TWO FACTORS UPDATE FOR CANONICAL POLYADIC DECOMPOSITION

Anastasia Sozykina
Skoltech
Moscow, Russia
anastasia.sozykina@skoltech.ru

Valentin Leplat
Innopolis University
Innopolis, Russia
v.leplat@innopolis.ru

Igor Vorona
Skoltech
Moscow, Russia
igor.vorona@skoltech.ru

Salman Ahmadi-Asl
Skoltech
Moscow, Russia
s.asl@skoltech.ru

Anh-Huy Phan
Skoltech
Moscow, Russia
a.phan@skoltech.ru

ABSTRACT

The current alternating optimization algorithms for canonical polyadic (CP) tensor decomposition face various challenges such as redundant update steps and the robustness of optimization when dealing with linearly dependent factor matrices. These concerns often result in prolonged iterations without substantial progress in the decomposition process. In response to these challenges, our paper introduces two novel optimization algorithms for computing CP decompositions relying on the update of two factor matrices simultaneously. We assess the performance of our algorithm by conducting thorough numerical experiments involving both synthetic and real-world data tensors. Through these experiments, we demonstrate the efficiency and benefits of our proposed approach.

1 INTRODUCTION

Tensors are mathematical objects that extend the concepts of scalars, vectors, and matrices to an arbitrary number of dimensions. Scalars are considered as zero-dimensional tensors and vectors as one-dimensional tensors, whereas matrices are viewed as two-dimensional tensors. Moving from two-dimensional matrices, a more complex structure is developed in the form of a tensor with dimensions depth, width, and height.

Among the various tensor decomposition models, the Canonical Polyadic Decomposition (CPD), also referred to as the CANDECOMP/PARAFAC (CP) model Hitchcock (1927), stands out as the most popular and widely used method. Given a tensor $\mathcal{Y}$ with dimensions $I_1 \times I_2 \times \cdots \times I_N$, CPD aims to find the optimal rank-R tensor approximation of $\mathcal{Y}$ (see illustrattion in Figure 1)

$$\mathcal{Y} \approx \hat{\mathcal{Y}} = \sum_{r=1}^{R} \boldsymbol{a}_{1r} \circ \boldsymbol{a}_{2r} \circ \cdots \circ \boldsymbol{a}_{Nr} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!] , \tag{1}$$

where "$\circ$" represents the outer product, and $\mathbf{A}_n = [\boldsymbol{a}_{n1}, \ldots, \boldsymbol{a}_{nR}]$ denote the factor matrices of the decomposition.

The CPD is a highly relevant tensor decomposition model that retains its core multilinear structure while achieving dimensionality reduction. It is widely used in areas where intact structural information is required, such as data mining, signal processing, neuroimaging, and chemometrics. CPD has shown to be a powerful tool for many applications in various fields, including separating signals in wireless communication systems, performing independent component analysis, estimating temporal and spectral patterns in EEG signals, blind identification, and even compressing Convolutional Neural Networks (CNN) to cite a few Cichocki et al. (2015); Lebedev et al. (2014); Zhang et al. (2016).

Furthermore, the CPD model offers the flexibility to incorporate additional constraints to enhance its utility, performance and come up with more interpretable decompositions. For instance, CPD can be extended to nonnegative tensor factorization Cichocki et al. (2009), which has emerged as an important tool in interpreting physical data where non-negativity constraints are natural. In the last two decades, additional constrained and regularized variants of CPD have been successfully used, we can mention the smooth CPD Yokota et al. (2016), orthogonal CPD Sørensen et al. (2012), graph regularized CPD Maki et al. (2018), discriminant CPD Frølich et al. (2018), and CPD with bounded sensitivity or intensity Phan et al. (2019).

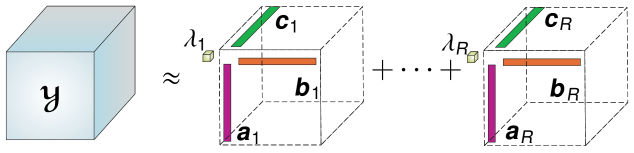


Figure 1: Illustration of the CPD.

Moreover, CPD can incorporate predefined design matrices as in the CANDELINC Carroll et al. (1970); Bro & Andersson (1998). Another variation of the model has been suggested specifically for cases where the factor matrices can be represented on subspaces spanned by certain dependence matrices. These dependence matrices typically consist of zeros and ones. This approach gives rise to two distinct methodologies: Parallel Profiles with Linear Dependencies (PARALIND), which addresses the rank-overlap problem as proposed by Bro et al. (2009), and Constrained PARAFAC (CONFAC), which is developed for the blind identification of underdetermined mixtures de Almeida et al. (2012). For both methodologies, the model is of the form

$$\mathcal{Y} \approx [\![\mathbf{A}^{(1)}\mathbf{Q}_1, \mathbf{A}^{(2)}\mathbf{Q}_2, \mathbf{A}^{(3)}\mathbf{Q}_3]\!]. \tag{2}$$

A more generalized extension of CPD, dubbed as the low-rank constrained CPD (LrCPD) Phan et al. (2021) relies on the hybrid Tucker-CPD decomposition model:

$$\mathcal{Y} \approx [\![\mathbf{UA}, \mathbf{VB}, \mathbf{WC}]\!], \tag{3}$$

where $\mathbf{U}^T\mathbf{U} = \mathbf{I}_{R_1}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_{R_2}$ and $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{R_3}$. This model showed promising results when the tensor rank of a tensor exceeds its dimensions.

As a take-home message, CPD has emerged as the go-to tensor decomposition model, showcasing its distinct and powerful capabilities across multiple disciplines. With its broad applicability and persisting relevance over time, CPD is poised to remain as a key tensor decomposition model across the diverse fields of data sciences. These key observations motivated us to propose new and efficient methods to compute a CPD.

1.1 Existing algorithms for computing a CPD

There are several algorithms to compute the CPD of a tensor such as Alternating Least Squares (ALS) algorithm Harshman (1970); Comon et al. (2009), the Weighted Krylov-Levenberg-Marquardt algorithm Tichavský et al. (2020), the non-linear least squares (NLS) algorithm Sorber et al. (2013), the generalized Schur decomposition Evert et al. (2022), the accelerated proximal gradient with momentum Nazih et al. (2021), and the accelerated stochastic gradient descent Siaminou & Liavas (2021). Efficient implementations of some CPD algorithms in Python can be found in Diniz (2019). The ALS is the simplest type of such algorithms and has been widely used, however it suffers from two main drawbacks

- The first issue with ALS-type algorithm is high redundancy in computing the update rules, which is related to computing the "matricized tensor times Khatri-Rao product".
- The second one occurs when several factor matrices have collinear loading components. In such scenarios, the condition numbers of linear systems in ALS steps are high, and the optimization process becomes inefficient.

Based on the above-mentioned observations, we propose innovative algorithms to address these challenges. In a nutshell, our approach involves updating two factor matrices instead of just one, which sets it apart from classical ALS-type algorithms. The key aspect here is the unfolding of the CP model along two arbitrary modes.

This specific unfolding results in a lower-order CP model with one factor matrix with the so-called "Khatri-Rao structure". For example, merging mode-1 and 2 of an order-4 tensor $\mathcal{Y} = [\![\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4]\!]$ of size $I \times J \times K \times L$ gives an order-3 CP tensor $\mathcal{Y}_{(1,2)} = [\![\mathbf{B}, \mathbf{A}_3, \mathbf{A}_4]\!]$, where $\mathbf{B} = \mathbf{A}_2 \odot \mathbf{A}_1$ after which the factor matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ can be retrieved through the best rank-1 approximation of the reshaped form of the columns of the matrix $\mathbf{B}$ to matrices of size $I \times J$, i.e., $\boldsymbol{b}_r = vec(\boldsymbol{a}_{1r}\boldsymbol{a}_{2r}^T)$ where $\boldsymbol{b}_r, \boldsymbol{a}_{1r}, \boldsymbol{a}_{2r}$ are the r-th columns of the matrices $\mathbf{B}$, $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively Phan et al. (2013). Due to the Khatri-Rao structure, reshaping column of $\boldsymbol{b}_r$ to matrices of size $I \times J$ gives a rank-1 matrix, or $\boldsymbol{b}_r = vec(\boldsymbol{a}_{1r}\boldsymbol{a}_{2r}^T)$. The FCP algorithm in Phan et al. (2013) exploits this Khatri-Rao structure in estimation of higher order CPD from lower order decomposition. In this paper, we follow a different approach by using this property to derive new update rules able to jointly update two factor matrices at once.

1.2 Contributions and Outline

In this paper, we address the main drawbacks of existing methods for computing a CPD of an input tensor. The key advantages and major contributions of our paper are the following:

- We propose a novel algorithm which updates two factor matrices instead of one as in the ALS steps. This property reduces redundancy of computing update rules and makes optimization more efficient by slightly avoiding risk of high condition numbers of factor matrices.
- We show empirically that our method achieves significantly better results than ALS and NLS methods if the penalty parameter is well chosen.
- We open the door of a new class of optimization methods for tensor decomposition able to update a larger substructure of a tensor decomposition at once. These methods are closer to the general optimization methods in their convergence robustness while simultaneously enjoying the efficiency of alternating methods.

The paper is organized as follows. Section 2 presents the key matrix decomposition model obtained after the unfolding along two arbitrary modes, the so-called "Khatri-Rao structured" linear regression model. Section 3 contains the proposed algorithms and the suggested implementation strategy. Section 4 presents the numerical experiments on a series of synthetic and real-life datasets. Finally, Section 5 summarizes our conclusions and perspectives.

2 Linear Regression with Khatri-Rao structured matrix

We start by considering the constrained linear regression problem as defined below

$$\min_{\mathbf{X}} \; f(\mathbf{X}) = \frac{1}{2}\,\|\mathbf{Y} - \mathbf{\Phi}\,\mathbf{X}^T\|_F^2 + \frac{\mu}{2}\|\mathbf{X}\|_F^2, \quad \text{s.t. } \mathbf{X} = \mathbf{V} \odot \mathbf{U}, \tag{4}$$

where $\mathbf{Y}$ is a data matrix of size $K \times (IJ)$, $\mathbf{\Phi}$ of size $K \times R$. The regressor, $\mathbf{X}$, is in the form of Khatri-Rao product of two matrices, $\mathbf{V}$ of size $J \times R$ and $\mathbf{U}$ of size $I \times R$.

In order to solve the above constrained optimization problem, we introduce the indicator function, $i_D(.)$, associated to the set $\mathcal{D} = \{\mathbf{X} | \mathbf{X} = \mathbf{V} \odot \mathbf{U}\}$, that is the set of Khatri-Rao structured matrices; this indicator function is $i_D(\mathbf{X}) = 0$ if $\mathbf{X} \in \mathcal{D}$, otherwise ∞.

By introducing an additional variable, $\mathbf{Z}$, one can equivalently reformulate Problem (4) as follows

$$\min \quad f(\mathbf{Z}) + i_D(\mathbf{X})\,, \quad \text{s.t.} \quad \mathbf{X} = \mathbf{Z}, \tag{5}$$

Problems given in (5) are well-suited for the use of the Alternating Direction Method of Multipliers (ADMM) framework. We first build the augmented Lagrangian function associated to Problem (5) as follows

$$\mathcal{L}_\gamma(\mathbf{X}, \mathbf{Z}, \mathbf{T}) = f(\mathbf{Z}) + i_D(\mathbf{X}) + \frac{1}{2\gamma}\left(\|\mathbf{Z} - \mathbf{X} - \mathbf{T}\|_F^2 - \|\mathbf{T}\|_F^2\right), \tag{6}$$

where $\gamma > 0$ is a positive scalar and $\mathbf{T}$ denotes the scaled dual variables associated to matrix equality constraints in (5).

Given the current iterates $(\mathbf{Z}^{(k)}, \mathbf{X}^{(k)}, \mathbf{T}^{(k)})$ with k denoting the iteration counter, the updates of the primal variables $\mathbf{Z}$, $\mathbf{X}$, and the dual variable $\mathbf{T}$ consist of the following iterations

$$\mathbf{Z}^{(k+1)} = \arg\min_{\mathbf{Z}} \; f(\mathbf{Z}) + \frac{1}{2\gamma} \|\mathbf{Z} - \mathbf{X}^{(k)} - \mathbf{T}^{(k)}\|_F^2 \,, \tag{7}$$

$$\begin{aligned}\mathbf{X}^{(k+1)} &= \arg\min_{\mathbf{X}} \; i_D(\mathbf{X}) + \frac{1}{2\gamma} \|\mathbf{Z}^{(k+1)} - \mathbf{T}^{(k)} - \mathbf{X}\|_F^2 \\ &= \Pi_{\mathcal{D}}(\mathbf{Z}^{(k+1)} - \mathbf{T}^{(k)}) \,, \end{aligned}\tag{8}$$

$$\mathbf{T}^{(k+1)} = \mathbf{T}^{(k)} + \mathbf{X}^{(k+1)} - \mathbf{Z}^{(k+1)} \,. \tag{9}$$

The update for $\mathbf{X}$ in (8) boils down to computing the orthogonal projection of $(\mathbf{Z}^{(k+1)} - \mathbf{T}^{(k)})$ onto the set $\mathcal{D}$, which is detailed in Section 2.2.

2.1 Update of $\mathbf{Z}$

Solving the optimization problem in (7) amounts to minimize a quadratic function without constraints, the solution can be computed in closed-form expression as follows

$$\begin{aligned}\mathbf{Z}^{(k+1)} &= \arg\min_{\mathbf{Z}} \; \frac{1}{2} \|\mathbf{Y} - \mathbf{\Phi}\mathbf{Z}^T\|_F^2 + \frac{\mu}{2}\|\mathbf{Z}\|_F^2 + \frac{1}{2\gamma} \|\mathbf{Z} - \mathbf{X}^{(k)} - \mathbf{T}^{(k)}\|_F^2 \\ &= (\mathbf{Y}^T\mathbf{\Phi} + \frac{1}{\gamma}(\mathbf{X}^{(k)} + \mathbf{T}^{(k)}))(\mathbf{\Phi}^T\mathbf{\Phi} + (\mu + \frac{1}{\gamma})\mathbf{I})^{-1}. \end{aligned}\tag{10}$$

2.2 Update of $\mathbf{X}$

We start by reshaping the vectors $\boldsymbol{z}_r^{(k+1)} - \boldsymbol{t}_r^{(k)}$ to matrices $\mathbf{H}_r$ of size $I \times J$, where $r = 1, 2, \ldots, R$. From (8) and by definition of the Khatri-Rao product each column of the matrix, $\mathbf{X}^{(k+1)}$, corresponds to vectorization of the best rank-1 approximation of the matrices $\mathbf{H}_r \approx \boldsymbol{u}_r\boldsymbol{v}_r^T$. This approximation has a unique optimal solution which can be computed in closed-form by using the truncated SVD according to the Eckart–Young–Mirsky theorem. Algorithm 1 summarizes our approach for solving Problem (4) with the sequential updates of $\mathbf{Z}$, $\mathbf{X}$ and $\mathbf{T}$ in closed-form given in Equations (9), (10) and (8).

Algorithm 1: Linear Regression with Khatri-Rao structured Regressor

```
Input: Data matrix $\mathbf{Y}$: $K \times (IJ)$, $\mathbf{\Phi}$ of size $K \times R$, an initialization for $(\mathbf{X}, \mathbf{Z})$, a maximum
       number of iterations $k_{\max}$, and a threshold $\epsilon$
Output: $\mathbf{X}$ is Khatri-Rao product such that it minimizes $\|\mathbf{Y} - \mathbf{\Phi}\mathbf{X}^T\|_F^2$
begin
    Initiate $\mathbf{T} = \mathbf{X} = 0$
    Precompute $\mathbf{W} = \mathbf{Y}^T\mathbf{\Phi}$, $\mathbf{Q} = \mathbf{\Phi}^T\mathbf{\Phi}$, $\tilde{\mathbf{Q}} = (\mathbf{Q} + (\mu + \frac{1}{\gamma})\mathbf{I})^{-1}$
    $k \leftarrow 1$
    while $k \leq k_{max}$ and $\frac{\|\mathbf{X}-\mathbf{Z}\|_F}{\|\mathbf{Z}\|_F} > \epsilon$ do
        $\mathbf{Z} \leftarrow (\mathbf{W} + \frac{1}{\gamma}(\mathbf{X} + \mathbf{T}))\tilde{\mathbf{Q}}$                // Update $Z$
        for $r = 1, \ldots, R$ do                            // Update each column of $X$
            $\mathbf{H}_r \leftarrow \texttt{reshape}(\mathtt{z_r} - \mathtt{t_r}, [\mathtt{I} \times \mathtt{J}])$
            $\mathbf{X}_r \leftarrow \mathrm{vec}\big(\boldsymbol{u}_r s_r \boldsymbol{v}_r^T\big)$ where $[\boldsymbol{u}_r, s_r, \boldsymbol{v}_r] = \texttt{svds}(\mathbf{H}_r, 1)$
        end for
        $\mathbf{T} \leftarrow \mathbf{T} + \mathbf{X} - \mathbf{Z}$                          // Dual ascent step updates $T$
        $k \leftarrow k + 1$
    end while
end
```

2.3 Adaptive Penalty Parameter Update

In most experiments, the damping regularization parameter, μ, can be set to zero or regularly decreases as the strategy used in the Levenberg-Marquardt algorithm. The penalty parameter, γ, can be set as the largest singular value of $\mathbf{\Phi}$, and fixed during the estimation. A small value of γ will make the algorithm converge slowly, while a large number may cause a divergence in the optimization process. Upper bound of $\gamma = 1/\beta$ is derived in (21). In addition, when the approximation error reaches a certain level, e.g., 10^{-2}, we can adjust γ following the spectral method Xu et al. (2017) or the following simple strategy Parikh & Boyd (2014):

$$\gamma^{(k+1)} = \begin{cases} \tau\gamma^{(k)}\,, & \text{if} \quad r > \eta s \\ \tau^{-1}\gamma^{(k)}, & \text{if} \quad s > \eta r \\ \gamma^{(k)} \end{cases} \tag{11}$$

where $r = \|\mathbf{Z}^{(k+1)} - \mathbf{X}^{(k+1)}\|_F$ denotes the relative primal residue, and $s = \|\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)}\|_F / \|\mathbf{T}^{(k+1)}\|_F$ corresponds to the dual residue, $\eta = 5$ or 10, $\tau = 1.5$ or 2. We can reset the parameter, e.g., every 20 iterations.

2.4 Complexity

Here we discuss the computational complexity of Algorithm 1. To begin with, the update for matrix $\mathbf{Z}$ in (10) requires the computation of the matrix product $\mathbf{Y}^T\mathbf{\Phi}$, which has a computational cost of $\mathcal{O}(IJKR)$. This cost can be significant when dealing with large data sizes. Furthermore, calculating the correlation matrix, $\mathbf{\Phi}^T\mathbf{\Phi}$, requires a computational cost of $\mathcal{O}(IJR^2)$, and $\mathcal{O}(R^3)$ for its inverse. Fortunately, we only need to compute these terms once. If the algorithm implements an adaptive update strategy for parameters γ (and μ), the changes will only affect the eigenvalues of $\mathbf{\Phi}^T\mathbf{\Phi}$ and not its eigenvectors. Later in the discussion, it will be shown that the computation of matrices $\mathbf{\Phi}^T\mathbf{\Phi}$ for the CPD model only incurs a cost of $\mathcal{O}(R^2(I+J))$. Finally, the update for $\mathbf{X}_r$ as the best rank-1 approximation of $\mathbf{H}_r$ can be obtained efficiently by using the power method in parallel for each r.

2.5 Convergence guarantees

The convergence of Algorithm 1 is presented in Theorem 1 and more details can be found in Appendix A.

Theorem 1. *For a sufficiently large $\beta = 1/\gamma$, the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{T}^{(k)})$ generated by Algorithm 1 applied to Problem (5) converges globally, that is regardless of where the initial point is, to the unique limit point $(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{T}^{(*)})$, which is a stationary point of the augmented Lagrangian function, $\mathcal{L}_\gamma$, and $\mathbf{X}^{(*)}$ is a stationary point of Problem (5).*

3 A Novel algorithm for computing the CPD

The computation of the CPD of a tensor $\boldsymbol{\mathcal{Y}}$ of order-N is usually achieved by minimizing the Frobenius norm of the error tensor

$$\min_{\{\mathbf{A}_n\}} \|\boldsymbol{\mathcal{Y}} - [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]\|_F^2\,. \tag{12}$$

Classical algorithms, such as ALS, sequentially update each factor of the decomposition at a time while keeping the other factors fixed. Each of these updates involves the unfolding of the model along one mode of the tensor, say the n-th mode, and rewriting the optimization problem as follows

$$\min_{\mathbf{A}_n} \|\mathbf{Y}_{(n)} - \mathbf{A}_n\mathbf{\Psi}_n^T\|_F^2, \tag{13}$$

where $\mathbf{\Psi}_n = \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1$. The ALS update for $\mathbf{A}_n$ is given by $\mathbf{A}_n = \mathbf{Y}_{(n)}\mathbf{\Psi}_n\,(\mathbf{\Psi}_n^T\mathbf{\Psi}_n)^{-1}$.

Several key insights can be made:

Remark 1. *(High redundancy) The above update is simple, but each update requires to compute the product $\mathbf{Y}_{(n)}\mathbf{\Psi}_n$, which is the most expensive step in the ALS algorithm.*

We give an example for order-3 CPD. In order to update $\mathbf{A}_1$, the ALS computes $\mathbf{Y}_{(1)}(\mathbf{A}_3 \odot \mathbf{A}_2)$, then computes $\mathbf{Y}_{(2)}(\mathbf{A}_3 \odot \mathbf{A}_1)$ for the update of $\mathbf{A}_2$.

The product between the tensor $\mathcal{Y}$ and $\mathbf{A}_3$ is common in the two updates. For higher-order CPD, the product between $\mathcal{Y}$ and Khatri-Rao products of all but two matrices $\mathbf{A}_n$ and $\mathbf{A}_{n+1}$ are shared for the two updates of $\mathbf{A}_n$ and $\mathbf{A}_{n+1}$.

Remark 2. *(Degeneracy and inaccurate update) Since $\mathbf{\Psi}_n^T\mathbf{\Psi}_n = (\mathbf{A}_N^T\mathbf{A}_N)\circledast\cdots\circledast(\mathbf{A}_{n+1}^T\mathbf{A}_{n+1})\circledast(\mathbf{A}_{n-1}^T\mathbf{A}_{n-1})\circledast\cdots\circledast(\mathbf{A}_1^T\mathbf{A}_1)$ (where "$\circledast$" denotes the Hadamard product), when the factor matrices consist of highly collinear loading components, the correlation matrix $\mathbf{\Psi}_n^T\mathbf{\Psi}_n$ becomes poorly conditioned, and the computation of its inverse is likely to be inaccurate.*

Motivated by these observations, we derive in the following a new algorithm which can efficiently address the redundancy issue by updating two factor matrices at a time. To achieve this, let us rewrite the optimization problem by unfolding the tensor along two arbitrary modes n and m, $n < m$:

$$\min_{\mathbf{A}_m,\mathbf{A}_n} \|\mathbf{Y}_{(n,m)}^T - \mathbf{\Psi}_{n,m}(\mathbf{A}_m \odot \mathbf{A}_n)^T\|_F^2, \tag{14}$$

where $\mathbf{\Psi}_{n,m} = \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{m+1} \odot \mathbf{A}_{m-1} \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1$ is the Khatri-Rao product of all but two matrices $\mathbf{A}_n$ and $\mathbf{A}_m$. One can easily see that the minimization problem in (14) boils down to the regression problem with Khatri-Rao structured matrix as in (4).

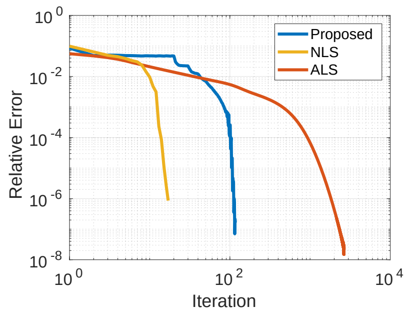
Our proposed algorithm updates two consecutive factor matrices, e.g., $\mathbf{A}_1$ and $\mathbf{A}_2$ by calling Algorithm 1, then proceed with another pair, $\mathbf{A}_3$ and $\mathbf{A}_4$, ..., $\mathbf{A}_j$ and $\mathbf{A}_{j+1}$, until all factor matrices are updated, see its pseudo-code in Algorithm 2. After updating $\mathbf{A}_1$ and $\mathbf{A}_2$, the algorithm shifts the dimensions of the tensor $\mathcal{Y}$ by 2, and updates the next pair. After updating all pairs of factor matrices, the algorithm randomly permutes the factor matrices.

Algorithm 2: CPD with Two Factors Update

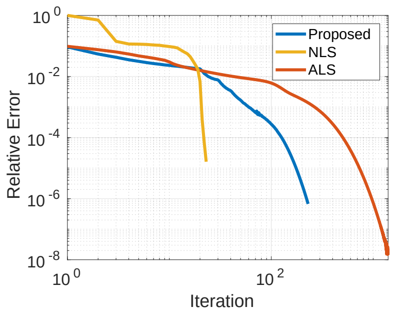
```
Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, and rank $R$
Output: $\hat{\mathcal{Y}} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N]\!]$ such that it minimizes $\|\mathcal{Y} - \hat{\mathcal{Y}}\|_F^2$
1  begin
2      Initialize $\hat{\mathcal{Y}}$
3      while a stopping criterion is not met do
4          $\mathcal{B} \leftarrow$ randompermutation(N)
5          $j \leftarrow 1$
6          while $j \leq |\mathcal{B}| - 1$ do
7              $(n, m) \leftarrow (\mathcal{B}(j), \mathcal{B}(j+1))$
8              $\mathbf{Y}_{(n,m)} \leftarrow (n, m)$-unfolding of $\mathcal{Y}$
9              $\mathbf{\Psi}_{n,m} \leftarrow \mathbf{A}_N \odot \cdots \odot \mathbf{A}_{m+1} \odot \mathbf{A}_{m-1} \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1$
10             Solve $\min_{\mathbf{X}} \|\mathbf{Y}_{(n,m)}^T - \mathbf{\Psi}_{n,m}\mathbf{X}^T\|$ s.t. $\mathbf{X} = \mathbf{A}_m \odot \mathbf{A}_n$ using Algorithm 1
11             $j \leftarrow j + 2$
12         end while
13     end while
14 end
```

3.1 Implementation

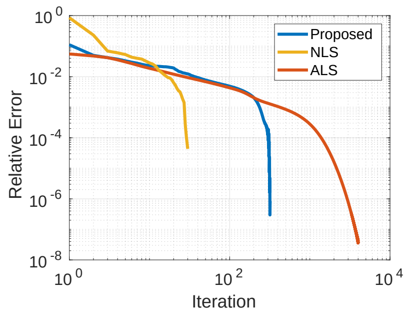
A key observation for an efficient implementation of our proposed Algorithm 2 is the following: the explicit computation of $\mathbf{\Psi}_{1,2}$ is actually not required. Indeed, in Algorithm 1, $\mathbf{\Psi}_{1,2}$ appears in the product $\mathbf{Y}_{(1,2)}^T\mathbf{\Psi}_{1,2}$, which can be computed via R projections of the tensor $\mathcal{Y}$ by $(N-2)$ vectors $\{\mathbf{A}_3(:,r), \ldots, \mathbf{A}_N(:,r)\}$, for $r = 1, 2, \ldots, R$. Finally, the correlation matrix $\mathbf{Q} = \mathbf{\Psi}_{1,2}^T\mathbf{\Psi}_{1,2}$ can be computed with the formula $(\mathbf{A}_N^T\mathbf{A}_N) \circledast \cdots \circledast (\mathbf{A}_3^T\mathbf{A}_3)$ by using the property of the Khatri–Rao product.



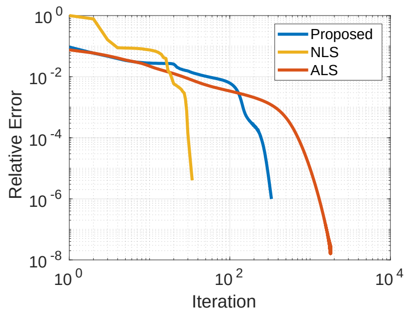
(a) $10 \times 10 \times 10$ rank 10



(b) $10 \times 10 \times 10 \times 10$, rank 10



(c) $50 \times 50 \times 50$, rank 20



(d) $50 \times 50 \times 50 \times 50$, rank 20

Figure 2: Convergence of algorithms in Example 1.

Remark 3. *Another advantage of the proposed algorithm compared to the ALS relies in the possibility to choose matrices $\mathbf{A}_n$ and $\mathbf{A}_m$ such that $\mathbf{\Psi}_{n,m}^T \mathbf{\Psi}_{n,m}$ are well conditioned. For example, we jointly update the two matrices $\mathbf{A}_n$ and $\mathbf{A}_m$ with highest collinearity degrees and these matrices should not join in computing $\mathbf{\Psi}_{i,j}^T \mathbf{\Psi}_{i,j}$, $(i, j) \neq (n, m)$ for updating the other $\mathbf{A}_i$ and $\mathbf{A}_j$. See Example 1.*

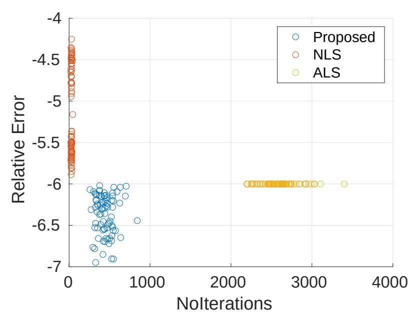
4 Numerical experiments

All tests are preformed using Matlab R2021a and Python 3.9.13 on a laptop Intel CORE i7-11800H CPU @2.30GHz 16GB RAM with GeForce RTX3060 GPU. The proposed algorithm is benchmarked against the state-of-the-art algorithms, ALS Harshman (1970) and NLS Sorber et al. (2013). We ran the decomposition in 5000 iterations and can stop earlier when the difference between consecutive relative errors ($\frac{\|\mathcal{Y}-\hat{\mathcal{Y}}\|_F}{\|\mathcal{Y}\|_F}$) is below 10^{-7} or the change in model parameters is less than 10^{-8}. The algorithms were initialized by the random factor matrices followed by 10 iterations of ALS.

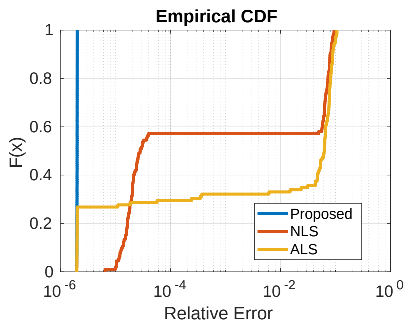
We consider challenging decomposition scenarios in which tensors are generated from high collinear factor matrices or tensors having ranks exceeding their dimensions. It is well known that the computation of the CPD for such tensors is hard due to the degeneracy problem. The results achieved by the algorithms are reported over a total of 100 Monte Carlo (MC) simulations.

Example 1 (Tensors with high collinear factor matrices). In the first experiment, we considered order-3 and 4 tensors of size $I \times \cdots \times I$ with $I = 10$ or 50, and rank $R = 10$ and 20, whose all factor matrices $\mathbf{A}_n$ have highly collinear loading components: $0.97 \leq \mathbf{A}_n^T(:, r)\mathbf{A}_n(:, s) \leq 0.99$, $n = 2, 3, 4$, and $0 \leq \mathbf{A}_1^T(:, r)\mathbf{A}_1(:, s) \leq 0.5$. All loading components have unit length. We used the Matlab routine `gen_matrix` Phan et al. (released 2020) to generate random matrices with specific collinearity degree.

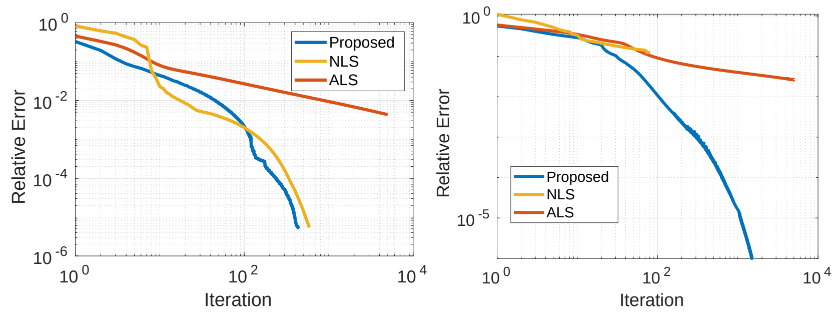
Figure 2 compares the relative errors obtained by three considered algorithms as functions of iteration for different scenarios. The NLS algorithm (Gauss-Newton with dogleg trust region) is an all-at-once optimization method, can handle such different scenarios effectively. The algorithm con-



(a) $50 \times 50 \times 50$ and rank 20



(b) $10 \times 10 \times 10$ and rank 25



(c) Multiplication tensors $(2 \times 3) \times (3 \times 2)$ and $(3 \times 3) \times (3 \times 3)$

Figure 3: (a) The relative errors comparison vs number of iterations of algorithms in Example 1. (b)The probability distribution of achieving a specific relative error in Example 2. (c) Convergence of algorithms in Example 3.

verges quickly in a dozen of iterations. ALS demands at most 2000 iterations (cycles to update all factor matrices) to fully explain the data. Our proposed algorithm, competes primarily with ALS, can reach a similar approximation level in a few hundred iterations. More comparisons are provided in Figure 3 (a). The proposed algorithm requires less number of iterations (update cycles) than ALS, in order to obtain accurate solutions.

Example 2 (Tensors with ranks larger than tensor dimensions). In this experiment, 3rd order random tensors of size $10 \times 10 \times 10$ and rank $R = 25$ are randomly generated. The three algorithms processed the tensors in less than 5000 iterations. Parameters are initialized by random numbers. The corresponding comparison results regarding success ratios defined as cumulative distribution function of the relative errors are reported in Figure 3(b). For this difficult scenario, the proposed algorithm attains a (nearly) perfect success ratio at the relative approximation error of 10^{-6}, while ALS succeeds in less than 30% of its runs. Both ALS and NLS get stuck in false local minima with relative errors greater than 10^{-2}.

Example 3 (Tensors associated with matrix multiplication $(2 \times 3) \times (3 \times 2)$ and $(3 \times 3) \times (3 \times 3)$). In this example, we decomposed multiplication tensors associated with the multiplication of two matrices. The first tensor is of size $6 \times 6 \times 4$ and rank-11, and the second tensor of size $9 \times 9 \times 9$ and rank-23 Strassen (1969); Tichavský et al. (2017), both contain only zeros and ones, and obey $\mathrm{vec}(\mathbf{AB}) = \mathcal{Y} \times_1 \mathrm{vec}\left(\mathbf{A}^T\right)^T \times_2 \mathrm{vec}\left(\mathbf{B}^T\right)^T$ for any matrices $\mathbf{A}$ and $\mathbf{B}$ of the size (2×3), (3×2) for the first tensor, and of size (3×3), (3×3) for the second tensor. Finding CPD of these tensors with minimal rank is related to seeking the fastest multiplication of two matrices.

Comparison of relative errors as a function of iterations for three considered algorithms is provided in Figure 3 (c). In most runs, ALS does not converge to a good solution even after 5000 iterations. NLS works well for the first tensor $(2 \times 3) \times (3 \times 2)$, but it stops earlier for the second tensor because the algorithm reaches one of its stopping conditions, the difference between objective function values and the difference of two consecutive estimates below 10^{-8}. The proposed algorithm can explain the tensor with a relative error below 10^{-6} in around 1000 iterations.

Example 4. (**Compression of ResNet18 convolutional layers)** In this experiment, we employed the proposed algorithm to the problem of compression of the convolutional layers in the ResNet18 model He et al. (2016). The purpose of this application is to develop a light-weight variant of the original neural networks. We decomposed the weights of each convolutional kernel with various ranks $R = 5, 10, 15, 30$. We ran each simulation 10 times, and reported the average results. Resnet18 comprises 4 residual blocks, each block is composed by 4 convolutional layers (and bacthnorm + activation layers). The sizes of the convolutional weights in different layers can be different, and take one of the following dimensions $(64 \times 64 \times 3 \times 3), (128 \times 64 \times 3 \times 3), (128 \times 128 \times 3 \times 3), (256 \times 256 \times 3 \times 3), (512 \times 256 \times 3 \times 3)$ and $(512 \times 512 \times 3 \times 3)$.

The obtained results for all convolutional layers are reported in Figures 4-5 (see the Appendix B). Besides, we investigated the resistance to perturbation of both algorithms. We perturbed the estimated tensor by noise 0.01 and checked the relative error of the tensor. The results of simulations including relative error and sensitivity (SS) Tichavský et al. (2019) are presented in Table 1 (see the Appendix B). As can be seen, the proposed algorithm in most of cases achieves better accuracy compared to the ALS. In addition, the proposed algorithm is more resistant to perturbations. These extensive simulations convinced us that the proposed algorithm can be efficiently utilized for compressing convolutional layer of deep neural networks.

5 Conclusions

We have presented a novel algorithm to address the problem of redundancy in the update rules of ALS-type algorithms, as well as their associated instability issues. Our approach involved updating two factor matrices simultaneously instead of one, and experimental validation confirmed the effectiveness of this idea in resolving the aforementioned difficulties. To evaluate the performance of our proposed algorithm, we conducted experiments using synthetic data tensors and real datasets, including convolutional weights of ResNet18 layers. The extensive simulation results consistently demonstrated the superiority of our algorithm over both the ALS and NLS algorithms across various scenarios. Finally, we established the global convergence of the sequence of iterates generated by Algorithm 1, which is used to solve each subproblem, specifically, the linear regressions with Khatri-Rao structured matrix. However, the potential convergence guarantees for the global algorithm remain unknown. Addressing this uncertainty will be the primary objective of future research, which will also include the development of accelerated versions of Algorithm 1 that come with provable convergence guarantees.

Acknowledgements

The authors would like to thank Le Thi Khanh Hien for helpful discussions around the convergence analysis of our proposed algorithms.

This work was partially supported by the joint project Artificial Intelligence for Life (AIfoL) between the University of Sharjah and the Skolkovo Institute of Science and Technology.

References

Hédy Attouch, Jérôme Bolte, Patrick Redont, and Antoine Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438 – 457, 2010. doi: 10.1287/moor.1100.0449.

Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1):91–129, 2013.

Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Cham, 2017.

Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. The Łojasiewicz inequality for nonsmooth subanalytic functions with applications to subgradient dynamical systems. *SIAM Journal on Optimization*, 17(4):1205–1223, 2007. doi: 10.1137/050644641.

R. Bro and C.A. Andersson. Improving the speed of multiway algorithms - Part II: Compression. *Chemometrics and Intelligent Laboratory Systems*, 42:105–113, 1998.

R. Bro, R. A. Harshman, N. D. Sidiropoulos, and M. E. Lundy. Modeling multi-way data with linearly dependent loadings. *Journal of Chemometrics*, 23(7-8):324–340, 2009. ISSN 1099-128X. doi: 10.1002/cem.1206. URL `http://dx.doi.org/10.1002/cem.1206`.

J.D. Carroll, S. Pruzansky, and J.B. Kruskal. Candelinc: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1):3–24, 1970.

Zev Chonoles. Proof of rank-k matrices forms a closed subset of the space of matrices. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/43194 (version: 2011-06-04).

A. Cichocki, R. Zdunek, A.-H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley, Chichester, 2009.

A. Cichocki, D. P. Mandic, A.-H. Phan, C. Caifa, G. Zhou, Q. Zhao, and L. De Lathauwer. Tensor decompositions for signal processing applications. from two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015.

P. Comon, X. Luciani, and A. L. F. de Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23, 2009.

André LF de Almeida, Xavier Luciani, Alwin Stegeman, and Pierre Comon. Confac decomposition approach to blind identification of underdetermined mixtures based on generating function derivatives. *IEEE Transactions on Signal Processing*, 60(11):5698–5713, 2012.

Felipe Bottega Diniz. A fast implementation for the canonical polyadic decomposition, 2019. URL `https://arxiv.org/abs/1912.02366`.

Eric Evert, Michiel Vandecappelle, and Lieven De Lathauwer. Canonical polyadic decomposition via the generalized schur decomposition. *IEEE Signal Processing Letters*, 29:937–941, 2022.

L. Frølich, T. S. Andersen, and M. Mørup. Rigorous optimisation of multilinear discriminant analysis with Tucker and PARAFAC structures. *BMC Bioinformatics*, 19, 2018.

R.A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Khazhgali Kozhasov. On minimality of determinantal varieties. *Linear Algebra and its Applications*, 626:56–78, 2021. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2021.05.011.

V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

H. Maki, H. Tanaka, S. Sakti, and S. Nakamura. Graph regularized tensor factorization for single-trial eeg analysis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 846–850, 2018. doi: 10.1109/ICASSP.2018.8461897.

Marouane Nazih, Khalid Minaoui, and Pierre Comon. Computation of the regularized canonical polyadic decomposition of tensors using the accelerated proximal gradient with momentum. *hal-03152014*, 2021.

N. Parikh and S.P. Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

A.-H. Phan, P. Tichavský, and A. Cichocki. CANDECOMP/PARAFAC decomposition of high-order tensors through tensor reshaping. *IEEE Transactions on Signal Processing*, 61(19):4847–4860, 2013. ISSN 1053-587X. doi: 10.1109/TSP.2013.2269046.

A.-H Phan, P. Tichavský, and A. Cichocki. Error preserving correction: A method for CP decomposition at a target error bound. *IEEE Transactions on Signal Processing*, 67(5):1175–1190, 2019. doi: 10.1109/TSP.2018.2887192.

A.H. Phan, P. Tichavský, and A. Cichocki. TENSORBOX: MATLAB package for tensor decomposition, released 2020. URL `http://www.bsp.brain.riken.jp/~phan/tensorbox.php`.

Anh-Huy Phan, Petr Tichavský, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, and Andrzej Cichocki. Canonical polyadic tensor decomposition with low-rank factor matrices. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4690–4694. IEEE, 2021.

I. Siaminou and A. P Liavas. An accelerated stochastic gradient for canonical polyadic decomposition. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pp. 1785–1789. IEEE, 2021.

L. Sorber, M. Van Barel, and L. De Lathauwer. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ erms, and a new generalization. *SIAM Journal on Optimization*, 23(2):695–720, 2013. doi: 10.1137/120868323. URL `https://doi.org/10.1137/120868323`.

M. Sørensen, L. De Lathauwer, P. Comon, S. Icart, and L. Deneire. Canonical polyadic decomposition with a columnwise orthonormal factor matrix. *SIAM J. Matrix Anal. Appl.*, 33:1190–1213, 2012.

V. Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356, August 1969. ISSN 0029-599X. doi: 10.1007/BF02165411. URL `http://dx.doi.org/10.1007/BF02165411`.

P. Tichavský, A.-H. Phan, and A. Cichocki. Numerical CP decomposition of some difficult tensors. *J. Computational and Applied Mathematics*, 317:362–370, 2017.

P. Tichavský, A.-H. Phan, and A. Cichocki. Sensitivity in tensor decomposition. *IEEE Signal Processing Letters*, 26(11):1653–1657, 2019.

P. Tichavský, A. Phan, and A. Cichocki. Weighted Krylov-Levenberg-Marquardt method for canonical polyadic tensor decomposition. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3917–3921, 2020.

S. Wakabayashi. Remarks on semi-algebraic functions, April 5 2008.

Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019. doi: 10.1007/s10915-018-0757-z.

Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013. doi: 10.1137/120887795.

Z. Xu, M. A. T. Figueiredo, and T. Goldstein. Adaptive admm with spectral penalty parameter selection. In *International Conf. on Artificial Intelligence and Statistics - AISTATS*, volume N/A, pp. —, April 2017.

T. Yokota, Q. Zhao, and A. Cichocki. Smooth PARAFAC decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016. doi: 10.1109/TSP.2016.2586759.

X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016.

A Convergence analysis

We discuss here the convergence of the iterates generated by Algorithm 1 to solve Problem (5) in the case a fixed value for penalty weight of the augmented Lagrangian function is considered.

Here-under we first present the general formulation of the optimization problem solved by Algorithm 1

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Z}} \quad & f(\mathbf{Z}) + h(\mathbf{X}) \\ \text{s.t.} \quad & \mathbf{AX} + \mathbf{BZ} = 0 \end{aligned} \tag{15}$$

which is a non-convex with non-smooth term problem and where $f(\mathbf{Z}) := \frac{1}{2}\|\mathbf{Y} - \mathbf{\Phi}\mathbf{Z}^T\|_F^2 + \frac{\mu}{2}\|\mathbf{Z}\|_2^2$ with $\mu > 0$, $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = -\mathbf{I}$, and $h(.) := i_{\mathcal{D}}(.)$ is the indicator function associated to set $\mathcal{D} = \{\mathbf{X}|\mathbf{X} = \mathbf{U} \odot \mathbf{V}\}$, which corresponds, as explained in Section 2.2, to the set of matrices such that each of its column is a vectorized rank-1 matrix, and takes the value 0 if $\mathbf{X} \in \mathcal{D}$, and ∞ otherwise. Furthermore, we define the augmented Lagrangian function

$$\mathcal{L}_\beta(\mathbf{X}, \mathbf{Z}, \mathbf{W}) := \phi(\mathbf{X}, \mathbf{Z}) + \langle \mathbf{W}, \mathbf{AX} + \mathbf{BZ} \rangle_F + \frac{\beta}{2}\|\mathbf{AX} + \mathbf{BZ}\|_F^2 \tag{16}$$

where $\langle \mathbf{C}, \mathbf{D} \rangle_F = \text{trace}(\mathbf{C}^T\mathbf{D})$, $\mathbf{W}$ denotes the matrix of dual variables associated to the equality constraints, and $\phi(\mathbf{X}, \mathbf{Z}) := f(\mathbf{Z}) + i_{\mathcal{D}}(\mathbf{X})$ to ease the notation. Note that by posing $\mathbf{T} := \frac{\mathbf{W}}{\beta}$ ($\beta = 1/\gamma$ in (6)), the augmented Lagrangian function can be written in the so-called scaled dual form as follows

$$\mathcal{L}_\beta(\mathbf{X}, \mathbf{Z}, \mathbf{T}) := \phi(\mathbf{X}, \mathbf{Z}) + \frac{\beta}{2}\left(\|\mathbf{AX} + \mathbf{BZ} + \mathbf{T}\|_F^2 - \|\mathbf{T}\|_F^2\right). \tag{17}$$

The general non-convex ADMM for solving Problem (15) is recalled in Algorithm 3.

Algorithm 3: Nonconvex ADMM for solving Problem (4)

```
Input: Initial iterates $\mathbf{X}^{(0)}, \mathbf{Z}^{(0)}, \mathbf{W}^{(0)}$.
Output: $\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)}$
1 begin
2     while stopping criteria not satisfied do
3         $\mathbf{X}^{(k+1)} \leftarrow \arg\min_{\mathbf{X}} \mathcal{L}_\beta(\mathbf{X}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$
4         $\mathbf{Z}^{(k+1)} \leftarrow \arg\min_{\mathbf{Z}} \mathcal{L}_\beta(\mathbf{X}^{(k+1)}, \mathbf{Z}, \mathbf{W}^{(k)})$
5         $\mathbf{W}^{(k+1)} \leftarrow \mathbf{W}^{(k)} + \beta\left(\mathbf{AX}^{(k+1)} + \mathbf{BZ}^{(k+1)}\right)$
6     end while
7 end
```

It turns out that the convergence of the iterates generated by Algorithm 3 (or equivalently by Algorithm 1) to solve Problem (15) can be analysed using the results of Wang et al. (2019). We first recall several key assumptions to be satisfied to call the results from Wang et al. (2019). The first assumption is required to ensure the boundedness of the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$ and relies on the property of coercivity of the objection function within the feasible set:

A.1 (coercivity) Given the feasible set $\mathcal{F} := \{(\mathbf{X}, \mathbf{Z})|\mathbf{AX} + \mathbf{BZ} = 0\}$, the objective function $\phi(\mathbf{X}, \mathbf{Z})$ is called coercive over $\mathcal{F}$ if

$$\phi(\mathbf{X}, \mathbf{Z}) \to \infty \text{ as } \|(\mathbf{X}, \mathbf{Z})\| \to \infty \text{ with } (\mathbf{X}, \mathbf{Z}) \in \mathcal{F}. \tag{18}$$

The coercivity property guarantees that any continuous function possesses a global minimum, and equivalently all its level sets are compact. Given that the feasible set $\mathcal{F}$ is closed, if $\mathcal{F}$ is also bounded, it is therefore compact. Consequently, for any continuous function $\phi(\mathbf{X}, \mathbf{Z})$, a global minimum exists on $\mathcal{F}$, thereby satisfying assumption **A.1** trivially. An important benefit of assumption **A.1** is its relative weakness compared to assuming the objective function is coercive across the entire space. Additionally, assumption **A.1** can be omitted if the boundedness of the sequence can be proven through alternative methods.

The two next assumptions concerns the matrices, $\mathbf{A}$, and $\mathbf{B}$, to ensure a reverse control on the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)})$ based on the sequence $(\mathbf{AX}^{(k)}, \mathbf{BZ}^{(k)})$, which plays a crucial role in the convergence results in Wang et al. (2019).

A.2 (feasibility) $\mathrm{Im}(\mathbf{A}) \subseteq \mathrm{Im}(\mathbf{B})$, where Im(.) returns the image of a matrix.

A.3 (Lipschitz sub-minimization path) For any fixed $\mathbf{X}$, the problem:

$$\min_{\mathbf{Z}} \phi(\mathbf{X}, \mathbf{Z}) : \mathbf{BZ} = \mathbf{U}$$

has a unique minimizer, and $H(\mathbf{U}) := \arg\min_{\mathbf{Z}} \phi(\mathbf{X}, \mathbf{Z}) : \mathbf{BZ} = \mathbf{U}$ is a Lipschitz continuous map with constant $\bar{M}$.

One can see that assumption **A.3** is satisfied when $\mathbf{A}$ and $\mathbf{B}$ have full column rank[1]. The two last assumptions feature the regularity of the terms defining the objective function $\phi(\mathbf{X}, \mathbf{Z}) := f(\mathbf{Z}) + h(\mathbf{X})$.

A.4 (objective-f regularity) $f(.)$ is Lipschitz differentiable with constant L_f, that is the function f is differentiable and its gradient is Lipschitz continuous with constant L_f.

A.5 (objective-h regularity) $h(.)$ is in the form $h(x) := g(x) + h_0(x)$ where:

1. $g(x)$ is Lipschitz differentiable with constant L_g,
2. either $h_0(.)$ is lower semi-continuous.

Here-under we state the main converge result from Wang et al. (2019):

Theorem 2 (Theorem 1, Wang et al. (2019))**.** *Suppose A.1-A.5 hold, then Algorithm 3 converges subsequently for any sufficient large β, that is, starting from any $(\mathbf{X}^{(0)}, \mathbf{Z}^{(0)}, \mathbf{W}^{(0)})$, it generates a sequence that is bounded, has at least one limit point, and that each limit point $(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{W}^{(*)})$ is stationary point of $\mathcal{L}_\beta$. That is $0 \in \partial\mathcal{L}_\beta(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{W}^{(*)})$, or equivalently:*

$$\begin{aligned} 0 &= \mathbf{A}\mathbf{X}^{(*)} + \mathbf{B}\mathbf{Z}^{(*)} \\ 0 &\in \partial f(\mathbf{Z}^{(*)}) + \mathbf{B}^T\mathbf{W}^{(*)} \\ 0 &\in \partial h(\mathbf{X}^{(*)}) + \mathbf{A}^T\mathbf{W}^{(*)} \end{aligned} \tag{19}$$

Moreover, if $\mathcal{L}_\beta$ is a Kurdyka-Łojasiewicz (KL) function (Attouch et al. (2013); Bolte et al. (2007)), then $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$ converges globally, that is regardless of where the initial point is, to the unique limit point $(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{W}^{(*)})$.

Functions that satisfy the KL inequality encompass a variety of types, such as real analytic functions, semi-algebraic functions, and locally strongly convex functions. For further information and detailed explanations, please refer to Section 2.2 in Xu & Yin (2013) and the accompanying references.

We now establish some useful lemmas for our main convergence results.

Lemma 1. *Let $\mathbb{M}$ be the space of all $m \times n$ matrices, and $\mathcal{C} = \{\mathbf{X} \in \mathbb{M} | \mathit{rank}(X) \leq k, k \leq \mathit{min}(m, n)\}$ the collection of rank-k matrices, $\mathcal{C}$, forms a closed subset of $\mathbb{M}$.*

Proof. The proof is provided in Chonoles, however, we recall it for the sake of completeness. Two scenarios have to be considered:

1. $k = \min(m, n)$: then $\mathcal{C} = \mathbb{M}$, and then is a closed set.
2. $k < \min(m, n)$: we first define the continuous map $T : \mathbb{M} \to \mathbb{R}^d$ with $d = C_{k+1}^m \times C_{k+1}^n$ which sends a matrix to the ordered tuple of its $(k+1) \times (k+1)$ minors. For each matrix $\mathbf{X} \in \mathcal{C} \subseteq M$, these minors are equal to zero [2]. Therefore the set $\mathcal{C}$ is the preimage of $0 \in \mathbb{R}^d$ under the map T. Since $\mathbb{R}^d$ is Hausdorff, then $\{0\}$ is a closed set in $\mathbb{R}^d$. Moreover, the map T is continuous, therefore, $\mathcal{C}$ is a closed set in $\mathbb{M}$.

This concludes the proof. □

[1]Indeed, in that scenario, their null spaces are trivial.

[2]A matrix is of rank $\leq k$, i.e. $< k+1$, if and only if all of its $(k+1) \times (k+1)$ minors are zero.

Lemma 2. *The function $h(.)$ from Problem (15), that is the indicator function $i_{\mathcal{D}}(.)$ associated to the set $\mathcal{D} = \{\mathbf{X}|\mathbf{U} \odot \mathbf{V}\}$, is lower semi-continuous (lsc).*

Proof. From Section 2.2, recall that $\mathcal{D}$ is the set of matrices such that each of its columns is a vectorized rank-1 matrix and takes the value 0 if $\mathbf{X} \in \mathcal{D}$, and $+\infty$ otherwise. First, it is obvious that $i_{\mathcal{D}}(.)$ can be reformulated equivalently[3] as follows

$$i_{\mathcal{D}}(\mathbf{X}) = \sum_i^n i_{\mathcal{D}_i}(\mathbf{X}(:,i)), \quad 1 \leq i \leq n, \tag{20}$$

where $i_{\mathcal{D}_i}(.)$ denote the indicator functions associated to the sets $\mathcal{D}_i$ which corresponds to the sets of vectorized rank-1 matrices, which takes also the value 0 if $x_i \in \mathcal{D}_i$, and $+\infty$ otherwise. Therefore, each indicator function $i_{\mathcal{D}_i}(x_i)$ is lower-bounded by zero and then is nonnegative. Moreover, each $i_{\mathcal{D}_i}(x_i)$ is associated with a closed set using Lemma 1 which holds for $k = 1$, and so each $i_{\mathcal{D}_i}(x_i)$ is lower semicontinuous.

It remains to show that $i_{\mathcal{D}}(\mathbf{X})$ is also lower semi-continuous. Recall first the definition of a lsc function: a function $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$ is lower semi-continuous in $x \in \mathbb{R}^n$ if

$$f(x) \leq \liminf_{x' \rightarrow x} f(x') := \sup_{\epsilon > 0} \inf_{x' \in B(x,\epsilon)} f(x').$$

Suppose $f(x) := \sum_i^n f_i(x_i)$ with $x = [x_1; ...; x_n]$, and all $f_i(x_i)$ lsc at x_i and nonnegative, $f(x)$ is lsc since:

$$\begin{aligned} \liminf_{x' \rightarrow x} f(x') &= \liminf_{x' \rightarrow x} (\sum_i^n f_i(x_i')) \\ &\geq \sum_i^n \liminf_{x_i' \rightarrow x_i} f_i(x_i') \\ &\geq \sum_i^n f_i(x_i) := f(x) \end{aligned}$$

where the first inequality holds by the property of the limit inferior of a sum of nonnegative functions, and the last inequality by the lsc property of each $f_i(.)$, which concludes the proof.

□

Using the previous results, we come up with the following theorem which establishes the convergence of Algorithm 3 (or equivalently by Algorithm 1) for solving Problem (15) (or equivalently Problem (5)).

Theorem 3. *For a sufficiently large β, the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$ generated by Algorithm 3 (or equivalently by Algorithm 1) applied to Problem (15) converges globally, that is regardless of where the initial point is, to the unique limit point $(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{W}^{(*)})$, which is a stationary point of the augmented Lagrangian function, $\mathcal{L}_\beta$, and $\mathbf{X}^{(*)}$ is stationary point of Problem (4).*

Proof. In order to prove this theorem, we only need to verify the assumptions **A.1-A.5**, and that $\mathcal{L}_\beta$ is a Kurdyka-Łojasiewicz function.

Recall first that for our particular optimization problem we have $\mathbf{A} = \mathbf{I}$, and $\mathbf{B} = -\mathbf{I}$, then **A.2** is obvious. Moreover, **A.3** holds for both $\mathbf{I}$ and $-\mathbf{I}$ being full column rank.

Furthermore, recall that $\phi(\mathbf{X}, \mathbf{Z}) := f(\mathbf{Z}) + h(\mathbf{X})$ where $f(\mathbf{Z}) := \frac{1}{2}\|\mathbf{Y} - \mathbf{\Phi}\mathbf{Z}^T\|_F^2 + \frac{\mu}{2}\|\mathbf{Z}\|_F^2$ and $h(.) := i_{\mathcal{D}}(.)$. For $\mu > 0$, $f(\mathbf{Z})$ is strongly convex over the entire space, including $\mathcal{D}$, and then $f(\mathbf{Z})$ is super-coercive by [Corollary 11.17, Bauschke & Combettes (2017)] and then coercive. Therefore, **A.1** holds for the coercivity of $f(\mathbf{Z})$ over $\mathcal{D}$ and the specific form of $h(.) = i_{\mathcal{D}}(.)$ [4].

[3]In the sense that it does not change the optimization problem.

[4]Indeed, since $\mathbf{X} = \mathbf{Z}$, then $f(\mathbf{X}) + i_{\mathcal{D}}(\mathbf{X}) \rightarrow \infty$ as $\|\mathbf{X}\| \rightarrow \infty$ for $\mathbf{X} \in \mathcal{D}$ or not, that is for all $\mathbf{X}$.

Assumption **A.4** can be satisfied by setting $g(.) = 0$, and $h_0(.) = i_{\mathcal{D}}(.)$ which is lower semi-continuous by Lemma 2. Assumption **A.5** holds since $f(.)$ is Lipschitz differentiable with constant $L_f = \|\mathbf{\Phi}^T\mathbf{\Phi} + \mu\mathbf{I}\|_2$.

So far, we get the first guarantee of Theorem 2, that is the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$ generated by Algorithm 3 has limit points and all of its limit points are stationary points of the augmented Lagrangian $\mathcal{L}_\beta$. The next step consists in showing that $\mathcal{L}_\beta$ is a Kurdyka-Łojasiewicz (KL) function. In particular we show it is a semi-algebraic function. According to equation (16), $\mathcal{L}_\beta$ boils down to a sum of polynomial functions in $(\mathbf{X}, \mathbf{Z}, \mathbf{W})$ and an indicator function. First, it is well known that polynomial functions are semi-algebraic. Secondly, from Attouch et al. (2010), an indicator function is semi-algebraic if the set with which it is associated is semi-algebraic. The set of rank-1 matrices is semi-algebraic Kozhasov (2021), hence $i_{\mathcal{D}}(.)$ is semi-algebraic as well. Finally, by [Lemma 9, Wakabayashi (2008)] or references in Attouch et al. (2010), the finite sum of semi-algebraic functions is also semi-algebraic, therefore $\mathcal{L}_\beta$ is semi-algebraic and is a Kurdyka-Łojasiewicz function. By Theorem 2, the sequence $(\mathbf{X}^{(k)}, \mathbf{Z}^{(k)}, \mathbf{W}^{(k)})$ converges globally to the unique limit point $(\mathbf{X}^{(*)}, \mathbf{Z}^{(*)}, \mathbf{W}^{(*)})$.

Finally, based on the Karush–Kuhn–Tucker conditions for $\mathcal{L}_\beta$ given in equation 19, and since $\mathbf{A} = \mathbf{I}$, and $\mathbf{B} = -\mathbf{I}$, it is easy to show that $\mathbf{X}^{(*)}$ satisfies $-\nabla f(\mathbf{X}^{(*)}) \in \partial h(\mathbf{X}^{(*)})$, hence $\mathbf{X}^{(*)}$ is a (first-order) stationary point of Problem (4). This concludes the proof.

□

Remark 4. *(Sufficiently large β) In Theorem 3, it is assumed that β is sufficiently large. A lower bound is proposed in [Lemma 9, Wang et al. (2019)]. For our class of problems, the lower bound is as follows*

$$\beta > L_f \bar{M}^2 + 1 + C$$

where $C := L_f \bar{M} \lambda_{++}^{-1/2}(\mathbf{B}^T\mathbf{B})$ and $\lambda_{++}(\mathbf{B}^T\mathbf{B})$ denote the smallest strictly positive eigenvalue of $\mathbf{B}^T\mathbf{B}$, which is equal to 1 in our case. Furthermore, one can show that $\bar{M} = \|\mathbf{I}\|$, therefore we have

$$\beta > \|\mathbf{\Phi}^T\mathbf{\Phi} + \mu\mathbf{I}\|_2(\|\mathbf{I}\|^2 + \|\mathbf{I}\|) + 1. \tag{21}$$

B Simulations on Compression of ResNet18 convolutional layers

In this section, more details of the simulations for compressing the ResNet18 convolutional layers are presented. Table 1 reports the relative error (RelError), relative error of the perturbed tensor (RelErr (perturbed)) and sensitivity (SS) of convolutional layers weights using the tensor ranks 5, 10, 15 and 30 for the proposed algorithm and ALS algorithms. Also, Figure 4 illustrates the norms of rank-1 tensors of the approximations obtained by the proposed and ALS algorithms for different convolutional layers weights using tensor ranks 15 and 30. The simulation results demonstrate the effectiveness of the proposed algorithm in comparison to the ALS algorithm.

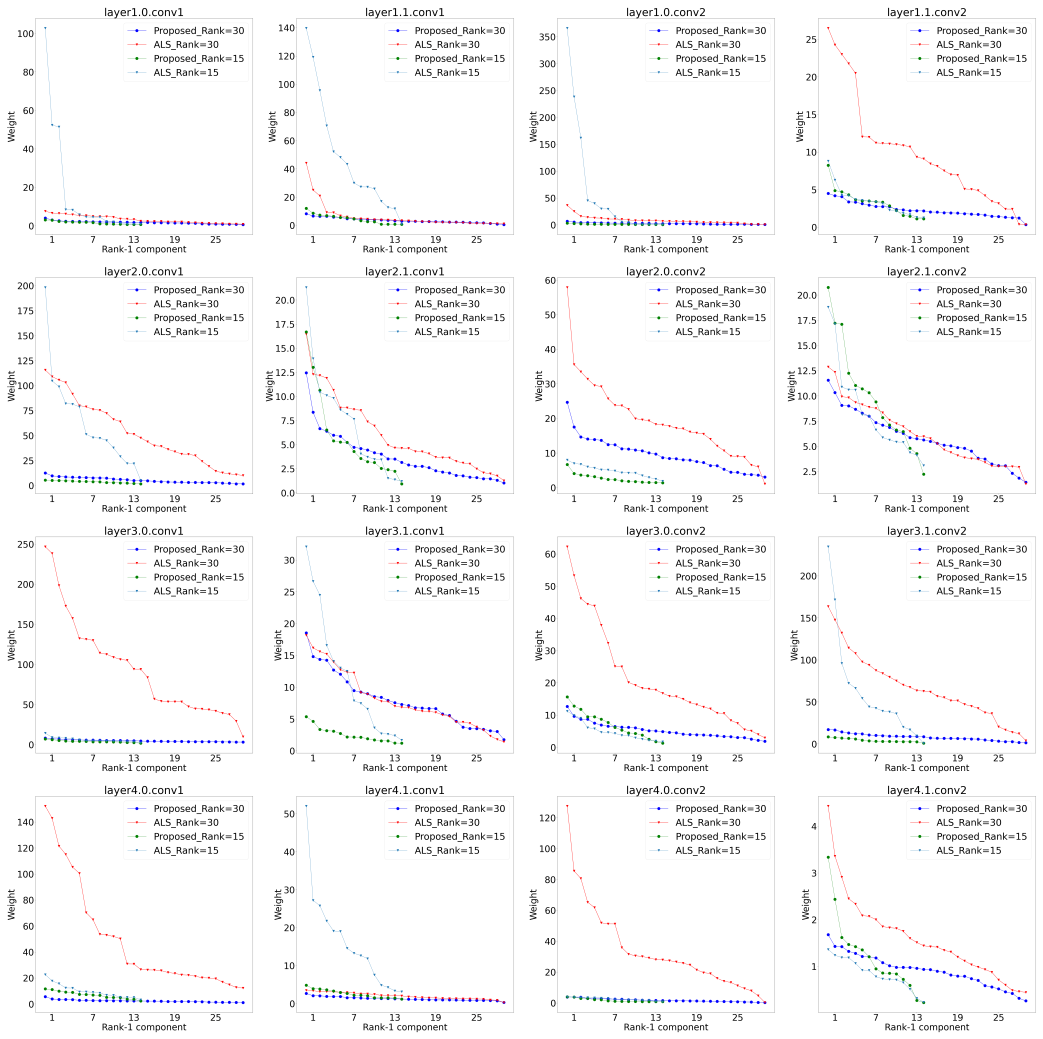


Figure 4: Illustration of norms of rank-1 tensors in the approximation of the convolutional layers weights with ranks 30 and rank 15. Plots marked by ● represent performance of proposed algorithm and plots marked by ▲ represent performance of the ALS algorithm.

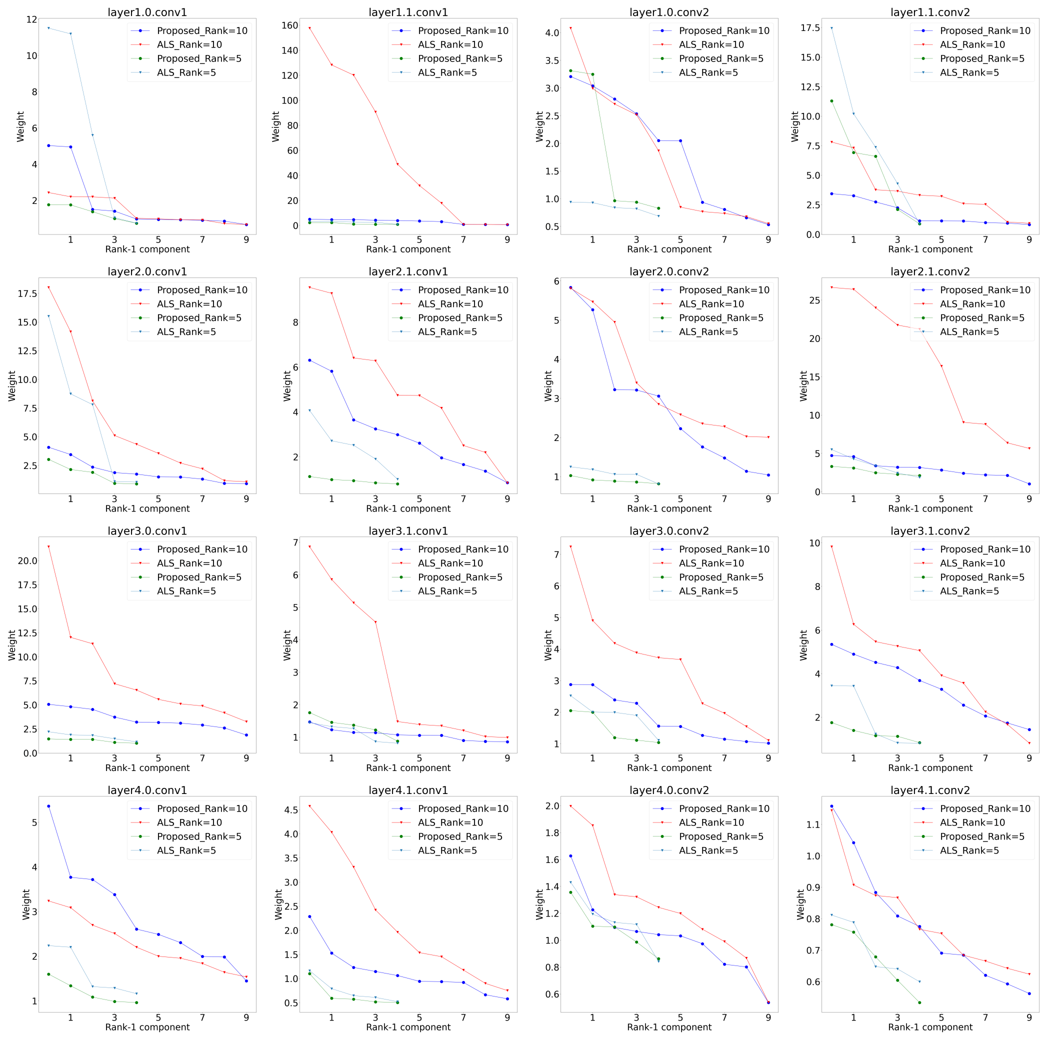


Figure 5: Illustration of norms of rank-1 tensors in the approximation of the convolutional layers weights with ranks 10 and rank 5. Plots marked by ● represent performance of proposed algorithm and plots marked by ▲ represent performance of the ALS algorithm.

Table 1: Relative error (RelError), relative error of the perturbed tensor (RelErr (perturbed)) and sensitivity (SS) of convolutional layers weights with ranks $5, 10, 15$ and 30 for the proposed algorithm and ALS algorithm.

	Rank	5			10			15			30		
Layer		RelError	RelErr (perturbed)	SS	RelError	RelErr (perturbed)	SS	RelError	RelErr (perturbed)	SS	RelError	RelErr (perturbed)	SS
layer1.0.conv1	Proposed	0.8043	0.8135	23.1873	0.6537	0.7110	78.2157	0.5301	0.5819	86.6482	0.3321	0.4900	195.0377
	ALS	0.7890	0.8321	187.9613	0.6522	0.7519	50.3130	0.5300	5.2579	2860.6073	0.3320	1.9053	487.8066
layer1.1.conv1	Proposed	0.8220	0.8300	25.5930	0.6831	0.7842	152.1266	0.5758	0.9085	376.8849	0.4276	0.8343	509.5671
	ALS	0.8220	0.8490	51.2009	0.6791	3.3163	8473.3064	0.6295	16e11	8996.4199	0.4275	1.6544	1321.4498
layer1.0.conv2	Proposed	0.8500	0.8660	37.1706	0.7362	0.7868	80.5953	0.6331	0.6602	63.2832	0.4485	0.7031	335.8023
	ALS	0.8519	0.8608	11.9549	0.7238	0.8150	70.6040	0.6331	23.2213	16561.4565	0.4484	3.0046	1806.0375
layer1.1.conv2	Proposed	0.7784	0.9219	163.1573	0.6581	0.6926	68.8397	0.5886	0.7828	245.7213	0.4709	0.6462	279.8689
	ALS	0.7783	0.8664	268.4477	0.6581	0.8652	179.2805	0.5884	0.7812	247.3091	0.4670	2.2899	2089.6178
layer2.0.conv1	Proposed	0.8914	0.9010	33.7667	0.8088	0.8356	77.6939	0.7411	0.9207	289.3839	0.5828	1.1448	895.7796
	ALS	0.8911	0.9293	222.9128	0.8022	1.0798	384.0700	0.7339	3.9890	12600.1646	0.5827	3.1674	19180.0117
layer2.1.conv1	Proposed	0.9308	0.9328	13.7634	0.8788	0.9362	141.3250	0.8369	1.1574	506.4314	0.7473	1.0406	568.8908
	ALS	0.9307	0.9404	51.1332	0.8780	0.9576	278.7394	0.8352	1.0534	704.1226	0.7473	1.1640	1061.7298
layer2.0.conv2	Proposed	0.9475	0.9492	13.0716	0.9034	0.9502	129.0985	0.8625	0.9220	173.7708	0.7604	1.7980	1895.3969
	ALS	0.9475	0.9511	16.5216	0.9023	0.9611	157.2415	0.8625	1.0076	373.7361	0.7588	1.7372	5040.8004
layer2.1.conv2	Proposed	0.9333	0.9532	55.4105	0.8885	0.9456	131.3680	0.8482	1.6395	1007.3973	0.7517	1.3535	974.9895
	ALS	0.9332	0.9771	81.8726	0.8870	1.6540	1341.6019	0.8481	1.3998	802.7880	0.7516	1.2999	1068.6194
layer3.0.conv1	Proposed	0.9503	0.9532	20.8838	0.9111	0.9595	163.4596	0.8762	0.9689	288.7377	0.7854	1.0478	727.0633
	ALS	0.9503	0.9574	31.0362	0.9110	1.0269	531.4882	0.8761	1.0286	523.1347	0.7833	3.4527	42064.7217
layer3.1.conv1	Proposed	0.9709	0.9739	22.2853	0.9519	0.9558	33.3387	0.9335	0.9712	161.4827	0.8833	1.4990	1493.4552
	ALS	0.9708	0.9736	18.1283	0.9519	0.9862	143.9417	0.9326	1.3037	1348.2310	0.8831	1.2204	1543.8235
layer3.0.conv2	Proposed	0.9750	0.9785	25.7569	0.9534	0.9643	67.8881	0.9340	1.1922	648.4950	0.8825	1.1494	853.2377
	ALS	0.9744	0.9780	35.8819	0.9533	0.9725	164.9044	0.9340	0.9968	405.4532	0.8825	1.5327	5812.9198
layer3.1.conv2	Proposed	0.9492	0.9550	20.5308	0.9119	1.0027	157.2978	0.8747	1.1061	327.0128	0.7880	1.9387	1526.0629
	ALS	0.9477	0.9842	39.6916	0.9107	1.3843	231.9508	0.8746	1948.4988	13263.6543	0.7879	160.4089	26159.7021
layer4.0.conv1	Proposed	0.9179	0.9281	19.1490	0.8645	0.9994	128.1193	0.8246	1.7029	612.7527	0.7503	1.0206	289.2258
	ALS	0.9178	0.9767	29.5483	0.8645	1.3262	90.6376	0.8241	2.1503	1018.9076	0.7502	57.7314	18427.0366
layer4.1.conv1	Proposed	0.7379	0.7798	8.7457	0.5865	0.8620	36.4655	0.4549	1.9542	174.2448	0.3163	1.2698	129.6146
	ALS	0.7378	1.0874	10.3198	0.5795	6.1267	92.5787	0.4523	15.6830	2033.7755	0.3161	6.0951	238.4433
layer4.0.conv2	Proposed	0.8045	0.8441	16.7329	0.6356	0.7256	31.3607	0.5524	1.0111	104.6715	0.4387	1.2306	203.1567
	ALS	0.8044	1.3312	18.0358	0.6355	2.9609	41.0731	0.5517	6.4380	160.9665	0.4383	20.2696	11291.1104
layer4.1.conv2	Proposed	0.6899	0.7478	8.8443	0.3969	0.6275	21.8960	0.3403	1.1242	63.0682	0.2464	1.1098	88.7904
	ALS	0.6898	3.5661	9.3045	0.3961	22.7436	22.1786	0.3402	17.3921	36.6722	0.2462	37.3849	216.4059