сложность составляет $O(N^4)$, но st-HOSVD имеет в 2-3 раза лучшую константу сложности.

Минимальные значения рангов Таккера связаны с рангами специальных матриц развертки, сформированных из исходного тензора

$A(i,j,k)$:

1. $A_1 = A(i,\overline{j,k})$ в `Python:`
`reshape(A,[N,N*N])`,
`SVD потребует` $O(N^4)$ `действий,`
2. $A_2 = A(j,\overline{i,k})$ в `Python:`
`transpose(A,[1,0,2]),reshape(A,[N,N*N])`,
`SVD потребует` $O(N^4)$,
3. $A_3 = A(k,\overline{i,j})$ в `Python:`
`transpose(A,[2,0,1]),reshape(A,[N,N*N])`,
`SVD потребует` $O(N^4)$.

Ранги этих трех матриц развёрток – это в точности минимальные ранги Таккера (r_1, r_2, r_3). Применим усеченное сингулярное разложение для сжатия развертки A_i: $A_i = U_i \Sigma_i V_i^\top$. Здесь U_i и V_i – матрицы с ортонормированными столбцами, а Σ_i — диагональная с сингулярными значениями на главной диагонали. Следовательно,

$$G(\alpha_1, \alpha_2, \alpha_3) = U_1^\top U_2^\top U_3^\top A. \tag{2}$$

Выше описан классический алгоритм HOSVD. Метод st-HOSVD [21] представляет собой последовательное понижение ранга посредством получения скелетных разложений матриц (например, SVD) и операций `reshape`, `transpose` следующим образом:

$$\begin{aligned}
A(i,\overline{j,k}) &= \sum_{\alpha_1}^{r_1} U_1(i_1,\alpha_1)\tilde{G}_{23}(\alpha_1,\overline{j,k}) = \sum_{\alpha_1}^{r_1} U_1(i_1,\alpha_1)\tilde{G}_{23}(\alpha_1,\overline{j,k}) = \\
&= \sum_{\alpha_1}^{r_1} U_1(i_1,\alpha_1)\tilde{G}_{23}(j,\overline{\alpha_1,k}) = \sum_{\alpha_1}^{r_1}\sum_{\alpha_2}^{r_2} U_1(i,\alpha_1)U_2(j,\alpha_2)\tilde{G}_3(\alpha_2,\overline{\alpha_1,k}) = \\
&= \sum_{\alpha_1}^{r_1}\sum_{\alpha_2}^{r_2} U_1(i,\alpha_1)U_2(j,\alpha_2)\tilde{G}_3(k,\overline{\alpha_1,\alpha_2}) = \\
&= \sum_{\alpha_1}^{r_1}\sum_{\alpha_2}^{r_2}\sum_{\alpha_3}^{r_3} U_1(i,\alpha_1)U_2(j,\alpha_2)U_3(k,\alpha_3)\tilde{G}(\alpha_3,\overline{\alpha_1,\alpha_2}).
\end{aligned} \tag{3}$$

Таким образом, после финальных операций `reshape` и `transpose` мы получаем искомое разложение Таккера

$$\sum_{\alpha_1}^{r_1}\sum_{\alpha_2}^{r_2}\sum_{\alpha_3}^{r_3} U_1(i,\alpha_1)U_2(j,\alpha_2)U_3(k,\alpha_3)G(\alpha_1,\alpha_2,\alpha_3). \tag{4}$$

Отсюда следует, что необходимо построить скелетные разложения для следующих матриц (предполагаем, что их ранги не превосходят числа R, что позволяет нам иметь верхние оценки сложности для операции сингулярного разложения):

1. $A(i,\overline{j,k})$ – потребует $O(N^2 \cdot N^2) = O(N^4)$ действий.
2. $\tilde{G}_{23}(j,\overline{\alpha_1,k})$ – потребует $O(N^2 \cdot NR) = O(N^3R)$ действий.
3. $\tilde{G}_3(k,\overline{\alpha_1,\alpha_2})$ – потребует $O(\min(N^2R^2, NR^4))$ действий.

Полная сложность алгоритма в такой версии составляет $O(N^4 + N^3R + \min(N^2R^2, NR^4))$ вместо исходных $O(3N^4)$ действий для классической процедуры HOSVD.

Действительно замечательный факт о разложении Таккера заключается в следующем: использование усеченного SVD в этих алгоритмах позволяет получить приближенное разложение Таккера со всеми необходимыми теоремами о его точности. Практический факт о приближенном алгоритме HOSVD состоит в следующем:

$$||A - B_t||_F \leqslant \sqrt{d}||A - B_*||_F,$$

где B_t – результат либо HOSVD, либо st-HOSVD (который математически эквивалентен HOSVD), а B_* – наилучшее приближение того же фиксированного мультилинейного ранга.

В общем случае представление d-мерных данных в формате разложения Таккера записывается следующим образом:

$$A(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \ldots \sum_{\alpha_d=1}^{r_d} G(\alpha_1, \alpha_2, \ldots \alpha_d) U_1(i_1, \alpha_1) \ldots U_d(i_d, \alpha_d). \quad (5)$$

Легко заметить, что для хранения ядра Таккера $G(\alpha_1, \alpha_2, \ldots \alpha_d)$ требуются $r_1 \cdot r_2 \cdot \ldots r_d$ ячеек памяти. Пусть $r_i = R$, тогда для хранения ядра разложения Таккера потребуется R^d ячеек памяти. Данное свойство существенно снижает пользу от использования такого тензорного формата в случаях данных высокой размерности d.

Алгоритм TTSVD

В случае трехмерных массивов формат тензорного поезда соответствует следующему представлению массива:

$$A(i,j,k) = \sum_{\alpha_1}^{r_1} \sum_{\alpha_2}^{r_2} G_1(i, \alpha_1) G_2(\alpha_1, j, \alpha_2) G_3(\alpha_2, k). \quad (6)$$

В определенном смысле данное выражение можно трактовать в качестве “не полностью завершенного” разложения Таккера и требует $Nr_1r_2 + Nr_1 + Nr_2$ ячеек памяти, что не кажется оптимальным способом сжатия. Однако его свойства становятся важными для данных, имеющих более высокую размерность, потому что он не имеет явных экспоненциальных требований к памяти, в отличие от разложения Таккера.

Его также можно вычислить с помощью процедуры на основе SVD, известной как алгоритм TTSVD, который в случае трехмерных массивов выглядит следующим образом:

$$A(i,\overline{j,k}) = \sum_{\alpha_1=1}^{r_1} G_1(i,\alpha_1)G_{2,3}(\alpha_1,\overline{j,k}) =$$

$$\sum_{\alpha_1=1}^{r_1} G_1(i,\alpha_1)G_{2,3}(\overline{\alpha_1,j},k) = \tag{7}$$

$$\sum_{\alpha_1=1}^{r_1}\sum_{\alpha_2=1}^{r_2} G_1(i,\alpha_1)G_2(\alpha_1,j,\alpha_2)G_3(\alpha_2,k).$$

Этот алгоритм имеет очевидную сложность $O(N^4)$ действий. Минимальные TT-ранги соответствуют рангам уже несколько иных матриц развертки исходного тензора:

$$A_1 = A(i,\overline{j,k}) \quad \text{and} \quad A_2 = A(\overline{i,j},k).$$

Приближенное усечение TTSVD обладает такими же квазиоптимальными свойствами точности, как и усечение HOSVD:

$$||A - B_t||_F \leqslant \sqrt{d-1}||A - B_*||_F,$$

где B_t – результат TTSVD, а B_* – наилучшее приближение тех же фиксированных TT-рангов [22]. Окончательный d-мерный тензорный поезд соответствует записи

$$A(i_1,...,i_d) = \sum_{\alpha_1}^{r_1}\sum_{\alpha_2}^{r_2}...\sum_{\alpha_{d-1}}^{r_{d-1}} G_1(i_1,\alpha_1)G_2(\alpha_1,i_2,\alpha_2)...G_{d-1}(\alpha_{d-1},i_d). \tag{8}$$

Такое представление существует для любого d-мерного массива и известно в литературе не только как тензорный поезд, но и как состояние матричного произведения (matrix product state) [26], поскольку элемент тензора $A(i_1,i_2,\ldots,i_d)$ можно вычислить с помощью $(d-2)$ операций произведения матрицы на вектор.

Метод выбора рангов при сжатии видеоданных

Алгоритм выбора аналитического ранга для st-HOSVD и TTSVD построен следующим образом:

1. Выберем целевое качество видео для сжатого формата, а именно определим целевое PSNR. Этот показатель качества выбран из-за его соответствия норме Фробениуса:

$$PSNR = 10\log_{10}\left(\frac{(2^n-1)^2}{MSE}\right) = -10\log_{10}\left(\frac{||\bar{A}-\bar{A^*}||_F^2}{255^2 H\cdot W\cdot C\cdot T}\right), \tag{9}$$

где $\bar{A}$ - развернутый в матрицу тензор, представляющий исходное видео (или его часть/фрагмент), $\bar{A^*}$ - развернутый тензор, содержащий видео, декодированное из сжатого формата, n - количество бит для кодирования интенсивности пикселя на пиксель на канал ($n = 8$ для `RGB24`), H - высота, W - ширина, T - количество кадров, и C - количество цветовых каналов.

2. Вычислим соответствующую норму Фробениуса для невязки:

$$||\bar{A} - \bar{A^*}||_F = \sqrt{255^2 \cdot H \cdot W \cdot C \cdot T \cdot \exp\left(-\frac{PSNR}{10}\right)} \tag{10}$$

3. Далее необходимо выбрать ранги для каждого учесённого сингулярного разложения в алгоритмах st-HOSVD или TTSVD. Для этого применим SVD к развернутому тензору $\bar{B}$, представляющему сжатое видео, на одном из шагов используемых алгоритмов сжатия. Сингулярные числа $\sigma_1, \sigma_2, \ldots, \sigma_R$ вычисляются из SVD и сортируются в порядке убывания. Учитывая связь между нормой Фробениуса и сингулярными значениями:

$$||\bar{B}||_F = \sqrt{\sum_{i=1}^{R} \sigma_i^2}, \tag{11}$$

получаем оценку нормы ошибки

$$||\bar{B} - \bar{B^*}||_F = \sqrt{\sum_{i=r_{tr}+1}^{R} \sigma_i^2}. \tag{12}$$

Здесь $\bar{B^*}$ — наилучшая аппроксимация ранга r_{tr} для матрицы $\bar{B}$. Для получения целевого PSNR выбирается минимальный r_{tr}, обеспечивающий норму Фробениуса невязок ниже порога, пересчитанного из PSNR. Учитывая количество SVD d, применяемых в рамках st-HOSVD или TTSVD, устанавливаем пороги, равномерно распределяющие ошибки между операциями применения SVD. Это может вызвать осложнения по причине дискретных эффектов из-за малых размерностей некоторых мод тензора, что вызывает превышение целевого PSNR. Таким образом, применение адаптивных алгоритмов распределения ошибок может стать предметом дальнейших исследований. Подводя итог, в данной работе мы вычисляем ранги тензорных разложений следующим образом:

$$r_{tr} = \max_{r_{tr}} \left[\sqrt{\sum_{i=r_{tr}+1}^{R} \sigma_i^2} > \sqrt{\frac{1}{d}} \cdot ||\bar{A} - \bar{A^*}||_F \right] = \tag{13}$$

$$\max_{r_{tr}} \left[\sqrt{\sum_{i=r_{tr}+1}^{R} \sigma_i^2} > \sqrt{\frac{255^2}{d} \cdot H \cdot W \cdot C \cdot T \cdot \exp\left(-\frac{PSNR}{10}\right)} \right]$$

Этот приём применяется независимо к каждому цветовому каналу, если не используется матрицизация размерности. Для st-HOSVD это означает, что при применении tr-SVD по цветовой моде тензора использовалось $r_{tr} = 3$.

Результаты и обсуждение

Артефакты сжатия видео могут существенно повлиять на визуальное восприятие видео конечным пользователем. Они особенно

заметны при кодировании со средне-низким целевым PSNR что минимизирует размер/битрейт для сжатого видео. Было намерено выбрано очень низкое целевое значение PSNR, равное 25 дБ, для визуального выявления характерных артефактов, вносимых алгоритмами st-HOSVD и TTSVD (с матрицизацией или без неё). Наглядный пример представлен на рис. 2.

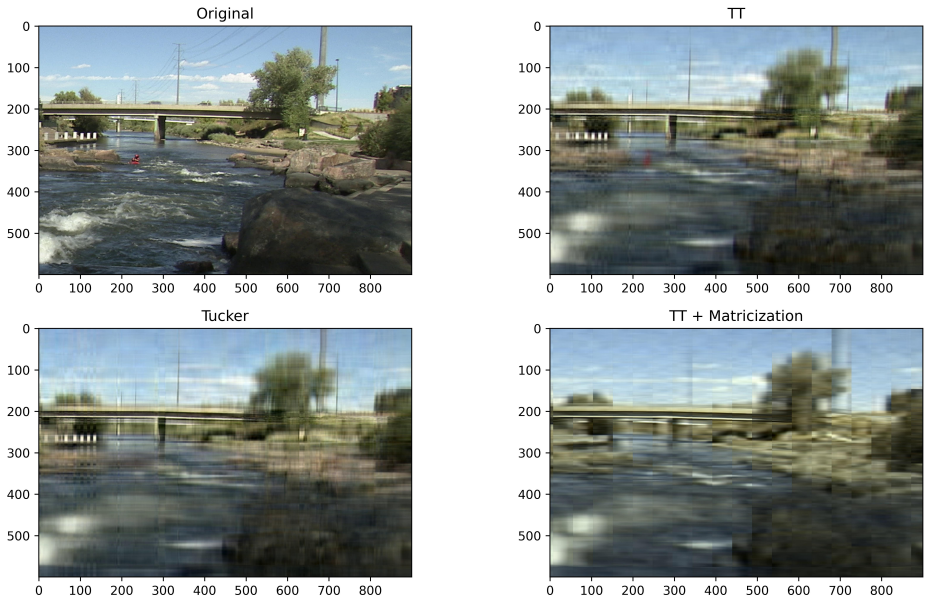

Рис. 2. Артефакты сжатия. Демонстрируются на 291 кадре видео *Red Kayak* и целевом PSNR 25 дБ для всех алгоритмов сжатия. Построчно слева-направо показаны кадры для исходного видео, видео, восстановленного из форматов TT, Tucker и TT (с предварительной матрицизацией).

Как для TTSVD, так и для st-HOSVD наблюдаются артефакты, имеющие структуру полос и линий (связанные с малым размером базиса по строкам и столбцам). Отдельные визуальные элементы на изображении, оказывают нелокальное влияние на весь столбец/строку (см. рис. 2, верхнее правое и нижнее левое изображения). Дополнительная матрицизация по пространственным измерениям вводит блочную структуру (см. рис. 2, нижнее правое изображение), тем самым уменьшая нелокальные артефакты сжатия. Блочная структура появляется после применения матрицизации к модам H-W. Условно говоря, каждый кадр разбивается на блоки, которые складываются путем добавления к тензору двух дополнительных мод. Выполняются еще два SVD, что обеспечивает разные базисные наборы строк и столбцов для каждого блока. Это уменьшает нелокальные эффекты, что, в свою очередь, улучшает визуальное восприятие сжатого видео.

В качестве следующего шага анализируется наличие малоранговых структур в видеороликах. Мотивация заключается в том, что тензорные форматы в этом случае особенно эффективны для сжатия данных

высокой размерности [11], [15], [27]. Этот аспект изучается для st-HOSVD и TTSVD с использованием анализа нормированных сингулярных чисел, получаемых на каждом шаге алгоритмов.

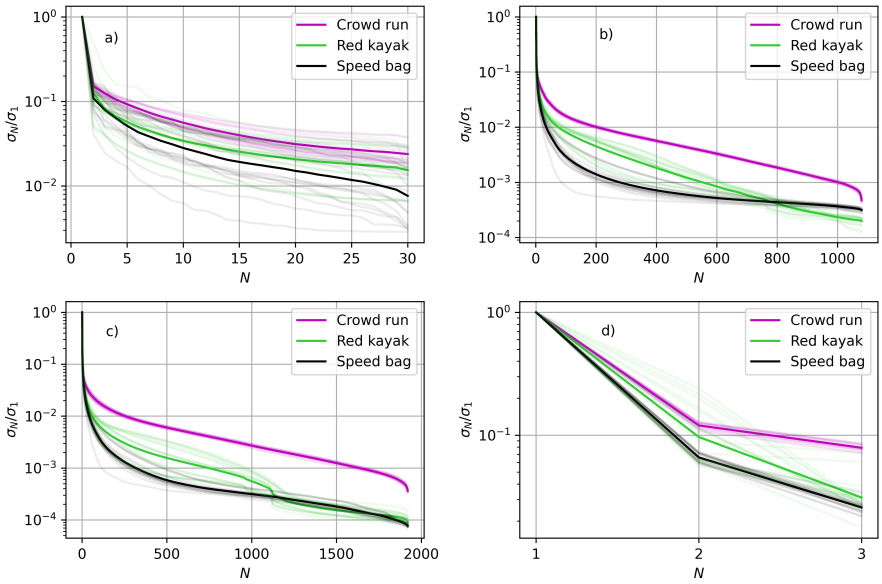

Рис. 3. Графики сингулярных чисел для формата Таккера (алгоритм st-HOSVD). Нормирование проведено по первому (наибольшему) сингулярному числу по каждой моде. Графики соответствуют SVD-разложениям видеофрагментов (30 кадров, 4D-тензор), развернутых по модам T, H, W и C соответственно. Полупрозрачными линиями показаны сингулярные числа для каждого фрагмента из 30 кадров, на которые делится видео. Яркие линии изображают соответсвуют сингулярным значениям, усредненным между всеми фрагментами. Пурпурный, зеленый и черный цвета используются для отображения результатов для видео *Crowd run*, *Red kayak* и *Speed bag* соответственно.

На рис. 3демонстрируются результаты применения st-HOSVD. Для каждого фрагмента сжатого видео наглядно видна различная временная и пространственная сложность. Из полученных результатов не наблюдается возможность сжатия по моде C, что, тем не менее, не распространяется на применение гиперспектральным данным. Для видео *Red kayak* с быстрой сменой сцены на графике виден перегиб для некоторых фрагментов по каналу W. Мы предполагаем, что это может быть связано с наличием статического фона в некоторых фрагментах, поскольку в этом случае для кодирования требуется меньше сингулярных значений в tr-SVD. Видео *Crowd run* имеет более высокую пространственно-временную сложность, что выражается в более медленном убывании сингулярных чисел вдоль соответствующих мод тензора. Очевидно, что в этом случае более низкий CR может быть достигнут с тем же целевым качеством, поскольку для выбранного целевого PSNR может быть отброшено меньшее количество

сингулярных чисел. Для всех исследованных видео сингулярные числа для временных и пространственных мод быстро убывают только на 1-2 порядка, а затем наблюдается относительно медленное их уменьшение при увеличении N. Следовательно, отсутствует ярко выраженная низкоразмерная структура, что вызывает необходимость проанализировать кривые RD, для уточнения компромисса, который может быть достигнут между количеством сохраняемых сингулярных чисел и потерями, которые возникают при сжатии видео.

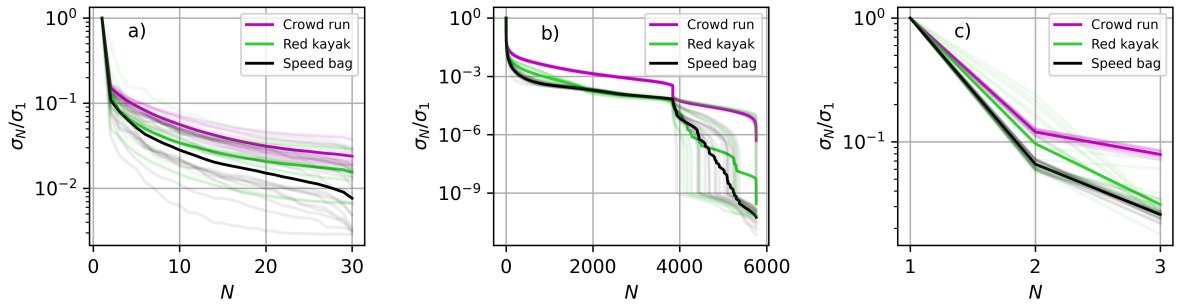

Рис. 4. Графики сингулярных чисел для TT-формата (алгоритм TTSVD). Нормирование проведено по первому (наибольшему) сингулярному числу по каждой моде. Графики соответствуют SVD-разложениям видеофрагментов (30 кадров, 4D-тензор), соответственно развернутых по 1-й, 2-й и 3-й модам в рамках алгоритма TTSVD. Полупрозрачными линиями показаны единичные значения для каждого фрагмента из 30 кадров, на которые делится видео.Яркие линии изображают соответсвуют сингулярным значениям, усредненным между всеми фрагментами. Пурпурный, зеленый и черный цвета относятся к результатам для видео *Crowd run*, *Red kayak* и *Speed bag* соответственно.

Аналогичные результаты наблюдаются для формата разложения тензорного поезда при применении алгоритма TTSVD. Они представлены на рис. 4. Этот алгоритм использует в данном случае только 3 tr-SVD для мод T, смешанной моды W-H и C. Так же не наблюдается возможность сжатия по моде C без значительных потерь. Малый размер вдоль C-моды тензора затрудняет обеспечение целевого PSNR из-за дискретных эффектов: отбрасывание даже одного сингулярного числа значительно снижает PSNR. Оно смещается далеко от предварительно определенного порогового значения. Высокая пространственно-временная сложность видео *Crowd run* выражается в более медленном убывании соответствующих сингулярных чисел. Различная пространственная сложность фрагментов видео, а также наличие смены сцены для *Speed bag* и *Red kayak* ограничивают анализ, основанный на сингулярных числах, усредненных между фрагментами. Действительно, резкое уменьшение сингулярных чисел возникает при разных N для каждого

фрагмента. Отсюда можно дополнительно предположить, что сжатие может быть улучшено с помощью использования алгоритмов обнаружения смены сцены.

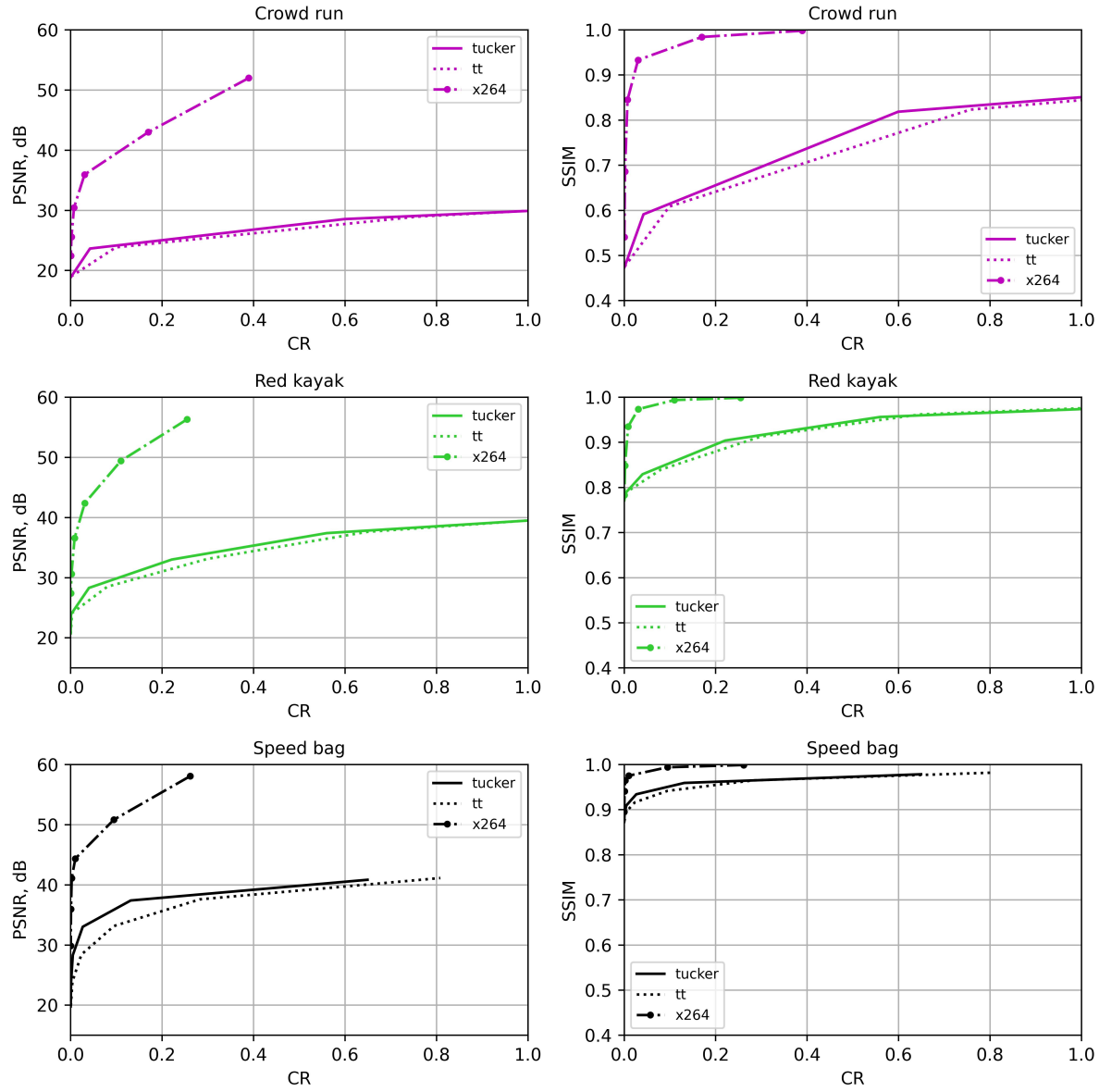

Рис. 5. Графики RD. Левый столбец: приведены графики PSNR-CR, правый столбец: приведены графики SSIM-CR. В 1-й, 2-й и 3-й строках показываны соответсвенно результаты для видео *Crowd run*, *Red kayak* и *Speed bag*. Пунктирная линия: st-HOSVD, сплошная линия: TTSVD и штрихпунктирная линия: `ffmpeg` (x264, medium пресет, CRF). Все количественные метрики качества были получены при помощи `ffmpeg`.

Представленные выше результаты говорят о необходимости дополнительного анализа графиков RD (см. рис. 5). Алгоритм st-HOSVD показывает лучшие результаты, чем TTSVD. Однако реализация стандарта `h264` (`x264`) с открытым исходным кодом по-прежнему

демонстрирует более высокие результаты. Отсюда можно прийти к выводу, что тензорные форматы не являются самодостаточными для целей сжатия, но они могут найти применение в рамках методик обработки и восстановления видео [27] или в качестве шага предварительной подготовки и сжатия данных для нейронных сетей. В качестве заключительной части данной работы, применена матрицизация к видео в совокупности с TTSVD. Результаты сжатия после матрицизации показаны на рис. 6. Матрицизация для формата TT улучшает качество сжатия, немного смещая кривую RD вверх. При этом, она практически совпадает с RD для st-HOSVD. Однако, наблюдаемого улучшения все же недостаточно для достижения результатов показанных `x264`.

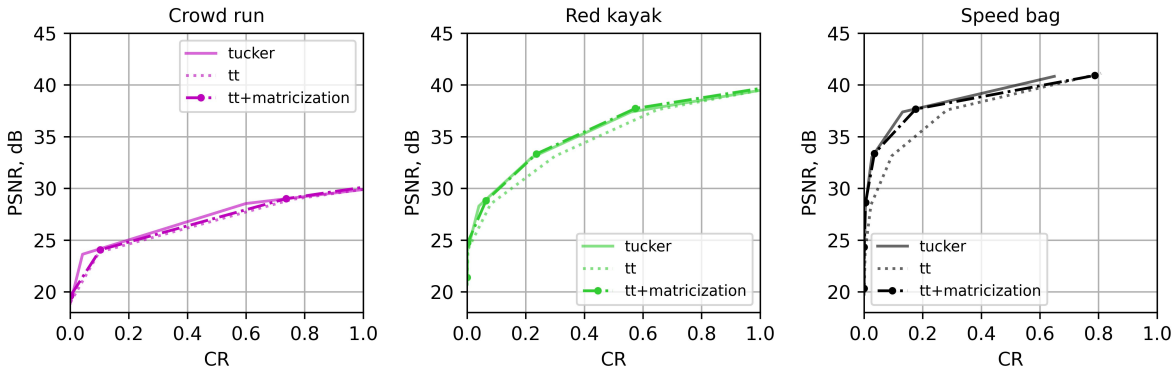

Рис. 6. Графики RD. Графики PSNR-CR показаны слева-направо соответственно для видео *Crowd run*, *Red kayak* и *Speed bag*. Пунктирная линия: TTSVD, сплошная линия: st-HOSVD, штрихпунктирная линия: TTSVD с матрицизированным видео (видеофрагмент из 30 кадров с размерностью (30, 1080, 1920, 3) и матрицизированный в тензор формы (30, 30, 36, 40, 48, 3)). Все количественные метрики качества были получены при помощи `ffmpeg`.

Заключение

Было проведено тематическое исследование возможностей сжатия видеоданных с использованием таких тензорных форматов как разложение тензорного поезда и разложение Таккера. Были применены классические алгоритмы на основе SVD для сжатия видео из открытой коллекции с различной пространственно-временной сложностью, а также для изучения применимости дополнительной матрицизации видеоданных. Несмотря на то, что на текущем этапе исследований полученные коэффициенты сжатия не достигают значений предоставляемых современными реализациями стандартов для сжатия видео, мы считаем, что подходы изучаемые в данной работе могут иметь практические приложения из-за гораздо более выраженной структуры разложений Таккера и тенорного поезда. Кроме того, матричная структура

применяемых тензорных разложений может оказаться полезной при использовании нейронных сетей для обработки видео, поскольку она позволяет уменьшить размерность входных данных без значительных потерь для процессов обучения и инференса.

Остается большое количество возможностей улучшения разработанного подхода, в том числе, за счет (i) адаптивного выбора неоднородных порогов при применении SVD в алгоритмах st-HOSVD и TTSVD (ii) внедрения алгоритмов обнаружения смены сцены для разделения видео на фрагменты и (iii) адаптации разработанного фреймворка для работы непосредственно в цветовом пространстве `yuv`. Мы надеемся, что это, в свою очередь, также облегчит использование тензорных форматов для сжатия видео с потерями в качестве самостоятельного подхода и/или в рамках новых методов, основанных на глубоком обучении.

Литература

1. *Antsiferova, A., Lavrushkin, S., Smirnov, M., Gushchin, A., Vatolin, D., Kulikov, D.* Video compression dataset and benchmark of learning-based video-quality metrics, arXiv preprint arXiv:2211.12109, 2022
2. *Huynh-Thu, Q., Ghanbari, M.* Scope of validity of PSNR in image/video quality assessment, Electronics letters, 44, 13 800–801, 2008
3. *Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P.* Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, IEEE transactions on image processing, 13(4), 600–612, 2004
4. *Li, Z., Bampis, C., Novak, J., Aaron, A., Swanson, K., Moorthy, A., Cock, J. D.* VMAF: The journey continues, Netflix Technology Blog. 25, (1), 2018
5. *Sullivan, G. J., Topiwala, P. N., Luthra, A.* The H. 264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions, Applications of Digital Image Processing XXVII, 5558, 2004, 454–474
6. *Sullivan, G. J., Ohm, J. R., Han, W.J., Wiegand, T.* Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on circuits and systems for video technology, 22 12, 1649-1668, 201
7. *Bross, B., Wang, Y. K., Ye, Y., Liu, S., Chen, J., Sullivan, G.J., Ohm, J.R.* Overview of the versatile video coding (VVC) standard and its applications, IEEE Transactions on Circuits and Systems for Video Technology, 31, (10), 3726–3764, 2021
8. *Liu, H., Chen, T., Lu, M., Shen, Q., Ma, Z.* Neural video compression using spatio-temporal priors, arXiv:1902.07383, 2019

9. *Matveev, S.A., Oseledets, I.V., Ponomarev, E.S., Chertkov, A.V.* Overview of visualization methods for artificial neural networks, Computational Mathematics and Mathematical Physics, 61(5), 887 – 899, 2022

10. *Pizarroso, J., Portela, J., Muñoz, A.* NeuralSens: sensitivity analysis of neural networks, arXiv preprint arXiv:2002.11423, 2020

11. *Sultonov, A., Matveev, S.A., Budzinskiy, S.* Low-rank nonnegative tensor approximation via alternating projections and sketching, Computational and Applied Mathematics, 42(2), 68, 2023

12. *Yuan, Q., Zhang, L., Shen, H.* Hyperspectral image denoising employing a spectral–spatial adaptive total variation model, IEEE Transactions on Geoscience and Remote Sensing, 50(10), 3660 – 3677, 2012

13. *Jia, H., Guo, S., Li, Z., Chen, X.A., Han, Z., Tang, Y.* Low-Rank Tensor Tucker Decomposition for Hyperspectral Images Super-Resolution, In Intelligent Robotics and Applications: 15th International Conference, ICIRA 2022, Proceedings, Part II, 502 – 512, 2022

14. *Sidiropoulos, N. D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., Faloutsos, C.* Tensor decomposition for signal processing and machine learning, IEEE Transactions on Signal Processing, 65(13), 3551 – 3582, 2017

15. *Usvyatsov, M., Ballester-Rippoll, R., Bashaeva, L., Gushchin, A., Schindler, K., Ferrer, G., Oseledets, I.* T4DT: Tensorizing Time for Learning Temporal 3D Visual Data, arXiv preprint arXiv:2208.01421, 2

16. *Kolda T.G, Bader B.W* Tensor decompositions and applications. SIAM review. SIAM Journal on Scientific Computing 51(3), 455–500 (2009)

17. *Tucker, L.R.* The extension of factor analysis to three-dimensional matrices, Contributions to mathematical psychology, 110119, 1964

18. *Tucker L.R.* Some mathematical notes on three-mode factor analysis, Psychometrika, 31(3), 279–311, 1966

19. *Oseledets I.V* Tensor-train decomposition. SIAM Journal on Scientific Computing 33(5), 2295–2317 (2011)

20. *Oseledets I.V.* Approximation of matrices with logarithmic number of parameters. Doklady Mathematics 80(2), 653–654 (2009)

21. *Badeau R., Boyer R.* Fast multilinear singular value decomposition for structured tensors. SIAM Journal on Matrix Analysis and Applications 30(3), 1008–1021 (2008)

22. *Oseledets I. V., Tyrtyshnikov E.E.* Breaking the curse of dimensionality, or how to use SVD in many dimensions. SIAM Journal on Scientific Computing 31(5), 3744–3759 (2009)

23. *Ahmadi-Asl S., Abukhovich S., Asante-Mensah M. G., Cichocki A., Phan A. H, Tanaka T., Oseledets I. V.* Randomized algorithms for computation of Tucker decomposition and higher order SVD (HOSVD). IEEE Access 9, 28684–28706 (2021)

24. *Dolgov S. V., Savostyanov D.* Parallel cross interpolation for high-precision calculation of high-dimensional integrals, Computer Physics Communications, 246, 106869, 2020

25. *Zvezdakova A.V., Kulikov D.L., Zvezdakov S.V., Vatolin D.S.* BSQ-rate: a new approach for video-codec performance comparison and drawbacks of current solutions, Programming and computer software, 46, 183–194 2020

26. *Zhang C., Jeckelmann E., White S. R.* Density matrix approach to local Hilbert space reduction. Physical review letters 80(12), 2661 (1998)

27. *Ahmadi-Asl S., Asante-Mensah M. G., Cichocki A., Phan A. H., Oseledets I., Wang J.* Fast Cross Tensor Approximation for Image and Video Completion, Signal Processing, 109121, 20