FAST LEARNING OF FAST TRANSFORMS, WITH GUARANTEES

Quoc-Tung Le[*†], *Léon Zheng*[*†◇], *Elisa Riccietti*[†], *Rémi Gribonval*[†]

[†] Univ Lyon, ENS de Lyon, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
[◇] valeo.ai, Paris, France

ABSTRACT

Approximating a matrix by a product of few sparse factors whose supports possess the *butterfly* structure, which is common to many fast transforms, is key to learn fast transforms and speed up algorithms for inverse problems. We introduce a hierarchical approach that recursively factorizes the considered matrix into two factors. Using recent advances on the well-posedness and tractability of the two-factor *fixed-support* sparse matrix factorization problem, the proposed algorithm is endowed with *exact recovery guarantees*. Experiments show that speed and accuracy of the factorization can be jointly improved by several orders of magnitude, compared to gradient-based optimization methods.

Index Terms— Matrix factorization, Sparsity, Butterfly structure, Hierarchical factorization, NP-hardness.

1. INTRODUCTION

Approximating a matrix as a product of two or more sparse factors is a fundamental problem, serving as a principle or as an intermediate step in many tasks. It finds a wide range of applications in various domains at the interface of signal processing and machine learning [1, 2, 3, 4, 5, 6]. Given a matrix $\mathbf{Z}$ and a positive integer J, the general sparse matrix factorization (SMF) asks to find J sparse factors $\mathbf{X}^J, \ldots, \mathbf{X}^1$ such that $\mathbf{Z} \approx \mathbf{X}^J \ldots \mathbf{X}^1$ (where "..." denotes the product of several matrices $\mathbf{X}^i, 1 \leq i \leq J$). In general, the sparsity constraint is encoded by a *family* of allowed matrix supports, which describes the sparsity *pattern* enforced on the factors.

One classical example of such a family corresponds to matrices that are k-sparse either by row or by column, as typically considered in the classical sparse coding problem [1]. Indeed, SMF somehow generalizes sparse coding, where it has been widely documented [7] that finding the best approximation of a vector $\mathbf{z}$ by $\mathbf{D}\mathbf{x}$, with fixed dictionary $\mathbf{D}$ and k-sparse coefficient vector $\mathbf{x}$, is NP-hard. However, while the sparse coding problem becomes an easy least squares regression problem when the support of $\mathbf{x}$ is known, SMF *with fixed supports* was recently shown [8] to surprisingly remain NP-hard to approximate, even with only $J = 2$ factors.

Interestingly, SMF with fixed support and $J = 2$ becomes well-posed [9, 10] and tractable [8] under certain assumptions on the prescribed supports of the factors. These analyses have led to a factorization method for this setting that exploits non-trivially the Singular Value Decomposition (SVD) on blocks and computes an optimal factorization. This contrasts with previous works on SMF [11, 12, 5, 4], which express it as an optimization problem:

$$\underset{\mathbf{X}^J,\ldots,\mathbf{X}^1}{\text{Minimize}} \ \|\mathbf{Z} - \mathbf{X}^J \ldots \mathbf{X}^1\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm, with support / sparsity constraints on $\mathbf{X}^\ell$ ($\ell = 1, \ldots, J$). Such works typically address (1) with iterative first-order methods, leading to the usual difficulties related to the non-convexity of the problem: sensitivity to initialization and high dependence on hyper-parameters such as stepsizes or stopping criteria.

Our main contribution is an algorithmic framework to address Problem (1) when the fixed supports of the $J \geq 2$ factors have the butterfly structure [6, 4]. This structure appears ubiquitously in fast transforms such as the Discrete Fourier Transform (DFT) or the Hadamard Transform (HT), hence the algorithm allows to approximate a given matrix by a learned fast transform. The algorithm incorporates the two-factor, SVD-based, algorithm from [8] into a generalized version of a hierarchical greedy sparse factorization strategy [5, 10].

Based on the identifiability and recovery guarantees of [8, 9, 10], the following *exact recovery guarantee* is ensured: if $\mathbf{Z}$ admits an exact butterfly factorization (i.e, $\mathbf{Z} = \mathbf{X}^J \ldots \mathbf{X}^1$ and $\mathbf{X}^\ell$ is a butterfly factor), then the proposed algorithm recovers such factors up to natural scaling ambiguities. Using the factorization of the HT and DFT matrices as test cases, experiments show that this method presents important advantages over iterative first-order algorithms: reliability in finding an optimal solution, ease of use (due to the absence of hyper-parameters to be tuned), improved approximation precision and running time by more than one order of magnitude.

Section 2 formally introduces butterfly supports and existing theory on fixed-support SMF for $J = 2$. The proposed algorithmic framework is discussed in Section 3 together with its exact recovery guarantees. Section 4 is dedicated to the demonstration of the superiority of the proposed algorithm to factorize linear operators with a butterfly structure.

[*] The first two authors contributed equally. This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009.

2. BUTTERFLY STRUCTURE AND TWO-LAYER FIXED-SUPPORT SPARSE FACTORIZATION

After introducing the butterfly structure, this section gives useful results on fixed-support SMF with $J = 2$ factors, notably on identifiability and tractability, in preparation for the proposed hierarchical algorithmic framework.

2.1. Butterfly structure

Given a matrix $\mathbf{X}$, we denote $\mathrm{supp}(\mathbf{X})$ the support of $\mathbf{X}$, and we represent $\mathrm{supp}(\mathbf{X})$ by a binary matrix of the same size as $\mathbf{X}$ (1 for indices in the support and 0 otherwise). Denoting the Kronecker product by $\otimes$ and the identity matrix of size N by $\mathbf{I_N}$, a square matrix of size $N = 2^J$ is said to admit the butterfly structure if it is the product of J factors whose supports are $\mathbf{S}^J, \dots, \mathbf{S}^1$, where $\mathbf{S}^\ell := \mathbf{I_{N/2^\ell}} \otimes \left[\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right] \otimes \mathbf{I_{2^{\ell-1}}}$. These factors are sparse, as their structure permits at most 2 nonzero coefficients per row and per column.

The butterfly structure appears in many works [6, 4] as it is involved in the fast transform of several well-known linear transforms, such as the DFT, the HT, but also the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST) [13, 4]. For instance, the Hadamard matrix $\mathbf{H_N}$ of size $N = 2^J$ admits the factorization: $\mathbf{H_N} = \mathbf{F}^J \dots \mathbf{F}^1$ with $\mathbf{F}^\ell := \mathbf{I_{N/2^\ell}} \otimes \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right] \otimes \mathbf{I_{2^{\ell-1}}}$ for $1 \le \ell \le J$ [4].

2.2. Two-layer fixed-support SMF

Approximating a matrix by another one admitting the butterfly structure can be viewed as an instance of SMF with fixed support. Hence, we devote the rest of this section to the introduction of several tools developed for the fixed-support SMF problem with $J = 2$ factors, which serves as an intermediate step for the proposed method.

Given a subset of indices $\mathbf{S} \subseteq [\![n]\!] \times [\![m]\!]$, where $[\![p]\!] := \{1, \dots, p\}$ for $p \in \mathbb{N}$, the set of matrices with a sparsity pattern included in $\mathbf{S}$ will be denoted by $\Sigma_{\mathbf{S}} := \{\mathbf{M} \in \mathbb{C}^{n\times m} \mid \mathrm{supp}(\mathbf{M}) \subseteq \mathbf{S}\}$. In the following, $\mathbf{S}$ will be called by abuse of terminology a matrix support. Consider a matrix $\mathbf{Z} \in \mathbb{C}^{n\times m}$ and a pair of matrix supports $\mathbf{S}^L \subseteq [\![n]\!] \times [\![r]\!]$, $\mathbf{S}^R \subseteq [\![m]\!] \times [\![r]\!]$, the problem of two-layer fixed-support SMF is formulated as the following optimization problem[1]:

$$\underset{(\mathbf{X},\mathbf{Y})\in\Sigma_{\mathrm{S}}}{\text{Minimize}} \quad \|\mathbf{Z} - \mathbf{X}\mathbf{Y}^\intercal\|_F^2, \text{ with } \Sigma_{\mathrm{S}} := \Sigma_{\mathbf{S}^L} \times \Sigma_{\mathbf{S}^R}. \quad (2)$$

For general choices of matrix $\mathbf{Z}$ and pair of (left and right) supports $\mathrm{S} := (\mathbf{S}^L, \mathbf{S}^R)$, Problem (2) with arbitrary $\mathbf{S}^L, \mathbf{S}^R$ is shown to be NP-hard [8]. However, there are nontrivial conditions on the supports that ensure both tractability and identifiability of Problem (2), i.e., uniqueness of its solution up to natural scaling ambiguities.

[1] We consider $\mathbf{X}\mathbf{Y}^\intercal$ instead of $\mathbf{X}\mathbf{Y}$ for consistency with existing analysis [8, 9, 10] where it eases the notations without changing the problem.

Algorithm 1 Two-layer fixed-support matrix factorization

procedure FSMF($\mathbf{Z}$, $\mathbf{S}^L$, $\mathbf{S}^R$)
  Initialize $\mathbf{X} = \mathbf{0}, \mathbf{Y} = \mathbf{0}$.
  Denote $(\boldsymbol{\mathcal{S}}^i)_{i=1}^r := \varphi(\mathbf{S}^L, \mathbf{S}^R)$.
  Denote $\mathcal{P} = \{P_c\}_{c=1}^C$ the partition of $[\![r]\!]$ into equivalence classes of indices defined by: $i \sim j \iff \boldsymbol{\mathcal{S}}^i = \boldsymbol{\mathcal{S}}^j$.
  for $P := P_c \in \mathcal{P}$ **do**
    $I = \mathrm{rowsupp}(\boldsymbol{\mathcal{S}}^k)$, $J = \mathrm{colsupp}(\boldsymbol{\mathcal{S}}^k), k \in P$.
    Use the SVD of $\mathbf{Z}_{I,J}$ to find $(\mathbf{U}, \mathbf{V}) \in \arg\min\{\|\mathbf{Z}_{I,J} - \mathbf{U}\mathbf{V}^\intercal\|^2, \mathbf{U} \in \mathbb{C}^{|I|\times|P|}, \mathbf{V} \in \mathbb{C}^{|J|\times|P|}\}$.
    Assign $\mathbf{X}_{I,P} = \mathbf{U}$, $\mathbf{Y}_{J,P} = \mathbf{V}$.
  end for
  return $(\mathbf{X}, \mathbf{Y})$
end procedure

2.3. Identifiability

Identifiability of Problem (2) can be studied when $\mathbf{Z}$ admits an exact factorization $\mathbf{Z} = \mathbf{X}\mathbf{Y}^\intercal$ with $(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}$. Following the general framework introduced in [9], we represent a pair $(\mathbf{X}, \mathbf{Y})$ by its r-tuple of so-called *rank-one contributions*

$$\varphi(\mathbf{X}, \mathbf{Y}) := (\mathbf{X}_i \mathbf{Y}_i{}^\intercal)_{i=1}^r, \quad (3)$$

where $\mathbf{M}_i$ denotes the i-th column of matrix $\mathbf{M}$. The support constraint $\mathrm{S} = (\mathbf{S}^L, \mathbf{S}^R)$ is then represented by the r-tuple of rank-one supports $\mathcal{S} = (\boldsymbol{\mathcal{S}}^i)_{i=1}^r := \varphi(\mathbf{S}^L, \mathbf{S}^R)$, where the matrix supports $\mathbf{S}^L$, $\mathbf{S}^R$ are identified to their binary matrix representation. With this lifting approach [14, 15, 16], identifiability of the two factors $(\mathbf{X}, \mathbf{Y})$ is shown to be equivalent to identifiability of the rank-one contributions $\varphi(\mathbf{X}, \mathbf{Y})$ from their sum, except in trivial degenerate cases [9, 10].

Assume now that $\mathbf{Z}$ is a sum of rank-one contributions $\boldsymbol{\mathcal{C}}^i$: $\mathbf{Z} = \sum_{i=1}^r \boldsymbol{\mathcal{C}}^i$, with $\mathrm{supp}(\boldsymbol{\mathcal{C}}^i) \subseteq \boldsymbol{\mathcal{S}}^i$, $\mathrm{rank}(\boldsymbol{\mathcal{C}}^i) \le 1$, $i \in [\![r]\!]$. Then, when the rank-one supports $(\boldsymbol{\mathcal{S}}^i)_{i=1}^r$ are pairwise disjoint, the rank-one matrices $(\boldsymbol{\mathcal{C}}^1, \dots, \boldsymbol{\mathcal{C}}^r)$ are directly identified as the submatrices of $\mathbf{Z}$ restricted to $\boldsymbol{\mathcal{S}}^i$, $i \in [\![r]\!]$. This yields a simple sufficient condition for identifiability in exact matrix factorization with fixed support [9]. Despite its simplicity, this condition will play a crucial role to prove exact recovery guarantees of hierarchical multi-layer approaches.

2.4. Tractability and polynomial algorithm

We now present a sufficient condition for tractability of Problem (2), and a polynomial algorithm to solve it [8] even when $\mathbf{Z}$ does not admit an exact factorization. To introduce it, denote $\mathbf{Z}_{I,J} \in \mathbb{C}^{|I|\times|J|}$ as the submatrix of $\mathbf{Z}$ restricted to rows and columns indexed by $I \subseteq [\![n]\!]$ and $J \subseteq [\![m]\!]$ respectively. Denote also $\mathrm{colsupp}(\mathbf{M})$ as the subset of indices $i \in [\![r]\!]$ such that $\mathbf{M}_i$ is a nonzero column and $\mathrm{rowsupp}(\mathbf{M}) := \mathrm{colsupp}(\mathbf{M}^\intercal)$. The algorithm to address Problem (2) is described in Algorithm 1.

Theorem 1 ([8]). *Let $\varphi(\mathbf{S}^L, \mathbf{S}^R) := (\boldsymbol{\mathcal{S}}^i)_{i=1}^r$. If the supports $\boldsymbol{\mathcal{S}}^i$ ($i \in [\![r]\!]$) are pairwise disjoint or identical, then Algorithm 1 yields an optimal solution to Problem* (2).

The proof of Theorem 1 can be found in [8]. Its main idea is that if $\boldsymbol{\mathcal{S}}^i$ ($i \in [\![r]\!]$) are pairwise disjoint or identical, Problem (2) reduces to finding low rank approximations of submatrices of the target matrix $\mathbf{Z}$. A generalized algorithm with guarantees under more relaxed condition is also proposed in [8]. Interestingly, these tractable conditions also ensure nice properties to Problem (2): if a solution is locally optimal, it is also globally optimal, despite its non-convexity [8].

3. HIERARCHICAL METHOD FOR MULTI-LAYER BUTTERFLY FACTORIZATION

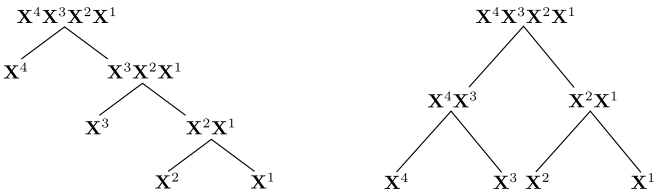
Going back to the SMF problem with $J \geq 2$ factors, we now introduce a heuristic method using Algorithm 1 to rapidly approximate a matrix as a product of *multiple* butterfly factors through a hierarchical paradigm [5, 10].

The method recursively uses Algorithm 1 to factorize a matrix into two factors at each intermediate level of the hierarchy. Denote the J butterfly supports of size 2^J by $(\mathbf{S}^J, \ldots, \mathbf{S}^1)$. For a given intermediate factor $\mathbf{H}$, at a certain level of the hierarchy characterized by some integers p and q, the intermediate step in the hierarchical factorization method consists of: (a) choosing an integer $p \leq \ell < q$; (b) applying Algorithm 1 to solve Problem (2) with matrix $\mathbf{H}$ and support constraints $\mathbf{S}^L := (\mathbf{S}^q \ldots \mathbf{S}^{\ell+1})$, $\mathbf{S}^R := (\mathbf{S}^\ell \ldots \mathbf{S}^p)^\top$. The obtained factors are written $(\mathbf{X}, \mathbf{Y}) := \mathrm{FSMF}(\mathbf{H}, \mathbf{S}^L, \mathbf{S}^R)$. This choice of $(\mathbf{S}^L, \mathbf{S}^R)$ here is well-defined since $\mathbf{S}^L$ and $\mathbf{S}^R$ are both binary matrices. Moreover, it has been shown that these supports have the following nice property:

Lemma 2 ([10]). *For any $1 \leq p \leq \ell < q \leq J$, the supports $\mathbf{S}^L := (\mathbf{S}^q \ldots \mathbf{S}^{\ell+1})$ and $\mathbf{S}^R := (\mathbf{S}^\ell \ldots \mathbf{S}^p)^\top$ are such that $\varphi(\mathbf{S}^L, \mathbf{S}^R)$ has disjoint rank-one supports.*

By Theorem 1, at each intermediate factorization, the pair $(\mathbf{X}, \mathbf{Y}) := \mathrm{FSMF}(\mathbf{H}, \mathbf{S}^L, \mathbf{S}^R)$ thus yields *an optimal solution* of Problem (2) with matrix $\mathbf{H}$.

There are many ways to recursively apply Algorithm 1 to factorize a matrix $\mathbf{Z}$ into J butterfly factors. For example, at the first level of the hierarchy, one can approximate $\mathbf{Z}$ by $\mathbf{X}\mathbf{Y}^\top$ with $(\mathbf{X}, \mathbf{Y}) = \mathrm{FSMF}\left(\mathbf{Z}, \mathbf{S}^J, (\mathbf{S}^{J-1} \ldots \mathbf{S}^1)^\top\right)$. Then, $\mathbf{Y}$ is similarly factorized into the products of $J-1$ factors with supports $\mathbf{S}^{J-1}, \ldots, \mathbf{S}^1$. Another way is to approximate at the first level the matrix $\mathbf{Z}$ by $\mathbf{X}\mathbf{Y}^\top$ with $(\mathbf{X}, \mathbf{Y}) = \mathrm{FSMF}\left(\mathbf{Z}, (\mathbf{S}^J \ldots \mathbf{S}^{\lfloor J/2 \rfloor + 1}), (\mathbf{S}^{\lfloor J/2 \rfloor} \ldots \mathbf{S}^1)^\top\right)$, where $\lfloor \cdot \rfloor$ denotes the floor function. After that, $\mathbf{X}$ (resp. $\mathbf{Y}$) will be factorized into $J - \lfloor J/2 \rfloor$ (resp. $\lfloor J/2 \rfloor$) factors constrained by the supports $\mathbf{S}^J, \ldots, \mathbf{S}^{\lfloor J/2 \rfloor + 1}$ (resp. $\mathbf{S}^{\lfloor J/2 \rfloor}, \ldots, \mathbf{S}^1$). In fact, each way of performing a hierarchical factorization corresponds to a tree. The two trees corresponding to the two schemes mentioned above are illustrated in Figure 1, and will be respectively referred to as *unbalanced* and *balanced* tree. As shown in [10], for any $\mathbf{Z}$ possessing the butterfly structure, our method guarantees the exact recovery of its butterfly factors, for any choice of tree. Hence, constraining the factors to butterfly supports avoids some pitfalls of the hierarchical paradigm with less structured sparsity constraints [17].



(a) Unbalanced (b) Balanced

Fig. 1: Two types of tree for hierarchical factorization, with $J = 4$ factors. A factorization into two factors is performed at each non-leaf node of the tree.

Theorem 3 (Exact recovery guarantee). *If a matrix $\mathbf{Z} = \mathbf{X}^J \ldots \mathbf{X}^1$ has the butterfly structure, then for any choice of tree, the hierarchical factorization will yield J butterfly factors $\bar{\mathbf{X}}^J, \ldots, \bar{\mathbf{X}}^1$ verifying $\mathbf{Z} = \bar{\mathbf{X}}^J \ldots \bar{\mathbf{X}}^1$, such that there exist invertible diagonal matrices $\mathbf{D}^J, \ldots, \mathbf{D}^0$ with $\mathbf{X}^\ell = \mathbf{D}^\ell \bar{\mathbf{X}}^\ell (\mathbf{D}^{\ell-1})^{-1}$, and the convention $\mathbf{D}^J = \mathbf{D}^0 = \mathbf{I}_{2^J}$.*

Proof sketch (details in [10]). The matrix to factorize is shown recursively to be shaped as $\mathbf{H} := \mathbf{D}'\mathbf{X}^q \ldots \mathbf{X}^p \mathbf{D}^{-1}$ with $\mathbf{D}, \mathbf{D}'$ diagonal invertible. By Lemma 2: a) the supports $\mathbf{S}^L := (\mathbf{S}^q \ldots \mathbf{S}^{\ell+1})$ and $\mathbf{S}^R := (\mathbf{S}^\ell \ldots \mathbf{S}^p)^\top$ satisfy the condition of Theorem 1, so $(\mathbf{X}, \mathbf{Y}) := \mathrm{FSMF}(\mathbf{H}, \mathbf{S}^L, \mathbf{S}^R)$ is an optimal solution of Problem (2) with matrix $\mathbf{H}$; and b) by the identifiability condition of Section 2.3, $(\mathbf{X}, \mathbf{Y})$ is equivalent to $\left(\mathbf{D}'\mathbf{X}^q \ldots \mathbf{X}^{\ell+1}, (\mathbf{X}^\ell \ldots \mathbf{X}^p \mathbf{D}^{-1})^\top\right)$ up to scaling. □

4. EXPERIMENTS

We now show experimentally the advantages of the proposed hierarchical factorization. All methods are implemented in Python and available in open source for reproducible research [18]. An implementation of the algorithm in C++ via Python and Matlab wrappers is also provided by the FAµST 3.25 toolbox at `https://faust.inria.fr/`[2].

4.1. Outperforming iterative optimization methods

In [4], an iterative optimization technique is introduced to approximate a matrix by a product $\mathbf{BP}$, where $\mathbf{B}$ is the product of butterfly factors and $\mathbf{P}$ is a permutation matrix, a model that covers both the DFT and HT matrix. The approach consists of two steps: first, a joint optimization of the butterfly

[2]The authors thank Hakim Hadj-Djilani for this implementation.

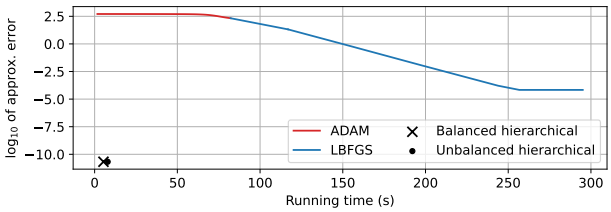


Fig. 2: Approximation error (in Frobenius norm) of the DFT matrix of size 512 with 9 factors: iterative method [4] vs. hierarchical method. Cumulated running times at each iteration are shown for the former, total running time for the latter.

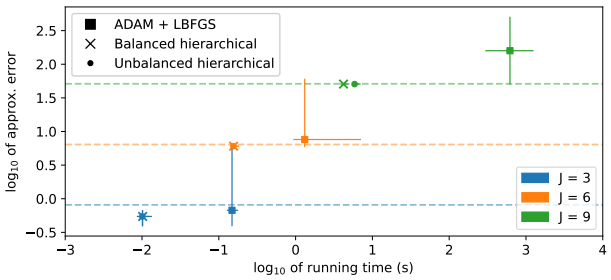


Fig. 3: Approximation error (in Frobenius norm) of a noisy DFT matrix of size 2^J: iterative method [4] vs. hierarchical method. The added noise matrix has i.i.d. complex centered Gaussian entries with variance $\sigma^2 = 0.005$ for both the real and imaginary part. Markers show the average running time / error. Error bars (almost invisible for the hierarchical approaches) show the extreme values over 10 different realizations of the noise matrix. As a baseline, horizontal dashed lines show the average of noise matrix norms.

factors and of a relaxed, continuous parametrization of $\mathbf{P}$ is performed using ADAM [19, Chapter 8], and a permutation matrix $\mathbf{P}$ is selected (ADAM is re-run several times with different random initializations if it fails to find a good $\mathbf{P}$); second, the selected $\mathbf{P}$ is kept fixed and the LBFGS algorithm [20, Chapter 7] is employed to fine-tune the butterfly factors.

Due to the structure of the relaxed optimization of the permutation of [4], the selected $\mathbf{P}$ turns out to be equivalent to the exploration of *eight* pre-chosen permutations. As an alternative, we apply our hierarchical factorization method to $\mathbf{Z}\mathbf{P}^\intercal$ for these eight permutations (since $\|\mathbf{Z} - \mathbf{B}\mathbf{P}\|_F^2 = \|\mathbf{Z}\mathbf{P}^\intercal - \mathbf{B}\|_F^2$ if $\mathbf{P}$ is a permutation matrix) and choose the one with the lowest error. In comparison to [4], this leads to a factorization method that is free of hyper-parameter tuning (step sizes, stopping criteria, etc.). We compare it to the iterative method of [4], run with 50 iterations of ADAM (we found out that performing more than 50 iterations yields similar precision), followed by 20 iterations of LBFGS (same setting as [4]), on the factorization of the DFT matrix of size 2^J with J butterfly factors, both in the exact and the noisy scenario.

For the noiseless DFT matrix, Figure 2 shows that the hierarchical approach, whether balanced or unbalanced, gives both better precision and much shorter running time, with gains of several orders of magnitude. Similar results were obtained with the HT matrix (not shown here).

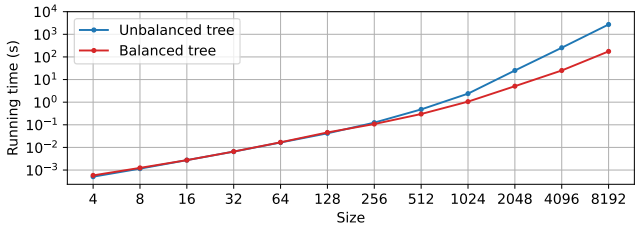


Fig. 4: Running time of the hierarchical method for a noisy matrix of size 2^J generated as a product of J random butterfly factors with i.i.d. standard Gaussian entries on their supports. The Gaussian noise matrix has entrywise variance $\sigma^2 = 0.01$.

We also compare the robustness of the approaches to noise. For the noisy DFT matrix, Figure 3 shows the instability of the iterative method [4], as it can give poor approximation of the noisy DFT matrix over several factorization experiments. In contrast, the proposed method is not only faster, but also finds more reliably a good approximation of the noisy DFT matrix, with an approximation error of the same order of magnitude as the norm of the noise matrix.

4.2. Balanced hierarchical factorization is faster

Finally, we compare the computational time of the balanced and unbalanced hierarchical method and show that both can learn other fast transforms than the DFT or HT, even in the noisy setting. The methods are run on a matrix $\mathbf{Z}$ of size 2^J generated as a noisy version of a product $\mathbf{B}$ of J random butterfly factors. For $J \geq 11$, Figure 4 shows that an order of magnitude in running time is gained with the balanced method compared to the unbalanced one. Moreover, for the parameters used in Figure 4 to generate $\mathbf{Z}$ and $\mathbf{B}$, we checked that the balanced method yields an approximation error similar to the norm of the noise matrix $\|\mathbf{Z} - \mathbf{B}\|_F$, while the unbalanced one yields an error twice larger than $\|\mathbf{Z} - \mathbf{B}\|_F$.

5. CONCLUSION

This paper proposes a new approach for sparse matrix factorization with butterfly supports. Besides being free from hyper-parameters, the method has exact recovery guarantees and significantly improves over existing first-order methods to recover classical linear operators, both in terms of speed and accuracy, even in the noisy setting. It can serve as building block to "project" any matrix onto the butterfly structure, with expected applications to many problems where it is desirable to replace dense matrices with learned fast transforms, e.g., in dictionary learning or neural network training.

6. REFERENCES

[1] I. Tošić and P. Frossard, "Dictionary learning," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[2] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double Sparsity: Learning Sparse Dictionaries for Sparse Signal Approximation," *IEEE Transactions on Signal Processing*, vol. 58, pp. 1553 – 1564, Apr. 2010.

[3] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling," *Proceedings of the IEEE*, vol. 98, pp. 1045 – 1057, Jul. 2010.

[4] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Re, "Learning Fast Algorithms for Linear Transforms Using Butterfly Factorizations," in *Proceedings of the 36th International Conference on Machine Learning*. June 2019, pp. 1517–1527, PMLR.

[5] L. Le Magoarou and R. Gribonval, "Flexible Multi-layer Sparse Approximations of Matrices and Applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 688–700, June 2016.

[6] L. Le Magoarou and R. Gribonval, "Chasing butterflies: In search of efficient dictionaries," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 3287–3291.

[7] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2013.

[8] Q.-T. Le, E. Riccietti, and R. Gribonval, "Spurious Valleys, Spurious Minima and NP-hardness of Sparse Matrix Factorization With Fixed Support," *arXiv preprint arXiv:2112.00386*, 2021.

[9] L. Zheng, E. Riccietti, and R. Gribonval, "Identifiability in Two-Layer Sparse Matrix Factorization," *arXiv preprint arXiv:2110.01235*, 2021.

[10] L. Zheng, E. Riccietti, and R. Gribonval, "Efficient Identification of Butterfly Sparse Matrix Factorizations," *arXiv preprint arXiv:2110.01230*, 2021.

[11] C.-J. Lin, "Projected Gradient Methods for Nonnegative Matrix Factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[12] N. Guan, D. Tao, Z. Luo, and B. Yuan, "NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2882–2898, June 2012.

[13] D. F. Elliott and K. R. Rao, *Fast Transforms Algorithms, Analyses, Applications*, Elsevier, 1983.

[14] S. Choudhary and U. Mitra, "Identifiability Scaling Laws in Bilinear Inverse Problems," *arXiv preprint arXiv:1402.2637*, 2014.

[15] L. Le Magoarou, *Matrices efficientes pour le traitement du signal et l'apprentissage automatique*, Ph.D. thesis, INSA de Rennes, 2016, Written in French.

[16] F. Malgouyres and J. Landsberg, "On the identifiability and stable recovery of deep/multi-layer structured matrix factorization," in *2016 IEEE Information Theory Workshop (ITW)*, 2016, pp. 315–319.

[17] Q.-T. Le and R. Gribonval, "Structured Support Exploration for Multilayer Sparse Matrix Factorization," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3245–3249.

[18] L. Zheng, Q.-T. Le, E. Riccietti, and R. Gribonval, "Code for reproducible research — Fast learning of fast transforms, with guarantees," Feb. 2022, code repository available at https://hal.inria.fr/hal-03552956, updates at https://github.com/leonzheng2/butterfly.

[19] Y. Goodfellow, I. J.and Bengio and A. Courville, *Deep Learning*, MIT Press, Cambridge, MA, USA, 2016.

[20] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, second edition, 2006.