Department: Head
Editor: Name, xxxx@email

Tensorlab$^+$: A case study on reproducibility in tensor research

Stijn Hendrikx[*†], **Raphaël Widdershoven**[*], **Nico Vervliet**[*†], **Lieven De Lathauwer**[*†]
[*]Department of Electrical Engineering (ESAT), KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium.
[†]Group Science, Engineering and Technology, KU Leuven Kulak, E. Sabbelaan 53, B-8500 Kortrijk, Belgium.

***Abstract*—Tensor methods emerge as an important class of basic techniques, generalizing matrix methods to multi-way data and models. We have recently released Tensorlab$^+$, which is a downloadable archive of code and data that allows peers to reproduce the experiments reported in our publications on tensor decompositions and applications. We briefly discuss the basic tensor tools and give an introduction to the contents of Tensorlab$^+$. We elaborate on the steps that were taken to ensure the reproducibility of the experiments and the quality of the code.**

REPLICATION is the gold standard for guaranteeing the validity of scientific claims in research. However, performing an independent replication of a study, especially if it is a large study, can be expensive and time-consuming. Furthermore, replication studies are not well incentivized, as journals often prefer original research. Many experiments in research are numerical, for which reproduction is an attractive alternative to replication for assessing scientific claims, as it requires far less time and resources than the latter.

Reproducibility in research is defined as "obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis" [1]. In the context of numerical experiments, this boils down to running the code of the original authors on the data they collected or used. As a result, numerical experiments are reproducible when the code and data are publicly available. While any code that produces the figures and tables from the corresponding paper technically suffices, high quality code is much more useful to the broader research community. Naturally, reproducing the results of numerical experiments is much easier than replicating them through an independent study, since the latter requires collecting a new dataset and performing the analysis again. The former allows anyone to inspect the data, numerical analysis and results of the original authors in order to assess the validity of their claims. The only cost of reproduction is the computational effort required

 Published by the IEEE Computer Society © 2023 IEEE

to rerun the experiments.

Besides enabling reproducibility, sharing code and data has other major advantages. First, the different steps of the numerical analysis are briefly discussed in a paper, but many details are left out as it is often infeasible to present every single one. By sharing the code, every parameter and every step of the analysis is made available, making it much easier for other researchers to gain a deep knowledge of and to build upon the original research. Second, in numerical experiments, results are often compared to results obtained by similar algorithms in terms of both accuracy and computation time. If the code for these algorithms is available, such a comparison can easily be performed. If this is not the case, one might have to write one's own implementation using the limited details in the paper. A possible skill gap in terms of code development with the original authors may result in incorrect conclusions being drawn from this comparison. For example, an increase in performance could be interpreted as an algorithmic improvement, while the actual cause is that it is being compared to an implementation of an algorithm that is worse than the original, unavailable one. Given the advantages, sharing code should be the standard in research involving computational experiments.

A considerable amount of research on reproducibility has been published in literature, including in this journal. An entire issue was dedicated to this topic [2]. In another notable example, the authors developed software that allows anyone to reproduce numerical results [3]. Some articles indicate concern about the current state of reproducibility in research. A survey conducted by *Nature* revealed that 90% of the respondents think there is a reproducibility crisis and 70% tried and failed to reproduce research performed by others [4]. Reproducibility has also received attention in machine learning in general and even in research on tensor decompositions [5]. The latter work is focused on the retrieval of the true factor matrices of tensor decompositions that are unique, whereas the present paper is about achieving reproducibility in tensor research in general.

Despite the attention given to reproducibility in literature and all the advantages of sharing code, in practice and in our personal experience, code and data are often not available. The largest barrier withholding authors from publishing their code is the time and effort it takes to clean up and document the code [6]. This is understandable, as software development is not a researcher's main focus. Still, sharing code that is lacking in quality is already more helpful than not sharing any code at all.

Since the code for some of our own publications was not available either, we have recently released Tensorlab^{+}[1] [7]. It is a downloadable archive of high-quality code and data. Its main purpose is to enable others to easily reproduce the experiments in the corresponding publications. It also allows peers to study the algorithms in detail and facilitates their use as a baseline for comparison with other algorithms in future research. The approach that we adopted for the development of this archive may be useful for anyone interested in releasing reproducible code related to their own work.

We first elaborate in more detail on the contents and objectives of Tensorlab^{+} and how these objectives have been achieved. Tensors and tensor decompositions form the foundation of our research and therefore are also at the core of Tensorlab^{+}. These concepts are introduced in the next section. In the last section, we showcase some example applications of tensors, for which the results can be reproduced through Tensorlab^{+}. We specifically focus on some key elements of these experiments that guarantee their reproducibility.

Tensorlab^{+}

In 2016, we released the latest version of Tensorlab[2], which is a MATLAB software package for working with tensors, computing tensor decompositions and performing complex optimization. It consists mainly of tensor decomposition algorithms and utility and visualization functions, which are designed in a user-friendly way. Tensorlab^{+}, on the other hand, is a downloadable archive of code and data specifically aimed at reproducing experiments from our publications. Tensorlab^{+} is an extension of Tensorlab, which is where it gets its name. In this section, we

[1] https://www.tensorlabplus.net
[2] https://www.tensorlab.net

elaborate on the contents of Tensorlab+ and our objectives for this archive besides reproducibility. These objectives have been achieved by translating them into a set of concrete guidelines for code development. Adherence to these guidelines was checked by employing quality control procedures and tools.

Concretely, Tensorlab+ is a collection of scripts, algorithms and utility functions, which are grouped per publication, as illustrated in Figure 1. Currently, it contains code and data for reproducing the results of 43 publications and counting. Depending on the needs of the user, the code for each publication can be downloaded separately or the entire archive can be downloaded at once. First and foremost, the archive contains the algorithms that were introduced in their respective publications in the case that those algorithms were not yet available in Tensorlab. Second, tutorials have been included that illustrate how the algorithms can be applied, which makes them easy to use. Third, the archive contains experiment scripts that can be used to reproduce the figures and tables from the corresponding paper. These scripts contain all the correct parameter values and can thus be run without needing to be altered. Finally, for each paper, a README file is included, which gives an overview of the included files and lists code and data dependencies.

On top of the main objective of reproducibility, we aim to ensure that Tensorlab+ is of high quality and easy to use. Specifically, the former means that the code should be readable, well-documented, and efficient and that it functions correctly and consistently. The latter means that it should be easy to reproduce an experiment or apply an algorithm without a need to study all the details in the paper or code. These additional objectives both improve the reproducibility of the results and make the code more valuable in its own right. To achieve the objectives for Tensorlab+, we treat the project with a software development approach, which entails clear guidelines, testing and internal code reviews.

While Tensorlab+ aims to facilitate reproducibility in tensor research, it is important to note that the software is subject to specific licensing terms set forth by KU Leuven. The license grants a non-transferable, time-limited use for academic, non-commercial purposes only. Commercial use requires a separate license agreement. Any redistribution, modification, or improvement of the software necessitates prior written approval from KU Leuven. Users are advised to review the full license agreement on the website for a comprehensive understanding of these terms and their implications for research use and reproducibility.

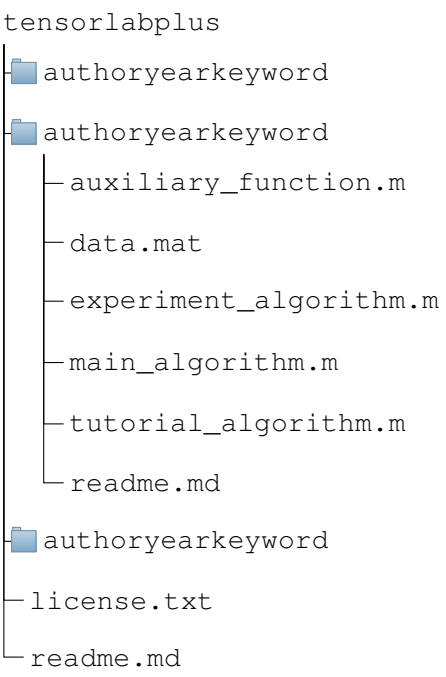


Figure 1. In the full archive, the code for each publication is located in a separate folder. The name of this folder consists of the last name of the first author, the publication year and a keyword for convenience.

Development guidelines

The general objectives were translated into a concrete set of guidelines. We wrote documentation that outlines those, to streamline the process of adding new code to Tensorlab+. Most guidelines correspond to good coding practices that are common in software development but less so in research. They can be summarized as:

1) Ensure easy readability of code by adding line comments and by using a proper naming convention for variables and functions. All files have documentation in the style of built-in Matlab functions. Documentation of a function consists of a one-line summary on top, followed by a longer summary that includes a description of input and

output arguments, and names of related functions. Additionally, the names of the authors, the version history and references are given. The format of an experiment script is similar, but the longer summary is a description of the experiment itself and it references the reproduced figure or table. Readability is a key aspect of reproducibility since it makes the different steps of the experiment, the algorithmic choices made by the authors and the parameters settings more clear.

2) Simplify and shorten code, without affecting its readability. Optimize code with a profiler and use built-in vectorized functions when possible.
3) Add tests for important functions and algorithms. In this way, changes to the code that introduce errors or hamper the reproducibility of experiments can be detected soon after their introduction.
4) Use version control software, e.g. GIT, for code development. Even small changes to code can cause an experiment to no longer produce the same results. By going back into the version history, the cause can be found and addressed. Since Tensorlab$^+$ is a large project involving the code for many papers, changes were often made simultaneously by multiple contributors and for different papers. Version control software makes combining these, possibly conflicting, changes far easier than doing so manually.

Quality control

To guarantee the reproducibility of code in different software environments, two different approaches are possible. One approach is to ensure that the environment in which the code is executed is always exactly the same, which in turn ensures the reproducibility of the results. To this end, one may use a package manager such as Conda or, taking it a step further, a virtual container like Docker. Another approach is to allow the environment to change and check whether the results remain reproducible. We opted for the second approach and created an automated tool for checking the reproducibility of the code. Specifically, this software tool runs all tests to ensure the correctness of the algorithms and also runs all experiment and tutorial scripts to ensure that they complete without errors. Additionally, the tool bundles the relevant files for each individual paper and for the entire collection, while resolving internal code dependencies in which functions are used for multiple papers. The resulting packages are then made available on the Tensorlab$^+$ website in compressed form.

The last step in the process of adding code to Tensorlab$^+$ is to have the new code reviewed by team members to check whether it adheres to the guidelines mentioned above. The review stage consists of multiple rounds, performed by at least two different reviewers. During these rounds, the quality of the code is further improved according to the reviewers' comments. Additionally, the reproducibility is checked by comparing the results of the experiments to those reported in the corresponding paper. These reviews were facilitated by GIT merge requests. These ensure that the review process is fully completed before the new code is added because approval is required for performing the merge.

Lessons learned

In our opinion, the most valuable lesson we learned when developing Tensorlab$^+$ was to keep reproducibility in mind from the very start of writing code. This may seem obvious, but a considerable amount of work on this project involved retroactively improving the code for experiments in older publications. For example, for some experiments it was difficult to retrieve the script that produced the reported results. In other cases, available older code produced results that did not exactly match those reported in the corresponding paper. Resolving these issues long after publication is of course much more time-consuming than putting a bit more effort into the code from the beginning. In the meantime, we have cultivated a mindset for incorporating the guidelines for reproducibility from the beginning of any new research project.

Initially, the guidelines were not written down and they evolved as the Tensorlab$^+$ project progressed. This means that code that was developed in the early stages, did not necessarily meet the later criteria. It is beneficial to set the standards for one's code as early as possible. Note that

good coding practices and documentation styles to start from are already available for different programming languages.

For a large project, we also advise the use of automated tools for tasks like testing, building and formatting code. In this way, relatively simple issues are resolved before the code is reviewed by team members so that the reviews can focus on more important aspects like reproducibility.

Introduction to tensors and decompositions

The experiments in Tensorlab$^+$ involve tensors and tensor decompositions. A tensor is a multiway array of numerical values and the order of a tensor corresponds to the number of dimensions. Therefore, vectors and matrices are first- and second-order tensors, respectively. Representing higher-order data by tensors is inherently advantageous since the multidimensional structure is preserved, while this structure would be lost if the data is flattened and represented by a matrix. Although the generalization from matrix to tensor is straightforward in a sense, that is, stacking multiple matrices of the same size in a third mode yields a third-order tensor, the properties of a tensor are far from simple generalizations of matrix properties. For example, the column rank, row rank and rank of a matrix are all equal and can easily be computed using a rank-revealing decomposition, for instance, the QR factorization. In a tensor, the dimensions of the subspaces spanned by the vectors along each mode are not necessarily equal to each other, nor equal to the rank of the tensor, which is in general NP-hard to compute. The different properties of tensors give tensor decompositions certain advantages over their matrix counterparts. For example, tensor decompositions generally offer greater compression ratios and some are unique under mild conditions. Thanks to these advantages, tensor methods have been applied in a wide range of fields, such as speech and audio processing, biomedical engineering, machine learning and chemometrics [8]. An overview of the tensor decompositions discussed in this section is shown in Figure 2.

A matrix of rank R can be written as the matrix product of two matrices, with R columns and R rows, respectively, and also as the sum of R rank-1 matrices

$$\mathbf{M} = \mathbf{A}\mathbf{B}^{\mathrm{T}} = \sum_{r=1}^{R} \mathbf{a}_r \mathbf{b}_r^{\mathrm{T}} = \sum_{r=1}^{R} \mathbf{a}_r \otimes \mathbf{b}_r, \quad (1)$$

with $\mathbf{M} \in \mathbb{R}^{I \times J}$, $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_R \end{bmatrix} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_R \end{bmatrix} \in \mathbb{R}^{J \times R}$ and $\otimes$ denoting the outer product. This decomposition is not unique, since

$$\mathbf{M} = (\mathbf{A}\mathbf{W}) \left(\mathbf{B}\mathbf{W}^{-\mathrm{T}}\right)^{\mathrm{T}} \quad (2)$$

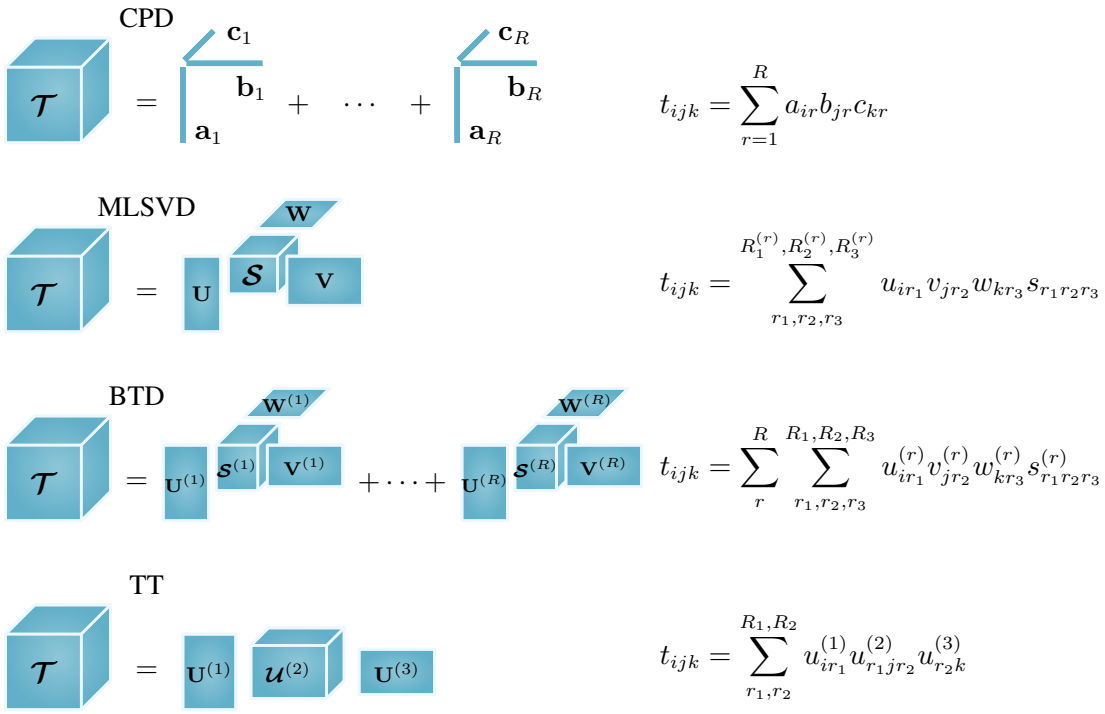
is also a rank-R decomposition of $\mathbf{M}$ for any invertible matrix $\mathbf{W} \in \mathbb{R}^{R \times R}$. Note that tensors and their decompositions are defined over the real field for simplicity; it is also possible to define them over the complex field.

The rightmost expression in Equation (1) is straightforward to generalize to higher-order tensors. Namely, a rank-1 tensor of order N equals the outer product of N nonzero vectors. A tensor can be decomposed as

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{\widetilde{R}} \mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r, \quad (3)$$

with $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I \times J \times K}$ and factor matrices $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{I \times \widetilde{R}}$, $\mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \dots & \mathbf{b}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{J \times \widetilde{R}}$ and $\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & \dots & \mathbf{c}_{\widetilde{R}} \end{bmatrix} \in \mathbb{R}^{K \times \widetilde{R}}$, which is called a polyadic decomposition. The rank R of a tensor is defined in terms of this decomposition, namely it is the minimum value of $\widetilde{R}$ for which a decomposition as in Equation (3) is possible or, in words, the minimal number of rank-1 tensors that sum up to $\boldsymbol{\mathcal{T}}$. In this case, the decomposition is called the canonical polyadic decomposition (CPD). Although computing the exact rank of a tensor is NP-hard in general, computing a low-rank approximation of a tensor often works well in engineering [9].

The CPD is unique under mild conditions, making it a very useful tool in applications such as blind signal separation (BSS). In BSS, the goal is to recover a set of unobserved source signals given multiple observed signals that are mixtures of these source signals. The mixing coefficients are unknown as well. Many types of BSS problems can be translated into the computation of the CPD of a tensor $\boldsymbol{\mathcal{T}}$ that is derived from the observed signals. The source signals and mixing vectors can then be retrieved if the decomposition in Equation (3) is unique. The most famous CPD



$$t_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$$

$$t_{ijk} = \sum_{r_1,r_2,r_3}^{R_1^{(r)},R_2^{(r)},R_3^{(r)}} u_{ir_1} v_{jr_2} w_{kr_3} s_{r_1 r_2 r_3}$$

$$t_{ijk} = \sum_{r}^{R} \sum_{r_1,r_2,r_3}^{R_1,R_2,R_3} u_{ir_1}^{(r)} v_{jr_2}^{(r)} w_{kr_3}^{(r)} s_{r_1 r_2 r_3}^{(r)}$$

$$t_{ijk} = \sum_{r_1,r_2}^{R_1,R_2} u_{ir_1}^{(1)} u_{r_1 j r_2}^{(2)} u_{r_2 k}^{(3)}$$

Figure 2. Graphical representations and elementwise formulas for basic tensor decompositions of third-order tensors.

uniqueness condition is Kruskal's condition. The generic variant of this condition, which ignores linear dependencies of factors that can only occur in a set of measure zero, is a simple relation between the dimensions of the tensor and its rank:

$$\min(I, R) + \min(J, R) + \min(K, R) \geq 2R + 2.$$

More conditions for the uniqueness of the CPD are available in literature; some are even less restrictive than Kruskal's condition and also have simple generic counterparts [8]. In contrast, a specific matrix decomposition like the singular value decomposition (SVD) owes its uniqueness (up to elementary indeterminacies) to much more restrictive constraints, including orthonormality of the singular vectors. Such constraints may not be very useful in BSS, since neither mixing vectors nor sources are orthonormal in general.

Additionally, tensor decompositions generally offer greater compression ratios for low-rank approximation than matrix decompositions. Consider a matrix and an Nth order tensor, in which each dimension is equal to I. The compression ratios for a matrix decomposition, as formulated in Equation (1), and the CPD are $\mathcal{O}\left(\frac{I^2}{2IR}\right)$ and $\mathcal{O}\left(\frac{I^N}{NIR}\right)$, respectively. The latter ratio even increases exponentially for increasing N. One might expect that R also increases as the order increases; however, this is not the case in applications where R has a physical meaning, such as the number of sources in BSS.

The multilinear singular value decomposition (MLSVD) aims to extend some of the very useful properties of the SVD to higher-order tensors. It can be expressed in the form

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{S}} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V} \cdot_3 \mathbf{W},$$

with factor matrices $\mathbf{U} \in \mathbb{R}^{I \times R_1}$, $\mathbf{V} \in \mathbb{R}^{J \times R_2}$ and $\mathbf{W} \in \mathbb{R}^{K \times R_3}$, core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{R_1 \times R_2 \times R_3}$ and $\cdot_n$ denoting the mode-n product. The generalization from SVD to MLSVD becomes more apparent if the SVD is also formulated using the mode-n product:

$$\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} = \boldsymbol{\Sigma} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V}.$$

Like the factors $\mathbf{U}$ and $\mathbf{V}$ in the SVD, factors $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ are column-wise orthonormal.

However, there are also differences between the properties of the SVD and the MLSVD. First, the dimensions of the subspaces spanned by the vectors along each mode of a tensor are not necessarily equal. The multilinear rank $(R_1, R_2, \ldots, R_N)$ of a tensor captures the dimensions of these subspaces. Second, whereas the matrix singular values appear directly on the diagonal of $\mathbf{\Sigma}$, the multilinear singular values equal the Frobenius norms of the slices of the core along each mode. A slice is a subtensor of order $N-1$, obtained by fixing all but one index. Third, the core tensor $\boldsymbol{\mathcal{S}}$ possesses no diagonal structure, unlike its counterpart $\mathbf{\Sigma}$ in the SVD. Instead, the slices of the core are mutually orthogonal along each mode. In fact, this property also holds true for $\mathbf{\Sigma}$, since the rows (and the columns) of a diagonal matrix are mutually orthogonal. Fourth, while truncating the SVD to R components yields the best rank-R approximation of a matrix, truncating the MLSVD does not necessarily yield an optimal low multilinear rank approximation in the least squares sense.

Although truncation typically yields a suboptimal low multilinear rank approximation, it is relatively close to the optimum in terms of approximation error, making the MLSVD a powerful tool for compression. The square of the approximation error is upper bounded by the sum of the squared multilinear singular values that have been discarded. Next, in applications with a sufficiently large signal-to-noise ratio, the dominant multilinear singular values correspond to the signal, while the other, less significant values correspond to noise. In this case, truncation serves as a method for signal denoising. The dominant multilinear singular vectors also constitute a basis for the "signal subspace" along each mode. Additionally, the MLSVD may be used for an extension of principal component analysis to the higher-order setting, since the multilinear singular vectors correspond to the principal components in each mode. Finally, the MLSVD can be computed as a sequence of N SVDs; note that here the orthogonality is desired, as it is numerically advantageous.

As a rank-1 tensor is rather simple, the CPD might not be the right tool in applications where the components of interest possess a more complex structure. In this context, the block term decomposition (BTD) may be a more suitable tool. Each term in a BTD has low multilinear rank instead of rank equal to one. This gives the model the flexibility to capture more complex components while retaining mild uniqueness properties.

Tensors suffer from the curse of dimensionality (CoD), that is, the number of entries in a tensor scales exponentially with the order; for a tensor in which each dimension is equal to I, the number of entries equals I^N. This makes it infeasible in terms of computation time and storage to work directly with tensors of very high order. The MLSVD is not suitable in the high-order setting, since the core has the same order as the original tensor. In this case, decompositions such as the tensor train (TT) or hierarchical Tucker decomposition are more appropriate, which are both specific types of the broader class of tensor networks. They are able to break the CoD because they consist of tensor factors of order three at most. Moreover, thanks to orthogonality, they can be computed in a numerically reliable manner, similar to the MLSVD. Tensor networks are of great interest in fields such as scientific computing and quantum many-body physics [10]. In fact, tensor trains have allowed one to solve very large linear systems, even involving more unknowns than the number of atoms in the universe, while requiring a limited amount of computation time.

Example applications of tensor methods

Tensorlab$^+$ contains many experiments that showcase applications of tensor tools. Some of these experiments are discussed in this section, with a special focus on how they were designed to be reproducible.

Script `experiment_figure4.m` in folder `bousse2017bss`, for example, reproduces a proof-of-concept of a tensor-based method for BSS [11]. Consider the basic linear signal model

$$\mathbf{X} = \mathbf{MS} + \mathbf{N}, \tag{4}$$

in which the rows of $\mathbf{X}$ are the observed signals, the rows of $\mathbf{S}$ are the source signals, $\mathbf{M}$ contains the mixing vectors and $\mathbf{N}$ is a perturbation caused by noise.

The matrix factorization in Equation (4) is generally not unique, as shown in Equation (2). Therefore, the source signals and mixing vectors

cannot be uniquely retrieved by computing a matrix decomposition of $\mathbf{X}$. This means that some assumptions are required to ensure their retrieval. For example, the source signals may be assumed to exhibit low-rank structure after segmentation, i.e., after stacking consecutive segments in a matrix [11]. This assumption is reasonable for signals that can be closely approximated by, for instance, a sum of exponentials, since segmenting a sum of exponentials produces a low-rank matrix. In the script, this assumption is exploited by segmenting the observed signals in $\mathbf{X}$ using the `segmentize` function from Tensorlab. It can be shown that segmenting the rows of $\mathbf{X}$ transforms the problem in Equation (4) into a BTD. The rth term of this BTD is a segmented representation of the contribution of the rth source signal to the observed data. This term has multilinear rank $(L_r, L_r, 1)$ if the signal can be expressed as a sum of L_r exponentials. The BTD is computed using the `lp_nls` algorithm that is included along with this script. The source signals can be retrieved, without the need of restrictive constraints like orthogonality, thanks to the uniqueness of the BTD under mild conditions. In this particular experiment, no noise $\mathbf{N}$ is present and the original source signals can be retrieved exactly.

The BTD in the reformulated BSS problem is computed iteratively using an optimization algorithm. The choice of the starting point is important for reproducibility, since it affects the number of iterations required to get to the final solution as well as the final solution itself. Optimization problems associated with the computation of tensor decompositions are generally nonconvex, which means that the algorithm might end up in a local optimum, depending on the choice of starting point. This variability in the final solutions can be mitigated by starting from a fixed point. For example, under reasonable conditions the CPD of a noise-free tensor can be computed deterministically by means of a generalized eigenvalue decomposition and this solution may provide a good initial point for the noisy case. However, it is not always possible to find a good initialization for more complicated decompositions, such as the BTD, so a random starting point is often used instead. As a result, running an optimization algorithm multiple times may result in different solutions of varying quality. In experiments such as `experiment_figure4.m` the goal is to show that the algorithm can solve a specific instance of the BSS problem. In this case, the reproducibility of the results can be guaranteed by using a fixed seed for the random number generator.

Besides experiments for a single concrete problem, papers typically also contain experiments that show the general performance of an algorithm, such as in `experiment_figure6.m` in folder `bousse2017bss`. Here, fixing the seed of a random number generator is not the appropriate way to guarantee reproducibility. Instead, the experiment should be designed in a way that reveals the average performance in a robust manner. To this end, one should perform a Monte Carlo experiment and aggregate the results using a robust statistic. For example, 100 Monte Carlo trials are performed in `experiment_figure6.m` and they are aggregated using the median. Depending on the number of trials, running the experiment can be computationally expensive. For some scripts, including this one, precomputed results are included such that figures in the corresponding publication can be created without effort. In other scripts, the number of trials is set to a lower default value that still allows one to verify the main conclusions, while the number that was used for the publication is mentioned in a comment. This gives a user the choice between reproducing the original results or quickly producing results that might be slightly different.

Tensorlab$^+$ further includes a variety of methods for computing a CPD or BTD with constraints imposed upon the factor matrices. Constraints can be useful for multiple reasons, including the incorporation of prior knowledge, for example, Vandermonde structure in harmonic retrieval or nonnegative values summing to one for concentrations. Besides offering algorithms for computing decompositions with specific constraints, Tensorlab$^+$ also includes experiments that use the structured data fusion framework [12]. This framework allows one to impose various constraints at once through parameterization, and to couple the decompositions of multiple tensors. Here, coupling means that the decompositions share some factor(s) or underlying generator(s).

Structured data fusion is very versatile and can be used to quickly try out different models and constraints on the problem at hand.

CONCLUSION

Reproduction is an attractive alternative to replication, especially in tensor-based science and engineering, since many experiments mainly consist of running code. Sharing code offers additional advantages besides making research reproducible. We have recently released a comprehensive downloadable archive of code and data for reproducing experiments in our publications. Our approach to ensuring the reproducibility of experiments might be useful to other researchers. We have given an overview of what a user can expect of Tensorlab$^+$, with an emphasis on the underlying motivations and insights of broader interest.

ACKNOWLEDGMENT

We thank Eric Evert for proofreading the manuscript.

This research received funding from the Flemish Government (AI Research Program). LDL, SH, RW and NV are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium. This work was supported by the Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no G0F6718N (SeLMA). NV is supported by postdoctoral fellowships (12ZM220N and 12ZM233N) from the Research Foundation—Flanders (FWO). KU Leuven Internal Funds: C16/15/059, IDN/19/014, C14/22/096.

References

[1] National Academies of Sciences, Engineering, and Medicine, *Reproducibility and Replicability in Science*. Washington, DC: The National Academies Press, 2019, ISBN: 978-0-309-48616-3. DOI: 10.17226/25303.

[2] S. Fomel and J. F. Claerbout, "Guest editors' introduction: Reproducible research," *Computing in Science & Engineering*, vol. 11, no. 1, pp. 5–7, 2009. DOI: 10.1109/MCSE.2009.14.

[3] M. Schwab, N. Karrenbach, and J. Claerbout, "Making scientific computations reproducible," *Computing in Science & Engineering*, vol. 2, no. 6, pp. 61–67, 2000. DOI: 10.1109/5992.881708.

[4] M. Baker, "Reproducibility crisis," *Nature*, vol. 533, no. 26, pp. 353–66, 2016. DOI: 10.1038/533452a.

[5] T. Adali, F. Kantar, M. A. B. S. Akhonda, S. Strother, V. D. Calhoun, and E. Acar, "Reproducibility in matrix and tensor decompositions: Focus on model match, interpretability, and uniqueness," *IEEE Signal Processing Magazine*, vol. 39, no. 4, pp. 8–24, 2022. DOI: 10.1109/MSP.2022.3163870.

[6] V. Stodden, "The scientific method in practice: Reproducibility in the computational sciences," Feb. 2010, MIT Sloan Research Paper No. 4773-10. DOI: 10.2139/ssrn.1550193.

[7] S. Hendrikx, M. Boussé, N. Vervliet, M. Vandecappelle, R. Kenis, and L. De Lathauwer, *Tensorlab$^+$*, Available online, Version of Feb 2022 downloaded from https://www.tensorlabplus.net.

[8] N. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017. DOI: 10.1109/TSP.2017.2690524.

[9] E. Evert and L. De Lathauwer, "Guarantees for existence of a best canonical polyadic approximation of a noisy low-rank tensor," *SIAM Journal On Matrix Analysis And Applications*, vol. 43, no. 1, pp. 328–369, Mar. 2022, ISSN: 0895-4798. DOI: 10.1137/20M1381046.

[10] B. N. Khoromskij, *Tensor Numerical Methods in Scientific Computing*. Berlin/Boston: De Gruyter, Jun. 2018, ISBN: 978-3-11-036591-7. DOI: 10.1515/9783110365917.

[11] M. Boussé, O. Debals, and L. De Lathauwer, "A tensor-based method for large-scale blind source separation using segmentation," *IEEE Transactions on Signal*

Processing, vol. 65, no. 2, pp. 346–358, Jan. 2017. DOI: 10 . 1109 / TSP . 2016 . 2617858.

[12] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 586–600, Jun. 2015, ISSN: 1932-4553. DOI: 10 . 1109 / JSTSP.2015.2400415.

Stijn Hendrikx is currently pursuing a PhD at the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of the Electrical Engineering Department (ESAT) of KU Leuven, Belgium. He obtained his M.S. degree in mathematical engineering from the Faculty of Engineering, KU Leuven, Belgium, in 2019. His research concerns the computation of tensor decompositions from incomplete or implicit data. Contact him at Stijn.Hendrikx@kuleuven.be.

Raphaël Widdershoven is currently pursuing a PhD at the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of the Electrical Engineering Department (ESAT) of KU Leuven, Belgium. He obtained his M.S. degree in mathematical engineering from the Faculty of Engineering, KU Leuven, Belgium, in 2021. His research concerns the computation of constrained and coupled tensor decompositions from implicit data. Contact him at Raphael.Widdershoven@kuleuven.be.

Nico Vervliet is currently a postdoc at the STADIUS Center for Dynamical Systems, Signal Processing, and Data Analytics of the Electrical Engineering Department (ESAT) of KU Leuven, Belgium. He received the Ph.D. degree in Electrical Engineering from the Faculty of Engineering, KU Leuven, Belgium, in 2018. His research interests include numerical multilinear algebra, the optimization and application of tensor decompositions in signal processing, and data analysis, with a focus on big data and data fusion. He is the lead developer of Tensorlab. He has been awarded the Nokia Bell Scientific Award 2019 and the IBM Innovation Award 2019. He is a member of the IEEE and the IEEE Signal Processing Society and of SIAM. Contact him at Nico.Vervliet@kuleuven.be.

Lieven De Lathauwer received his Ph.D. degree from the Faculty of Engineering, KU Leuven, Belgium, in 1997, where he is currently a full professor. From 2000 to 2007, he was a research associate with CNRS-ETIS. He is an associate editor of SIAM Journal on Matrix Analysis and Applications and has served as an associate editor of IEEE Transactions on Signal Processing. He was a co-recipient of the IEEE Signal Processing Society Signal Processing Magazine best paper award in 2018 and a co-recipient of the IEEE Signal Processing Society Donald G. Fink overview paper award in 2023. His research concerns the development of tensor tools for engineering applications. He is a Fellow of IEEE, the Society for Industrial and Applied Mathematics (SIAM), and the European Association for Signal Processing (EURASIP). Contact him at Lieven.DeLathauwer@kuleuven.be.