DiCoDiLe: Distributed Convolutional Dictionary Learning

Thomas Moreau, Alexandre Gramfort

DiCoDiLe: Distributed Convolutional Dictionary Learning

Thomas Moreau and Alexandre Gramfort

Abstract—Convolutional dictionary learning (CDL) estimates shift invariant basis adapted to represent signals or images. CDL has proven useful for image denoising or inpainting, as well as for pattern discovery on multivariate signals. Contrarily to standard patch-based dictionary learning, patterns estimated by CDL can be positioned anywhere in signals or images. Optimization techniques consequently face the difficulty of working with extremely large inputs with millions of pixels or time samples. To address this optimization problem, we propose a distributed and asynchronous algorithm, employing locally greedy coordinate descent and a soft-locking mechanism that does not require a central server. Computation can be distributed on a number of workers which scales linearly with the size of the data. The parallel computation accelerates the parameter estimation and the distributed setting allows our algorithm be used with data that does not fit into a single computer's RAM. Experiments confirm the theoretical scaling properties of the algorithm, allowing us to learn patterns on images from the Hubble Space Telescope containing tens of millions of pixels.

Index Terms—Convolutional Dictionary Learning, Distributed System, Coordinate Descent.

1 Introduction

Convolutional sparse linear models consider that signals or images are obtained by linear combination of a few patterns localized in time or space. Patterns, which typically have a finite support, encode local variations such as edges in images or transient events in signals. The benefit of this model over traditional sparse coding technique based on local patches is that it is shift-invariant. While sparse coding encodes every local patch in the data, the convolutional linear model selects and encodes few patches leading to sparser and sharper reconstruction [3]. Over the last ten to fifteen years, these models have been used successfully for various signal and image processing applications. It was first proposed by Grosse et al. [14] for music recognition. It was then used extensively for denoising and inpainting of images [5, 17, 22, 28]. Beyond denoising or inpainting, recent applications used this shift-invariant model as a way to learn and localize relevant patterns in signals or images. Yellin et al. [29] used it to count the number of blood-cells present in holographic lens-free images. Jas et al. [15] and Dupré la Tour et al. [10] learned recurring patterns in univariate and multivariate signals from neuroscience. This model has also been used for denoising astronomical data [26] and localizing predefined patterns from space satellite images [9].

Convolutional dictionary learning (CDL) is the technique commonly employed to estimate the parameters of such shift-invariant models. The dictionary is here formed by the set of patterns (or atoms), whose positions in time or space are encoded by some sparse activation vectors. CDL then boils down to an optimization problem, where the variables are the dictionary as well as the activations. This problem is non-convex and approximate solutions can be found by using alternating minimization over each variable. The estimation of the activation – which is commonly referred to as convolutional sparse coding (CSC) – is the critical bottleneck when it comes to applying CDL to large data.

As CSC is a convex problem, classical optimization tools can be employed. Chalasani et al. [6] used an accelerated proximal gradient method based on Fast Iterative Soft-Thresholding Algorithm (FISTA; [2]). Bristow et al. [5] proposed the Fast Convolutional Sparse Coding (FCSC) method, a splitting algorithm based on Alternating Direction Method of Multipliers (ADMM; [12]). While it is possible to leverage the Fast Fourier Transform to reduce the complexity of these approaches [28], both algorithms require to compute Fourier transforms of the entire data at each iteration. The cost can therefore be prohibitive for large signals or images. To avoid such costly global operations, several works have proposed to use local computations [14, 17, 19, 22, 31]. A first line of work proposed by Kavukcuoglu et al. [17] uses greedy coordinate descent (GCD; [21]) with an efficient mechanism to compute the update value. In their seminal work, Grosse et al. [14] propose to adapt the Feature Sign Search algorithm to only process small window of the signal at a time. Similarly, Papyan et al. [22] proposed to solve the problem locally over smaller continuous chunks of signal and then aggregate these local results by solving a global optimization problem. In our recent work [19], we proposed to use locally greedy coordinate descent (LGCD). Based on updates similar to GCD, this method has a lower per-iteration complexity as it relies on local computation in the data. It does so like a patch-based technique would do, yet solving the non-local problem.

To further reduce the computational burden of CDL, recent studies consider parallelization strategies. Skau and Wohlberg [25] proposed a parallel method based on consensus ADMM. It does so by splitting the computation across the different atoms. This limits the number of cores that can be used to the number of atoms in the dictionary, and it still requires Fourier transforms on the entire data. Concurrently, we proposed an alternative parallel algorithm for CSC, called DICOD [19], that distributes GCD updates when working with one dimensional signals. DICOD could be used with multidimensional data such as images, yet only by splitting the data along one dimension, hence limiting its interest in this setting.

In the present paper, we propose a distributed solver for CDL, which can be used to learn recurring patterns in large multidimensional data. For the CSC step, we extend DICOD to use local updates with lower complexity while still working in the asynchronous setting without a central server. We detail a soft-lock mechanism, necessary to ensure the convergence of the algorithm with a grid of workers while remaining asyn-

- *T.Moreau and A. Gramfort are with Université Paris-Saclay, Inria, CEA, Palaiseau, 91120, France.*

chronous. Then, we explain how to also leverage the distributed set of workers to further reduce the cost of the dictionary update. This is done by precomputing in parallel sufficient statistics used in the gradient operator. Extensive numerical experiments confirm the agreement between the theoretical complexity of our algorithm and its running time, allowing us to learn patterns from an astronomical image formed by tens of millions of pixels. Code is made available to replicate our results.

In Section 2, we describe the convolutional linear model and the estimation of its parameters with CDL. Our distributed algorithm DiCoDiLe is introduced in Section 3. Section 4 presents results on simulations, signals and images.

Notation For a vector $U \in \mathbb{R}^P$ we denote $U_p \in \mathbb{R}$ its p^{th} coordinate. For a finite d-dimensional domain $\Omega = \prod_{i=1}^d [0, T_i[$, where $[0, T_i[$ are the integers from 0 to $T_i - 1$, $|\Omega| = \prod_{i=1}^d T_i$ is its size. We will denote $T = \prod_{i=1}^d T_i$ whenever the link between T and T_i is unambiguous. We denote $\mathcal{X}_\Omega^P$ (resp. $\mathcal{X}_\Omega^{P\times Q}$) the space of observations defined on Ω with values in $\mathbb{R}^P$ (resp. $\mathbb{R}^{P\times Q}$). For instance, $\mathcal{X}_\Omega^3$ with $d = 2$ is the space of RGB-images with height T_1 and width T_2. The value of an observation $X \in \mathcal{X}_\Omega^P$ at position $\omega \in \Omega$ is given by $X[\omega] = X[\omega_1, \dots, \omega_d] \in \mathbb{R}^P$, and its restriction to the p^{th} coordinate at each position is denoted $X_p \in \mathcal{X}_\Omega^1$. All signals are 0-padded, *i.e.* for $\omega \notin \Omega$, $X[\omega] = 0$. The convolution operator is denoted $*$ and is defined for $X \in \mathcal{X}_\Omega^P$ and $\boldsymbol{Y} \in \mathcal{X}_\Omega^{P\times Q}$ as the signal $X * \boldsymbol{Y} \in \mathcal{X}_\Omega^Q$ with

$$(X * \boldsymbol{Y})[\omega] = \sum_{\tau\in\Omega} X[\omega - \tau] \cdot \boldsymbol{Y}[\tau] \ , \quad \forall \omega \in \Omega \ . \tag{1}$$

For any signal $X \in \mathcal{X}_\Omega^P$, the reversed signal is defined as $X^\intercal[\omega] = X[T_1 - \omega_1, \dots, T_d - \omega_d]^\top$. Computing the convolution with $X^\intercal$ is equivalent to computing the cross-correlation with X. The p-norm is defined as $\|X\|_p = \left(\sum_{\omega\in\Omega} \|X[\omega]\|_p^p\right)^{1/p}$. We denote $\mathcal{F}$ the Discrete Fourier transform (DFT) of a signal and for a multivariate signal $X \in \mathcal{X}_\Omega^P$, $\widehat{X} = \mathcal{F}(X)$ denotes the DFT of each of its coordinates. Finally, we will also denote ST the soft-thresholding operator defined as

$$\mathrm{ST}(u, \lambda) = \mathrm{sign}(u) \max(|u| - \lambda, 0) \ .$$

2 Convolutional Dictionary Learning

In this section, we recall the convolutional linear model and the classical way to estimate its parameters by alternating minimization.

2.1 Convolutional Linear Model

For an observed $X \in \mathcal{X}_\Omega^P$, signal or image, the convolutional linear model reads

$$X = Z * \boldsymbol{D} + \xi \ , \tag{2}$$

with $\boldsymbol{D} \in \mathcal{X}_\Theta^{K\times P}$ a dictionary of K atoms $\boldsymbol{D}_k$ in dimension P defined on a sub-domain $\Theta = \prod_{i=1}^d [0, L_i[\subset \Omega$, also called support of the atom. The activations are denoted $Z \in \mathcal{X}_\Omega^K$ and $\xi \in \mathcal{X}_\Omega^P$ is an additive noise term which is typically considered white and Gaussian. The dictionary elements $\boldsymbol{D}_k$ are prototypical patterns present in the data. They typically have a much smaller support Θ than the full domain Ω (i.e. $L = |\Theta| \ll T$). The activations Z_k encode the localization of the pattern k in the data. A non-zero coefficient $Z_k[\omega]$ indicates the presence of the pattern $\boldsymbol{D}_k$ around the position ω. As patterns typically occur at few locations in each observation, the coding signal Z is considered to be sparse. The aim of this model is truly to separate the shape of the local variation in the observation – encoded in the dictionary – from their localization in space or time. Figure 1 illustrates the decomposition of a temporal signal with one pattern.

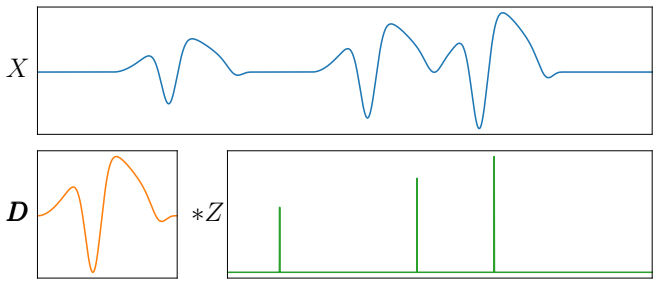


Fig. 1. Decomposition of a noiseless univariate signal X (*blue*) as the convolution $Z * \boldsymbol{D}$ between a temporal pattern $\boldsymbol{D}$ (*orange*) and a sparse activation signal Z (*green*).

The parameters of this model are estimated by solving the following CDL optimization problem:

$$\min_{\substack{Z,\boldsymbol{D}\\ \|\boldsymbol{D}_k\|\le 1}} \underbrace{\frac{1}{2}\|X - Z * \boldsymbol{D}\|_2^2}_{F(Z,\boldsymbol{D})} + \underbrace{\lambda\|Z\|_1}_{G(Z)} \ , \tag{3}$$

where the first term $F(Z, \boldsymbol{D})$ measures how well the model fits the data and the second term $G(Z)$ promotes sparse activations. The regularization parameter $\lambda > 0$ balances these two terms. The constraint $\|D_k\|_2 \le 1$ is used to avoid the scale ambiguity in our model due to the ℓ_1 constraint. Indeed, the ℓ_1-norm of Z can be made arbitrarily small by rescaling D by $\alpha \in \mathbb{R}_+^*$ and Z by $\frac{1}{\alpha}$. This problem is bi-convex, *i.e.* it is not jointly convex in $(Z, \boldsymbol{D})$ but it is convex in each of its coordinates. It is usually solved with an alternating minimization algorithm, updating at each iteration one of the block of coordinates, Z or $\boldsymbol{D}$. As each of the sub-problems are convex, the alternated descent is guaranteed to decrease the cost function, converging to a critical point of (3). It is however not guaranteed to be a global minimizer.

2.2 Convolutional Sparse Coding (CSC)

Given a dictionary of patterns $\boldsymbol{D}$, CSC aims to retrieve the sparse decomposition Z^* associated to the signal X. When minimizing only over Z, (3) becomes

$$Z^\star = \operatorname*{argmin}_Z \frac{1}{2}\|X - Z * \boldsymbol{D}\|_2^2 + \lambda\|Z\|_1 \ . \tag{4}$$

This boils down to a Lasso problem with a Toeplitz design matrix. Therefore, classical optimization techniques for the Lasso can be applied with the same convergence guarantees. In practice, these methods are adapted to leverage the convolutional structure of the problem, reducing the computational complexity to solve (4).

2.2.1 *Locally Greedy Coordinate Descent (LGCD)*

Coordinate descent (CD) is an algorithm which updates a single coefficient at each iteration. Following Moreau et al. [19], we use locally greedy coordinate selection to solve efficiently the CSC (4). Algorithm 1 summarizes the computations for LGCD.

Coordinate update For (4), it is possible to compute efficiently the optimal update for a given coordinate when all the others are fixed. Denoting q the iteration number, and $\widetilde{Z}_{k_0}^{(q)}[\omega_0]$ (atom

k_0 at position ω_0) the optimal value of the activation $Z_{k_0}^{(q)}[\omega_0]$, the update reads

$$\widetilde{Z}_{k_0}^{(q)}[\omega_0] = \frac{1}{\|\boldsymbol{D}_{k_0}\|_2^2}\text{ST}(\beta_{k_0}^{(q)}[\omega_0], \lambda) \ , \tag{5}$$

where $\beta^{(q)}$ is an auxiliary variable in $\mathcal{X}_\Omega^K$ defined for $(k, \omega) \in [1, K] \times \Omega$ as

$$\beta_k^{(q)}[\omega] = \left(\left(X - Z^{(q)} * \boldsymbol{D} + Z_k^{(q)}[\omega] e_\omega * \boldsymbol{D}_k\right) * \boldsymbol{D}_k^\intercal\right)[\omega] \ , \tag{6}$$

and where e_ω is a dirac (canonical vector) with value 1 in ω and 0 elsewhere. The value of the auxillary variable in (k_0, ω_0) corresponds to the gradient of F for a point $Z^{(q)}$ with the coordinate (k_0, ω_0) set to 0. To enable efficient coordinate update, an important property is that $\widetilde{Z}^{(q)}[\omega]$ only depends on a small neighborhood around ω.

Definition 2.1 (Θ-neighborhood)**.** *Let us consider a support $\Theta = \prod_{i=1}^d [0, L_i[$. For $\Omega_0 \in \Omega$, the Θ-neighborhood $\mathcal{V}_\Theta(\omega_0)$ is defined as*

$$\mathcal{V}_\Theta(\omega_0) = \prod_{i=1}^d [\omega_{0,i} - L_i + 1, \omega_{0,i} + L_i[\ .$$

Proposition 2.2 (Weak dependence)**.** *If the coordinate (k_0, ω_0) is updated in $Z^{(q)}$, then the value of $\widetilde{Z}^{(q+1)}$ differs from $\widetilde{Z}^{(q)}$ only for ω in a Θ-neighborhood $\mathcal{V}_\Theta(\omega_0)$ of ω_0.*

Proof. If the coordinate $Z_{k_0}[\omega_0]$ is updated with an additive update $\Delta Z_{k_0}[\omega_0] = \widetilde{Z}_{k_0}^{(q)}[\omega_0] - Z_{k_0}^{(q)}[\omega_0]$, Kavukcuoglu et al. [17] showed that $\beta^{(q+1)}$ can be obtained using the relation

$$\beta_k^{(q+1)}[\omega] = \beta_k^{(q)}[\omega] - (\boldsymbol{D}_{k_0} * \boldsymbol{D}_k^\intercal)[\omega - \omega_0] \Delta Z_{k_0}^{(q)}[\omega_0] \ , \tag{7}$$

for all $(k, \omega) \neq (k_0, \omega_0)$. As the support of $(\boldsymbol{D}_{k_0} * \boldsymbol{D}_k^\intercal)[\omega]$ is $\mathcal{V}_\Theta(0)$, this implies that the value $\beta_k^{(q+1)}$ only changes in the neighborhood $\mathcal{V}_\Theta(\omega_0) = \omega_0 + \mathcal{V}_\Theta(0)$. □

This proposition shows that the optimal value for a coordinate only depends on the current values of the activations in a neighborhood of size $|\mathcal{V}_\Theta(0)| = 2^d L$. This corresponds to twice the size of the dictionary in each dimension. Using this property, the computational complexity of updating a coordinate is $\mathcal{O}(2^d KL)$, as it is only necessary to update β and $\widetilde{Z}^{(q)}$ on $\mathcal{V}_\Theta(\omega_0)$.

Coordinate selection The selection of the updated coordinate $(k_0^{(q)}, \omega_0^{(q)})$ can follow different strategies. Following our work in [10, 19], we propose to select the coordinates with a locally greedy rule (LGCD).

Definition 2.3 (Disjoint and contiguous partition)**.** *Let us consider a domain Ω. For $M \in \mathbb{N}$, a M disjoint and contiguous partition $\{\mathcal{C}_m\}_{m=1}^M$ is a set of subsets of Ω such that*

$$\prod_{i=1}^d [\omega_i, \omega_i'[\subset \mathcal{C}_m \quad \textit{for} \quad \omega, \omega' \in \mathcal{C}_m \ ,$$
$$\bigcup_m \mathcal{C}_m = \Omega \ , \quad \mathcal{C}_m \cap \mathcal{C}_{m'} = \emptyset \ \textit{ for } \ m \neq m' \ .$$

At iteration q, the coordinate to update is selected greedily among the m-th sub-domain $\mathcal{C}_m$ with

$$(k_0^{(q)}, \omega_0^{(q)}) = \underset{(k,\omega) \in [1,K] \times \mathcal{C}_m}{\text{argmax}} \Big| \underbrace{\widetilde{Z}_k^{(q)}[\omega] - Z_k^{(q)}[\omega]}_{\Delta Z_k^{(q)}[\omega]} \Big| \ , \tag{8}$$

Algorithm 1 Locally Greedy Coordinate Descent

Require: $\boldsymbol{D}, X$, parameter $\epsilon > 0$, sub-domains $\mathcal{C}_m$,
Initialization: $\forall (k, \omega) \in [1, K] \times \Omega$,
$Z_k[\omega] = 0, \quad \beta_k[\omega] = \left(\boldsymbol{D}_k^\intercal * X\right)[\omega]$
repeat
  for $m = 1 \dots M$ **do**
    $\forall (k, \omega) \in [1, K] \times \mathcal{C}_m$,
    $$\widetilde{Z}_k^{(q)}[\omega] = \frac{1}{\|\boldsymbol{D}_k\|_2^2}\text{ST}(\beta_k^{(q)}[\omega], \lambda) \ ,$$
    Choose $(k_0, \omega_0) = \arg \max_{(k,\omega) \in [1,K] \times \mathcal{C}_m} |\Delta Z_k[\omega]|$
    Update β using (7) and $Z_{k_0}[\omega_0] \leftarrow \widetilde{Z}_{k_0}^{(q)}[\omega_0]$
  end for
until $\|\Delta Z^{(q)}\|_\infty < \epsilon$

with $m = q \mod M$. This strategy cycles through each of the M sub-domains and select the coordinate in $\mathcal{C}_m$ which is the farthest away from its optimal value when all the other coordinates are fixed. The quantity $\Delta Z_k[\omega]$ acts as a proxy for the reduction of the cost function obtained with the update of coordinate (k, ω) to the value $\widetilde{Z}_k^{(q)}[\omega]$ [21]. The complexity of the locally greedy selection rule is linear with the size of the sub-segments $\mathcal{O}(K|\mathcal{C}_m|)$. In the case where $M = KT$, this rule boils down to the cyclic CD [11] with a coordinate selection complexity in $\mathcal{O}(1)$, while setting $M = 1$ boils down to greedy CD [21] where the complexity of the selection is linear with the size of the signal $\mathcal{O}(KT)$. The selection of M in LGCD acts as a tradeoff between the computational complexity and the convergence speed of these two methods.

As it was shown in recent studies [16, 20], updating in priority the important coordinates by greedy selection leads to faster convergence rates, although it increases the cost of one iteration. The LGCD strategy proposes to use sub-domains $\mathcal{C}_m$ such that the computational complexities of selecting and updating the coordinate match. By using sub-domains of the $\mathcal{V}_\Theta(0)$'s size *i.e.* $|\mathcal{C}_m| = 2^d L$, the global iteration complexity of LGCD scales with 2^d times the size of the dictionary $\boldsymbol{D}$ *i.e.* $\mathcal{O}(2^d KL)$. By doing so LGCD achieves faster convergence rates than non-greedy coordinate selection rule while limiting the per-iteration cost.

Stopping criterion The stopping criterion used for this algorithm follows [21]. The algorithm is stopped once the magnitude of all possible coordinate updates are below a threshold $\epsilon > 0$ specified by the user, *i.e.*

$$\max_{(k,\omega) \in [1,K] \times \Omega} |\Delta Z_k^{(q)}[\omega]| < \epsilon \ . \tag{9}$$

2.2.2 Global solvers

Chalasani et al. [6] propose to solve (4) by proximal gradient methods exploiting the convolution in the gradient computation to save time. The core idea behind this method is to use a proximal splitting of the cost function between its smooth part $F(\cdot, \boldsymbol{D})$ and a proximable part G. Proximal gradient descent (ISTA) reads

$$Z^{(q+1)} = \text{prox}_{\gamma G}\left(Z^{(q+1)} - \gamma \nabla_Z F(Z^{(q+1)}, \boldsymbol{D})\right) \ , \tag{10}$$

where γ is the step size of the algorithm, usually chosen as the inverse of the smoothness constant of F, and $\text{prox}_{\gamma G}$ is the proximal operator of γG. For $G(\cdot) = \lambda \|\cdot\|_1$, this corresponds to the soft-thresholding operator $\text{ST}(\cdot, \gamma\lambda)$. To make this algorithm

efficient the key is to rely on the FFT for the gradient computations, as described in [28]. Doing the computation in the Fourier domain, the computational complexity of each iteration of this algorithm is $\mathcal{O}(KT \log T)$. It is also possible here to use an accelerated version FISTA to solve the problem [2, 6].

An alternative global algorithm proposed by Bristow et al. [5] is based on the ADMM algorithm [12]. The core idea in this line of work is to introduce an auxiliary variable and split the two part of the objective with an equivalent problem

$$\min_{\widehat{Y},Z} \frac{1}{2} \left\| \widehat{X} - \widehat{Y}\widehat{\boldsymbol{D}} \right\|_F^2 + \lambda \|Z\|_1 \quad \text{s.t. } \mathcal{F}(Z) = \widehat{Y} \ . \tag{11}$$

This problem is then solved using ADMM in its scaled form [4, Section 3.1.1]. The critical part in this algorithm is the minimization of the augmented Lagrangian relatively to Y which necessitates multiple FFT as well as the inversion of T linear systems of size $K \times K$. The overall complexity of this method is thus $\mathcal{O}(KT(\log T + K^2))$.

For these global solvers, the complexity of each iteration scales with the size of the data in $\mathcal{O}(T \log T)$ and the convergence rate is independent of T. In contrast, the iteration of LGCD are independent of T but the number of iterations needed to reach convergence scales with the number of non-zero coefficients which also depends on T. For large and sparse data, LGCD tends to be more efficient as the overall complexity depends linearly in T and not in $T \log T$. However, the choice between a local vs a global algorithm depends on the sparsity level of the desired solution, controlled by the value of the regularization parameter λ [10].

2.3 Dictionary Update

Given an activation signal $Z^{(q)}$, the dictionary is updated to improve how the model reconstructs the signal. The problem reads

$$\boldsymbol{D}^* = \underset{\|\boldsymbol{D}_k\|_2 \leq 1}{\operatorname{argmin}} \frac{1}{2} \left\| X - Z^{(q)} * \boldsymbol{D} \right\|_2^2 \ . \tag{12}$$

This problem is smooth and convex and can be solved using algorithms designed for smooth constrained optimization.

2.3.1 Projected Gradient Descent

The projected gradient descent (PGD; [7]) and its accelerated version (APGD) are possible strategies. These two algorithms are equivalent to ISTA and FISTA iterations where the proximal operator is a simple projection [7]. The efficiency of these two methods relies on pre-computations of sufficient statistics for fast evaluation of the gradient of the objective $\nabla_{\boldsymbol{D}} F$ [10]. The computations can be factorized as

$$\nabla_{\boldsymbol{D}} F(Z, \boldsymbol{D}) = Z^{\tilde{}} * (X - Z * \boldsymbol{D}) = \boldsymbol{\Psi} - \boldsymbol{\Phi} * \boldsymbol{D} \ , \tag{13}$$

where $\boldsymbol{\Phi} \in \mathcal{X}^{K \times K}_{\mathcal{V}_\Theta(0)}$ and $\boldsymbol{\Psi} \in \mathcal{X}^{K \times P}_{\Theta}$ are respectively defined as the convolution $Z^{\tilde{}} * Z$ restricted to $\mathcal{V}_\Theta(0)$ and the convolution $Z^{\tilde{}} * X$ restricted to Θ, *i.e.*

$$\boldsymbol{\Phi}[\omega] = Z^{\tilde{}} * Z = \sum_{\tau \in \Omega} Z[\omega + \tau] Z^\top[\tau] \qquad \text{for } \omega \in \mathcal{V}_\Theta(0) \ , \tag{14}$$

$$\boldsymbol{\Psi}[\omega] = Z^{\tilde{}} * X = \sum_{\tau \in \Omega} Z[\omega + \tau] X^\top[\tau] \qquad \text{for } \omega \in \Theta \ . \tag{15}$$

The complexity to obtain the two constants $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ scales as $\mathcal{O}(K(K+P)LT)$. This is comparable with the cost of computing the gradient directly $\mathcal{O}(KPT \log T)$. However, once the two constants are pre-computed, the gradient of F is obtained using (13) with complexity $\mathcal{O}(2^d K^2 P L \log(L))$. This is independent of the data size T, making each iteration efficient.

2.3.2 Related solvers

The block coordinate descent for dictionary learning [18], which consists in updating one atom at a time, can also be extended to the convolutional case. As PGD for full dictionary updates, this technique benefits from sufficient statistics pre-computations and the complexity is similar. Another possibility is to rely on ADMM to update the dictionary [5, 28]. However, ADMM algorithms can be slow to reach a convergence, yielding sub-optimal reconstruction performances as the feasible point is not directly optimized but results from a consensus. Yellin et al. [29] proposed to adapt the K-SVD dictionary update method [1] to the convolutional setting but this technique is adapted to an ℓ_0 penalty for the activation Z. It also requires to compute costly singular value decompositions (SVD).

2.4 Choice of initialization and regularization parameter

In practice, the quality of the solution obtained by CDL is impacted by the choice of the regularization parameter λ. Also, due to the iterative nature of the solvers, the initialization of the dictionary $\boldsymbol{D}^0$ can play an important role.

Regularization parameter selection

The choice of a regularization parameter λ in the context of dictionary learning is often selected by cross-validation on a subsequent task such as denoising or classification. However in the context of purely unsupervised learning, *e.g.* matrix factorization, this is still an open problem. This problem has recently received some attention using Bayesian methods [23, 27]. For the specific case of CDL using an ℓ_1 norm, it is useful to note that there exists a value $\lambda_{\max}$ such that for $\lambda \geq \lambda_{\max}$, 0 is solution of (4). The first order condition for (4) reads

$$\nabla_Z F(0, \boldsymbol{D}^{(0)}) = X * \boldsymbol{D}^{(0)\tilde{}} \in \partial G(0) = [-\lambda, \lambda]^{KT} \ . \tag{16}$$

This condition is always verified if $\lambda > \lambda_{\max} = \|X * \boldsymbol{D}^{(0)\tilde{}}\|_\infty$. In the following, the regularization parameter λ is chosen as a fraction of $\lambda_{\max}$. Using this data dependent value, the ratio $\lambda / \lambda_{\max}$ is chosen in $[0, 1]$, making the definition of the regularization parameter common to all input signals X.

Dictionary initialization

The standard strategy to initialize $\boldsymbol{D}^{(0)}$ for CDL methods is to generate random atoms with Gaussian white noise. However, as these atoms generally poorly correlate with the data, the initial value of $\lambda_{\max}$ is low compared to the following ones, leading to dense activation Z. For dictionary learning, a classical way to initialize the dictionary is to take samples from the input data [18]. In the convolutional setting, this boils down to taking random patches of the size of the dictionary directly from the data. This can be theoretically motivated by the work of Zhang et al. [30] that demonstrate improvements in pattern recovery under suitable assumptions.

3 Distributed Convolutional Dictionary Learning (DiCoDiLe)

In this section, we introduce DiCoDiLe, a distributed algorithm for convolutional sparse coding. Its steps are summarized in Algorithm 2. The CSC part of this algorithm will be referred to as DiCoDiLe$_Z$. It extends our previous work DICOD [19] by handling multidimensional data, in particular images. The algorithm proposes to distribute the computation on a grid of workers. To guarantee convergence with multidimensional data in an asynchronous setting, we introduce the notion of

Algorithm 2 DiCoDiLe with W workers

Require: $\boldsymbol{D}^{(0)}, X$, stopping criterion ν.
1: **repeat**
2: Compute $Z^{(q+1)}$ with DiCoDiLe$_Z(X, \boldsymbol{D}^{(q)}, W)$.
3: Compute ϕ and ψ with (14) and W workers.
4: Update $\boldsymbol{D}^{(q+1)}$ with PGD and Armijo backtracking line-search.
5: **until** Cost variation is smaller than ν

soft-locks described in Subsection 3.1. Details on how the dictionary updates can also benefit from the distributed setting are presented in Subsection 3.2.

3.1 DiCoDiLe$_Z$: Distributed CSC with coordinate descent

In the convolutional setting, coordinates that are far enough – relatively to the size of the dictionary – are only weakly dependent (Proposition 2.2). It is thus natural to distribute the computation of CD by splitting the data in continuous sub-domains [19].

The DiCoDiLe$_Z$ algorithm, described in Algorithm 3, is distributed over W workers which update asynchronously the coordinates of Z. We consider $\{\mathcal{S}_w\}_{w=1}^W$, a disjoint and contiguous partition of Ω. Each worker $w \in [1, W]$ is in charge of updating the coordinates of the sub-domain $\mathcal{S}_w$. These sub-domains are also partitioned with disjoint and contiguous sub-domains $\mathcal{C}_m^{(w)}$ of size $|\mathcal{V}_\Theta(0)|$. Then, the worker w chooses at each iteration q an update candidate (k_0, ω_0) which corresponds to a greedy update in one of its sub-domain $\mathcal{C}_m^{(w)}$, *i.e.*

$$(k_0, \omega_0) = \underset{(k,\omega)\in[1,K]\times\mathcal{C}_m^{(w)}}{\operatorname{argmax}} |\Delta Z_k[\omega]| \ , \tag{17}$$

with $m = q \mod M$. The sub-domains $\mathcal{C}_m^{(w)}$ are selected in a cyclic order but empirical test showed no significant differences for random sub-domain selection. Then, this candidate update is accepted or rejected using the soft-lock mechanism described in Subsection 3.1.1. If the candidate is rejected, the worker moves on to the next sub-domain $\mathcal{C}_{m+1}^w$ without updating a coordinate. If the candidate is accepted, the value of the coordinate $Z_{k_0}[\omega_0]$ is replaced by $\widetilde{Z}_{k_0}^{(q)}[\omega_0]$ and β is updated with (7). To ensure the convergence, it is sufficient to notify the neighboring workers if a local update changes the value of β outside of $\mathcal{S}_w$, *i.e.* if $\mathcal{V}_\Theta(\omega_0) \not\subset \mathcal{S}_w$.

Definition 3.1 (Θ-Border)**.** *Let us consider a sub-domain, $\mathcal{S}_w = \prod_{i=1}^d [l_i, u_i[$, where (l_i, u_i) are the lower and upper limits of this sub-domain for dimension i. For $\Theta = \prod_{i=1}^d [0, L_i[$, the Θ-border is defined as the set of points $\omega \in \mathcal{S}_w$ such that $\mathcal{V}_\Theta(\omega) \not\subset \mathcal{S}_w$*, i.e.

$$\mathcal{B}_\Theta(\mathcal{S}_w) = \prod_{i=1}^d [l_i, l_i + L_i[\cup [u_i - L_i, u_i[\ . \tag{18}$$

In the case where the updated coordinate (k, ω_0) is in $\mathcal{S}_w \setminus \mathcal{B}_\Theta(\mathcal{S}_w)$, then $\mathcal{V}_\Theta(\omega_0)$ is included in the set of coordinates $\mathcal{S}_w$ updated by the worker w. In this case, the update is not at the border of the sub-domain, so the values of β and $\widetilde{Z}^{(q)}$ need no update for the other workers w'. Thus, there is no need to exchange information between the workers. However, for $\omega_0 \in \mathcal{B}_\Theta(\mathcal{S}_w)$, it is necessary to notify other workers w' whose domain overlap with the Θ-neighborhood of ω_0 *i.e.* $\mathcal{V}_\Theta(\omega_0) \cap \mathcal{S}_{w'} \neq \emptyset$. The notification can be done by sending the triplet $(k_0, \omega_0, \Delta Z_{k_0}[\omega_0])$ to these workers. These information are sufficient to keep β up to date for all workers using (7). Figure 2 illustrates this communication process on a 2D example.

Algorithm 3 DiCoDiLe$_Z$ with W workers

Input: $\boldsymbol{D}, X$, parameter $\epsilon > 0$, sub-domains $\mathcal{S}_w$,
In parallel for $w = 1 \cdots W$
Compute a partition $\{\mathcal{C}_m^{(w)}\}_{m=1}^M$ of the worker domain $\mathcal{S}_w$ with sub-domains of size $2^d L$.
Initialize $\beta_k[\omega]$ and $Z_k[\omega] \ \ \forall (k, \omega) \in [1, K] \times \mathcal{S}_w$,
repeat
  for m=1...M **do**
    Receive messages and update Z and β with (7)
    Choose $(k_0, \omega_0) = \underset{(k,\omega)\in[1,K]\times\mathcal{C}_m^{(w)}}{\operatorname{argmax}} |\Delta Z_k[\omega]|$
    if $|\Delta Z_{k_0}[\omega_0]| < \max_{(k,\omega)\in[1,K]\times\mathcal{V}_\Theta(\omega_0)} |\Delta Z_k[\omega]$ **then**
      The coordinate is soft-locked, **goto 6**
    end if
    Update β with (7) and $Z_{k_0}[\omega_0] \leftarrow \widetilde{Z}_{k_0}^{(q)}[\omega_0]$
    if $\omega_0 \in \mathcal{B}_{2L}(\mathcal{S}_w)$ **then**
      Send $(k_0, \omega_0, \Delta Z_{k_0}[\omega_0])$ to neighbors
    end if
  end for
until global convergence $\|\Delta Z\|_\infty < \epsilon$

With this distributed algorithm, the inter-processes communications are minimal, as only $d+2$ values need to be exchanged. Moreover, as it does not rely on centralized communication, the necessary bandwidth between the workers is also reduced as a worker only communicates with its direct neighbors. This contrasts with other distributed algorithms for CDL such as [25] that requires centralized communication of full support values between the workers. Finally, the communication only occurs when $\omega_0 \in \mathcal{B}_\Theta(\mathcal{S}_w)$, which happens rarely especially when working with large data and patterns of small support. This reduces even further the need of bandwidth between workers.

3.1.1 Interferences and Soft-Locks in asynchronous setting

While it is now clear how communication should be done between workers, one need to clarify how to proceed to ensure convergence in the case of asynchronous updates. When W coordinates $(k_w, \omega_w)_{w=1}^W$ of Z are updated simultaneously by respectively $\Delta Z_{k_w}[\omega_w]$, the updates might not be independent. The local version of β used for the update does not account for the other updates. The cost difference resulting from all these simultaneous updates is denoted ΔE, and the cost reduction induced by only one update w is denoted $\Delta E_{k_w}[\omega_w]$. Simple derivations, detailed in Proposition A.2, show that

$$\Delta E = \overbrace{\sum_{i=1}^{W} \Delta E_{k_w}[\omega_w]}^{\text{iterative steps}} - \underbrace{\sum_{w\neq w'} (\boldsymbol{D}_{k_w} * \boldsymbol{D}^{\tilde{\gamma}}_{k_{w'}})[\omega_{w'} - \omega_w] \Delta Z_{k_w}[\omega_w] \Delta Z_{k_{w'}}[\omega_{w'}])}_{\text{interference}} \ , \tag{19}$$

If for all w, all other updates w' are such that if $\omega_{w'} \notin \mathcal{V}_\Theta(\omega_w)$, then $(\boldsymbol{D}_{k_w} * \boldsymbol{D}^{\tilde{\gamma}}_{k_{w'}})[\omega_{w'} - \omega_w] = 0$. The updates can be considered to be sequential as the interference term is zero. This also directly results from Proposition 2.2. When $\left|\{\omega_w\}_w \cap \mathcal{V}_\Theta(\omega_0)\right| = I_0 > 1$, the interference term does not vanish. Moreau et al. [19] show that if $I_0 < 3$, then, under mild assumption, the interference term can be controlled and DICOD converges. While this is sufficient for 1D partitioning as I_0 cannot be larger than 2 (if $\mathcal{S}_w$ is larger than $\mathcal{V}_\Theta(0)$), this analysis cannot be extended to $I_0 \geq 3$. This limits the partitioning of Ω along one single dimension.

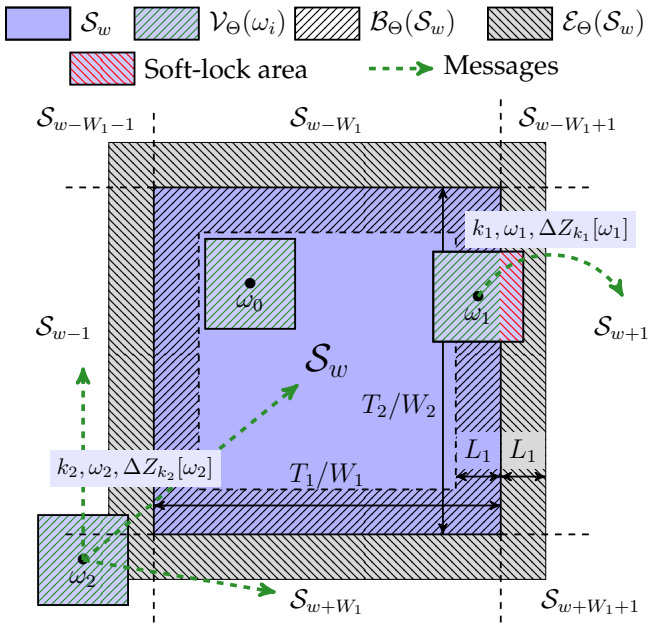


Fig. 2. Communication process in DiCoDiLe$_Z$ with $d = 2$ and 9 workers centered around worker w. The update in ω_0 is independent of the other workers. The update in ω_1 is performed if no better coordinate update is possible in the soft-lock area $\mathcal{V}_\Theta(\omega_1) \cap \mathcal{S}_{w+1}$ (*red hatched area*). If accepted, the worker $w + 1$ needs to be notified. The update in ω_2 changes the value of the optimal updates in the Θ-extension of the other workers sub-domains. Thus it needs to notify all the neighboring workers.

To avoid this limitation, we propose the soft-lock, a novel mechanism to avoid interfering updates of higher order. The idea is to avoid updating concurrent coordinates simultaneously. A classical tool in computer science to address such concurrency issue is to rely on a synchronization primitive, called *lock*. This primitive can ensure that at most one worker enters a protected block of code simultaneously. In our case, it would be possible to use this primitive to restrict updates in $\mathcal{B}_\Theta(\mathcal{S}_w)$ only when none of its neighbors is updating $\mathcal{B}_\Theta(\mathcal{S}_{w'})$. This would avoid interfering updates. However, the drawback of this method is that it prevents the algorithm to be asynchronous, as typical lock implementations rely on centralized communications. Instead, we propose to use an implicit locking mechanism which keeps our algorithm asynchronous.

Definition 3.2 (Θ-extension)**.** *We consider $\mathcal{S}_w = \prod_{i=1}^d [l_i, u_i[$ a sub-domain of Ω. For $\Theta = \prod_{i=1}^d [0, L_i[$, the Θ-extension of $\mathcal{S}_w$ is defined as $\mathcal{E}_\Theta(\mathcal{S}_w) = (\mathcal{S}_w + \mathcal{V}_\Theta(0)) \setminus \mathcal{S}_w$,* i.e.

$$\mathcal{E}_\Theta(\mathcal{S}_w) = \prod_{i=1}^d [l_i - L_i, l_i[\cup [u_i, u_i + L_i[\ . \tag{20}$$

The idea of the soft-lock is the following. Additionally to the sub-domain $\mathcal{S}_w$, each worker also maintains the value of $\widetilde{Z}^{(q)}$ and β on the Θ-extension $\mathcal{E}_\Theta(\mathcal{S}_w)$ (See Figure 2). At each iteration, the worker w considers a candidate coordinate (k_0, ω_0) to update from (17). If $\omega_0 \notin \mathcal{B}_\Theta(\mathcal{S}_w)$, the coordinate is updated like in LGCD. When $\omega_0 \in \mathcal{B}_\Theta(\mathcal{S}_w)$, the candidate is accepted if there is no better coordinate update in $\mathcal{V}_\Theta(\omega_0) \cap \mathcal{E}_\Theta(\mathcal{S}_w)$, *i.e.* if

$$|\Delta Z_{k_0}[\omega_0]| > \max_{k,\omega \in [1,K] \times \mathcal{V}_\Theta(\omega_0) \cap \mathcal{E}_\Theta(\mathcal{S}_w)} |\Delta Z_k[\omega]| \ . \tag{21}$$

In case of equality, the update in the sub-domain $\mathcal{S}_{w'}$ with the minimal index w' is preferred. Using this mechanism effectively prevents any interfering update. If two workers $w < w'$ have update candidates ω_0 and ω_0' such that $\omega_0' \in \mathcal{V}_\Theta(\omega_0)$, then with (21), only one of the two updates will be accepted. It will be either the largest one if $|\Delta Z_{k_0}[\omega_0]| \neq |\Delta Z_{k_0'}[\omega_0']|$ or the one from w if both updates have the same magnitude. The other update will be rejected and there will be no interference. By doing so this mechanism is fully asynchronous. The updates on the borders are implicitly ordered, preventing interference. The update are guaranteed to be performed as if they were run sequentially, ensuring convergence of the algorithm. This requires to increase slightly the communication between the workers and to double the size of the Θ-border in each direction. This is however not a bottleneck as demonstrated in the following experiments.

3.1.2 Speed-up analysis of DiCoDiLe$_Z$

In this section, we consider the speed-up provided by this distributed algorithm. As each worker runs LGCD locally, the computational complexity of a local iteration in a worker w is $\mathcal{O}(2^d KL)$. This complexity is independent of the number of workers used to run the algorithm. Thus, the speed-up obtained is equal to the number of updates that are performed in parallel. If we consider W update candidates (k_w, ω_w) chosen by the workers, the number of updates that are performed corresponds to the number of updates that are not soft-locked. When the updates are uniformly distributed on $\mathcal{S}_w$, the probability that an update candidate located in $\omega \in \mathcal{S}_w$ is not soft-locked can be lower bounded by

$$P(\omega \notin \textbf{SL}) \geq \prod_{i=1}^d \left(1 - \frac{W_i L_i}{T_i}\right) \tag{22}$$

Computations are detailed in Proposition B.1. When the ratios $\frac{T_i}{W_i L_i}$ are large for all i, *i.e.* when the size of the workers sub-domains $|\mathcal{S}_w|$ are large compared to the dictionary size L, then $P(\omega \notin \textbf{SL}) \simeq 1$. In this case, the expected number of accepted update candidates is close to W and DiCoDiLe$_Z$ scales almost linearly. When W_i reaches $\frac{T_i}{L_i}(\frac{2^{1/d}}{2^{1/d}-1})$, then $P(\omega \notin \textbf{SL}) \gtrsim \frac{1}{2}$, and the acceleration is reduced to $\frac{W}{2}$.

In comparison, we report in [19] a super-linear speed-up for DICOD on 1D signals. This is due to the fact that we are using GCD locally in each worker. GCD has a very high iteration complexity when W is small, as it scales linearly with the size of the considered sub-domain. Thus, when sub-dividing the signal between more workers, the iteration complexity is decreasing and more updates are performed in parallel. This explains why the speed-up is almost quadratic for low W. As the complexity of iterative GCD are worse than LGCD, DiCoDiLe$_Z$ has better performance than DICOD in low W regime. Furthermore, in the 1D setting, DICOD becomes similar to DiCoDiLe$_Z$ once the size of the workers sub-domain becomes too small. This happens when the local domain of LGCD becomes comparable to the size of the sub-domains $\mathcal{S}_w$. This implies that DiCoDiLe$_Z$ always outperforms DICOD.

3.1.3 Stopping criterion

In this distributed setting, it is non-trivial to know when the algorithm reaches convergence. To match the sequential stopping criterion of GCD (9), the convergence is considered to be reached once no worker can perform a coordinate update $|\Delta Z^{(q)}|$ that is higher than $\epsilon > 0$. However, as a worker can be affected by its neighboring workers, the stopping criterion of CD cannot be applied independently in each worker. Once a worker reaches this state, the potential update can still increase above ϵ because of updates in $\mathcal{E}_\Theta(\mathcal{S}_w)$ by another worker. To make sure the correct criterion is achieved, workers that

reach the local criterion are paused. They are restarted when a neighboring workers triggers an update on $\mathcal{E}_\Theta(\mathcal{S}_w)$. The global convergence can be decided when all workers are paused. While this process is based on centralized communication, it can still be performed asynchronously using message passing to a designated master process that will notify the end of the algorithm to all workers.

3.2 Dictionary updates in a distributed setting

For the dictionary, a change in a coordinate in Z at any point in Ω can impact the solution of (12). Moreover, the atoms are interdependent of each other. It is thus complicated to parallelize the dictionary update as it is centralized by nature and there is no natural division of the computation. However, the size of $\boldsymbol{D}$ is way smaller than Z and (12) can be solved efficiently when applying the PGD algorithm described in Subsection 2.3. The main computational bottleneck is the computation of the constants $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$. For these, it is possible to leverage the distributed setting as the computation are separable between the different workers. The computations can be made embarrassingly parallel among the workers as

$$\boldsymbol{\Phi}[\omega] = \sum_{w=1}^{W} \sum_{\tau \in \mathcal{S}_w} Z[\tau] Z^\top[\tau + \omega] \qquad \text{for } \omega \in \mathcal{V}_\Theta(0) \ , \tag{23}$$

$$\boldsymbol{\Psi}[\omega] = \sum_{w=1}^{W} \sum_{\tau \in \mathcal{S}_w} Z[\tau] X^\top[\tau + \omega] \qquad \text{for } \omega \in \Theta \ . \tag{24}$$

For all $(\omega, \tau) \in \mathcal{V}_\Theta(0) \times \mathcal{S}_w$, we have $\omega + \tau \in \mathcal{S}_w \cup \mathcal{E}_\Theta(\mathcal{S}_w)$. Thus, the computations of the inner sum $\boldsymbol{\Phi}^w$ can be made independently by each worker. These partial results are gathered and reduced with $\boldsymbol{\Phi} = \sum_{w=1}^{W} \boldsymbol{\Phi}^w$. By making use of these distributed computation, $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ can be computed with respectively $\mathcal{O}(K^2 |\mathcal{S}_w| \log(|\mathcal{S}_w \cup \mathcal{E}_\Theta(\mathcal{S}_w)|))$ and $\mathcal{O}(KP |\mathcal{S}_w| \log(|\mathcal{S}_w \cup \mathcal{E}_\Theta(\mathcal{S}_w)|))$ operations. The values of $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ are sufficient statistics to compute $\nabla_{\boldsymbol{D}} F$ and F. The dictionary can thus be updated without having to gather the full activation Z on one machine. This allows to learn a dictionary for a signal X that does not fit in memory, as each worker only needs to store a part of the full signal and the dictionary updates only require storing these sufficient statistics.

4 Numerical Experiments

The numerical experiments are performed on a SLURM cluster with 30 nodes. Each node has 20 physical cores, 40 logical cores and 250 GB of RAM. A worker is typically one logical core on one of the machine. In practice we use up to 400 workers. DiCoDiLe is implemented in Python [24] using `mpi4py` [8][1].

Experiments make use of simulated one dimensional signals, generated following the sparse convolutional linear model (2) with $d = 1$ and $P = 7$. The dictionary is composed of $K = 25$ atoms $\boldsymbol{D}_k$ of length $L = 250$. Each atom is sampled from a standard Gaussian distribution and then normalized. The sparse code entries are drawn from a Bernoulli-Gaussian distribution, with Bernoulli parameter $\rho = 0.007$, mean 0 and standard variation 10. The noise term ξ is sampled from a standard Gaussian distribution with variance 1. The length of the signals X is denoted T.

For experiments with images, we will use the standard colored image Mandrill ($P = 3$) at full resolution (512×512). Finally, for large scale experiments, we used an image of the Great Observatories Origins Deep Survey (GOODS) South field, acquired by the Hubble Space Telescope [13]. We used the STScI-H-2016-39 image[2] with resolution 6000×3600.

In all experiments, we used $\lambda = 0.1 \lambda_{\max}$ as a default regularization parameter, unless specified otherwise.

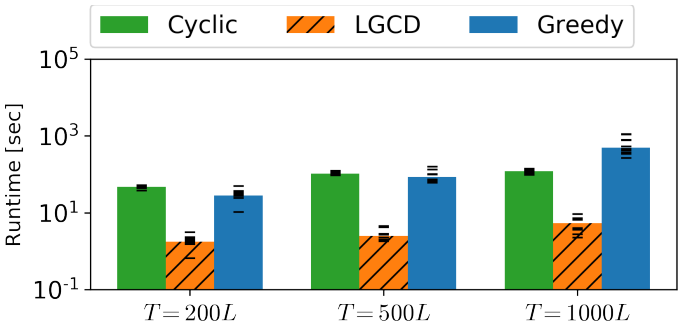


Fig. 3. Average running time of CD with four coordinate selection schemes – (*blue*) Locally Greedy, (*green*) Cyclic, (*orange*) Randomized and (*red*) Greedy – for two signal lengths and $\lambda = 0.5\lambda_{\max}$. LGCD consistently outperforms the two other strategies.

4.1 Performance of distributed sparse coding

The performances of the distributed sparse coding step in DiCoDiLe is impacted by the choice of coordinate descent strategy, as well as soft-lock mechanism and the data partitioning scheme. The following experiments discuss their effects on the scalability on DiCoDiLe$_Z$.

4.1.1 *Coordinate Selection Strategy*

Using one worker, Figure 3 compares the average running times of three CD strategies to solve (4): Cyclic, Greedy and LGCD. LGCD consistently outperforms the other strategies for both $T = 150L$ and $T = 750L$. For LGCD and Cyclic, the iteration complexity is constant compared to the size of the signal, whereas the complexity of each Greedy iteration scales linearly with T. This makes running time of Greedy explode when T grows. Besides, as LGCD also prioritizes the more important coordinates, it is more efficient than Cyclic that selects the coordinate to update independently of the data. This explains the performances observed in Figure 3 and justifies the use of LGCD as the base of the DiCoDiLe$_Z$ distributed algorithm. In the following, we will not include Cyclic coordinate selection in our distributed algorithm as it is less efficient. We include Greedy as its complexity depends on the size of the signal, meaning that it can be competitive when the workload is distributed on many workers.

4.1.2 *Need for soft-lock mechanism*

The reconstruction of the Mandrill with and without soft-locks is shown in Figure 4. It uses a dictionary $\boldsymbol{D}$ consisting of $K = 25$ patches of size 16×16 extracted randomly from the original image. The reconstructed image $Z * \boldsymbol{D}$ obtained without soft-lock shows artifacts around the edges of some of the sub-domains $\mathcal{S}_w$, as the activations diverge in these locations. This visually demonstrates the need for controlling the interfering updates when more than two workers can interfere. When using the soft-lock mechanism, the algorithm returns the expected result.

4.1.3 *Impact of data partitioning*

To demonstrate the impact of the domain partitioning strategy on images, Figure 5 shows the average runtime for DiCoDiLe$_Z$

1. Code available at github.com/tommoral/dicodile.

2. The image is publicly available on Hubble website: www.hubblesite.org/image/3920/news/58-hubble-ultra-deep-field

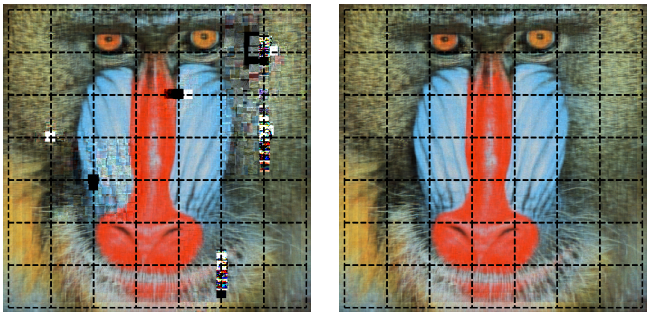


Fig. 4. Reconstruction of the image Mandrill using DiCoDiLe$_Z$ (*left*) without and (*right*) with soft-locks for a grid of 7×7 workers. The dashed lines show the partitions of the domain Ω. The algorithm diverges at the edges of some of the sub-domains due to interfering updates between more than two workers.

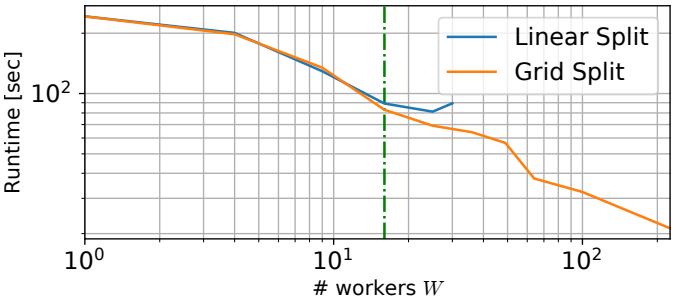


Fig. 5. Scaling on 2D images of DiCoDiLe$_Z$ with the number of workers for two partitioning strategies of Ω: (*blue*) Ω is split only along one direction, as in DICOD [19], (*orange*) Ω is split along both directions, on a grid. The running times are similar for low number of workers but the performances with the linear splits stop improving once W reaches the scaling limit $T_1/4L_1$ (*green*). With the grid of workers adapted to images, the performance improves further. The linear split stops when W reached T_1/L_1 because no more coordinate in $\mathcal{S}_w$ are independent from other neighbors.

using either a line of worker (unidirectional partitioning) or a grid of workers (bidirectional partitioning). In this experiment $K = 5$ and the atom size is 8×8. Both strategies scale similarly in the regime with low number of workers. But when W reaches $T_1/3L_1$, the performances of DiCoDiLe$_Z$ stops improving when using the unidirectional partitioning of Ω. Indeed in this case many update candidates are selected at the border of the sub-domains $\mathcal{S}_w$, therefore increasing the chance to reject an update. Moreover, the scaling of the unidirectional partitioning is limited to $W = T_1/2L_1 = 32$, whereas the bidirectional partitioning can be used with W up to 1024 workers.

4.1.4 Scaling of DiCoDiLe$_Z$

Scaling for 1D signals Figure 6 investigates the role of CD strategies in the distributed setting with 1D signals. The average runtime of DICOD (with GCD on each worker), and DiCoDiLe$_Z$ (with LGCD) are reported in Figure 6. DiCoDiLe$_Z$ with LGCD scales sub-linearly with the number of workers used, whereas DICOD scales almost quadratically. However, DiCoDiLe$_Z$ outperforms DICOD consistently, as the performance of GCD is poor for low number of workers using large sub-domains. The two coordinate selection strategies become equivalent when W reaches $T/4L$, as in this case DiCoDiLe has only one sub-domain $\mathcal{C}_1^{(w)}$. This is consistent for larger signals. Indeed, in the case where $T = 750$, DiCoDiLe$_Z$ scales

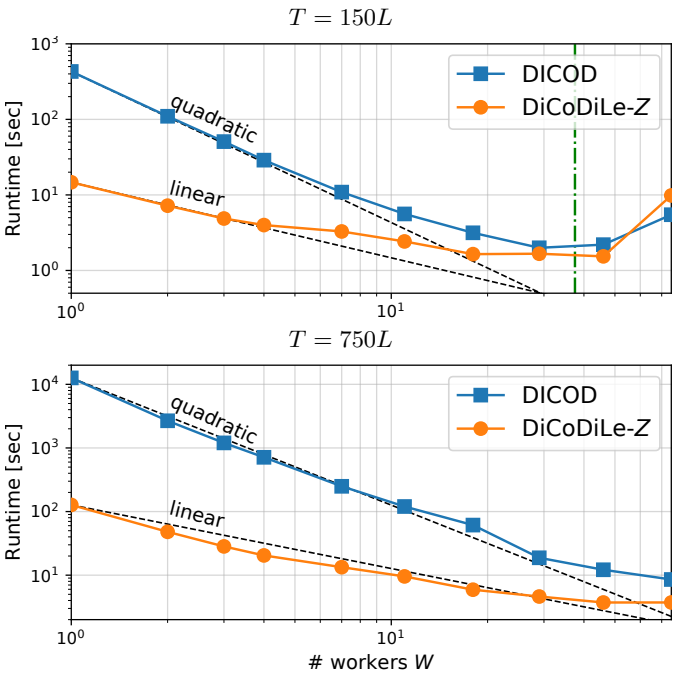


Fig. 6. Scaling of DICOD (*orange*) and DiCoDiLe$_Z$ (*blue*) with the number of workers W. DiCoDiLe$_Z$ scales sub-linearly while DICOD scales super-linearly. However, DiCoDiLe$_Z$ is more efficient than DICOD in a regime with a low number of workers W, leading to improved performances in all regimes. The green line denotes the number of cores where DiCoDiLe$_Z$ and DICOD become the same as DiCoDiLe$_Z$ only has one sub-domain in each worker. The results are consistent for both small (*top*, $T = 150L$) and large (*bottom* $T = 750L$) signals sizes.

linearly with the number of workers. We do not see the drop in performance, as in this experiments W does not reach the scaling limit $T/4L$. Importantly, DiCoDiLe$_Z$ still outperforms DICOD for all regimes.

Scaling for 2D signals Figure 7 illustrates the scaling performance of DiCoDiLe$_Z$ on images. The average running times of DiCoDiLe$_Z$ are reported as a function of the number of workers W for different regularization parameters λ and for greedy and locally greedy selection. As for the 1D CSC problem, the locally greedy selection consistently outperforms the greedy selection. This is particularly the case when using low number of workers W. As for signals, once the size of the sub-domains becomes smaller, the performances of both coordinate selection methods become closer as both algorithms become equivalent. One can also notice that the convergence of DiCoDiLe$_Z$ is faster for large λ. This is expected, as in this case, the solution Z^* is sparser and less coordinates need to be updated.

4.2 Comparison with Skau and Wohlberg [25]

Figure 8 displays the evolution of the objective function (3) as a function of time for DiCoDiLe and the Consensus ADMM algorithm[3] proposed by Skau and Wohlberg [25]. To the best of our knowledge, the work of Skau and Wohlberg [25] is the only approach in the literature that proposes a parallel algorithm for CDL. Both algorithms were run on a single node with 36 workers. We used the Hubble Space Telescope GOODS South image as input data. As the algorithm by Skau and Wohlberg [25] raised a memory error with the full image on

3. Code available at https://sporco.readthedocs.io/

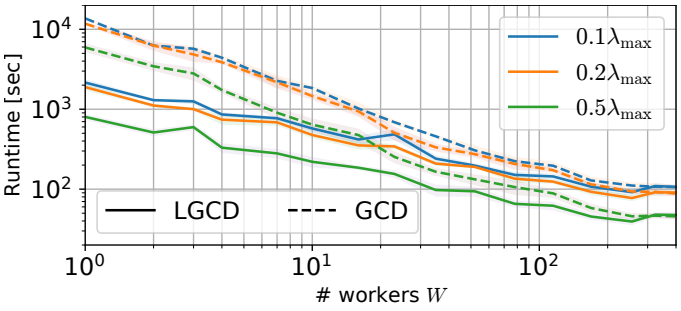


Fig. 7. Scaling of DiCoDiLe$_Z$ with the number of workers W for different values of λ and for two strategies of coordinate selection: Greedy and Locally Greedy. The convergence is faster when λ is large because the activation becomes sparser and less coordinates need to be updated. Also, the locally greedy coordinate selection outperforms the greedy selection up to a certain point where the performances become similar as the sub-domain $\mathcal{S}_w$ becomes too small to be further partitioned with sub-domains of size Θ.

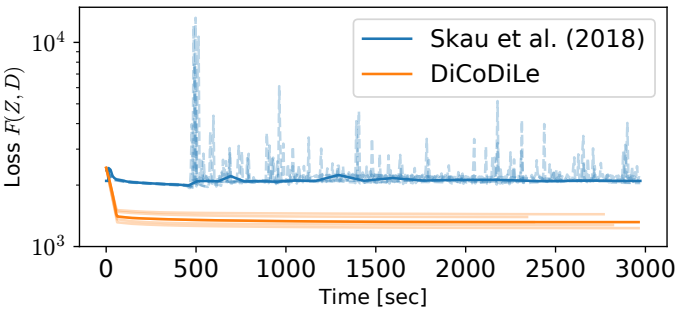


Fig. 8. Comparison of DiCoDiLe algorithm with Consensus ADMM dictionary learning proposed by Skau and Wohlberg [25]. Algorithms were run 5 times with 36 workers on a random-patch of size 512×512 taken from the Hubble Space Telescope GOODS South image. The solid lines denote the median curves obtained through interpolation.

a machine with 250 GB of RAM, we used a random-patch of size 512×512 as input data. Due to the non-convexity of the problem, the algorithms converge to different local minima, even when initialized with the same dictionary. To deal with this difficulty, the Figure 8 shows 5 restarts of both algorithms, as well as the median curve, obtained by interpolation. We can see that DiCoDiLe outperforms the Consensus ADMM as it converges faster and to a better local minima. To ensure a fair comparison, the objective function is computed after projecting back the atoms of the dictionary $\boldsymbol{D}_k$ to the ℓ_2-ball by scaling it with $\|\boldsymbol{D}_k\|_2^2$. This implies that we also needed to scale Z by the inverse of the dictionary norm *i.e.* $\frac{1}{\|\boldsymbol{D}_k\|_2^2}$. This projection and scaling step is necessary for the solver as ADMM iterations do not guarantee that all iterates are feasible (*i.e.*, satisfy the constraints). This also explains the presence of several bumps in the evolution curve of the objective function.

4.3 Learning dictionary on Hubble Telescope images

We present here results using DiCoDiLe to learn patterns from an image of the GOODS South field, acquired by the Hubble Space Telescope [13]. We used $W = 400$ workers to learn 25 atoms of size 32×32 with regularization parameter $\lambda = 0.1\lambda_{\max}$. The learned patterns are presented in Figure 9. This unsupervised algorithm is able to highlight structured patterns in the image, such as small stars. Patterns 1 and 7 are very fuzzy. This is probably due to the presence of very large object in the foreground. As this algorithm is not scale invariant,

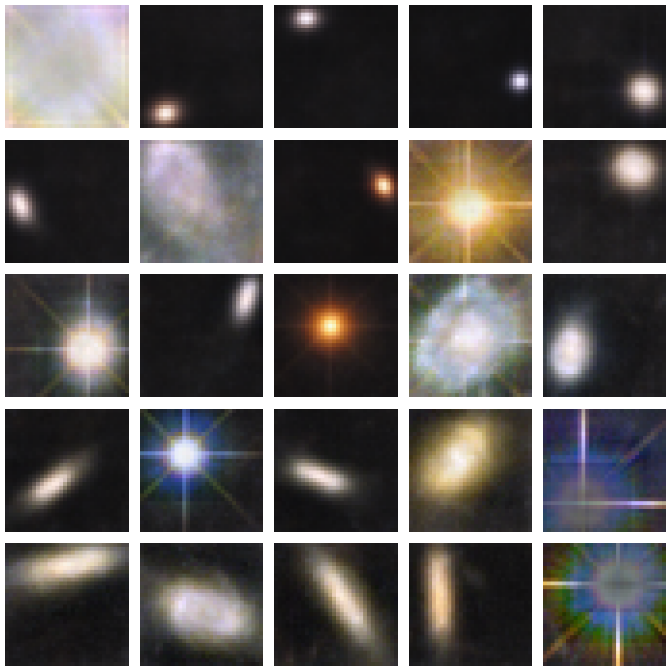


Fig. 9. 25 atoms 32×32 learned from the Hubble Telescope images (`STScI-H-2016-39-a`) using DiCoDiLe with 400 workers and a regularization parameter $\lambda = 0.1\lambda_{\max}$. The atoms are sorted based on $\|Z_k\|_1$ from the top-left to the bottom-right. Most atoms displays a structure which corresponds to spatial objects. The atoms 1 and 7 have fuzzier structure because they are mainly used to encode larger scale objects.

the objects larger than our patterns are encoded using these low frequency atoms.

5 Conclusion

This work introduces DiCoDiLe, an asynchronous distributed algorithm to apply convolutional dictionary learning to very large multidimensional data such as signals and images. The number of workers that can be used to solve the CSC scales linearly with the data size, allowing to adapt the computational resources to the problem dimensions. Using careful distributed pre-computations, the complexity of the dictionary update step is independent of the data size. Experiments show that it has good scaling properties and demonstrate that DiCoDiLe is able to learn patterns in astronomical images for large size which are not handled by other state-of-the-art parallel algorithms.

Appendix A
Computation for the cost updates

When a coordinate $Z_k[\omega]$ is updated to $\widetilde{Z}_k^{(q)}[\omega]$, the cost update is a simple function of $Z_k[\omega]$ and $\widetilde{Z}_k^{(q)}[\omega]$.

Proposition A.1. *If the coordinate (k_0, ω_0) is updated from value $Z_{k_0}[\omega_0]$ to value $\widetilde{Z}_{k_0}^{(q)}[\omega_0]$, then the change in the cost function $\Delta E_{k_0}[\omega_0]$ is given by:*

$$\begin{aligned}\Delta E_{k_0}[\omega_0] = &\frac{\|\boldsymbol{D}_{k_0}\|_2^2}{2}(Z_{k_0}[\omega_0]^2 - \widetilde{Z}_{k_0}^{(q)}[\omega_0]^2) \\ &- \beta_{k_0}[\omega_0](Z_{k_0}[\omega_0] - \widetilde{Z}_{k_0}^{(q)}[\omega_0]) \\ &+ \lambda(|Z_{k_0}[\omega_0]| - |\widetilde{Z}_{k_0}^{(q)}[\omega_0]|).\end{aligned}$$

Proof. We denote $Z_k^{(1)}[\omega] = \begin{cases} \widetilde{Z}_{k_0}^{(q)}[\omega_0], & \text{if } (k,\omega)=(k_0,\omega_0) \\ Z_k[\omega], & \text{elsewhere} \end{cases}$.
Let α_0 be the residual when coordinate (k_0,ω_0) of Z is set to 0. For $\omega \in \Omega$,

$$\begin{aligned}\alpha_{k_0}[\omega] &= (X - Z * \boldsymbol{D})[\omega] + \boldsymbol{D}_{k_0}[\omega-\omega_0]Z_{k_0}[\omega_0] \\ &= (X - Z * \boldsymbol{D})[\omega] + Z_{k_0}[\omega_0](e_{\omega_0} * \boldsymbol{D}_{k_0})[\omega_0] \ .\end{aligned}$$

We also have $\alpha_{k_0}[\omega] = (X - Z^{(1)} * \boldsymbol{D})[\omega] + \boldsymbol{D}_{k_0}[\omega-\omega_0]\widetilde{Z}_{k_0}^{(q)}[\omega_0]$. Then

$$\begin{aligned}\Delta E_{k_0}[\omega_0] =& \frac{1}{2}\sum_{\omega\in\Omega}(X - Z*\boldsymbol{D})^2[\omega] + \lambda\|Z\|_1 \\ &- \frac{1}{2}\sum_{\omega\in\Omega}\left(X - Z^{(1)}*\boldsymbol{D}\right)^2[\omega] + \lambda\|Z^{(1)}\|_1 \\ =& \frac{1}{2}\sum_{\omega\in\Omega}(\alpha_{k_0}[\omega] - \boldsymbol{D}_{k_0}[\omega-\omega_0]Z_{k_0}[\omega_0])^2 \\ &- \frac{1}{2}\sum_{\omega\in\Omega}\left(\alpha_{k_0}[\omega] - \boldsymbol{D}_{k_0}[\omega-\omega_0]\widetilde{Z}_{k_0}^{(q)}[\omega_0]\right)^2 \\ &+ \lambda(|Z_{k_0}[\omega_0]| - |\widetilde{Z}_{k_0}^{(q)}[\omega_0]|) \\ =& \frac{1}{2}\sum_{\omega\in\Omega}\boldsymbol{D}_{k_0}[\omega-\omega_0]^2(Z_{k_0}[\omega_0]^2 - \widetilde{Z}_{k_0}^{(q)}[\omega_0]^2) \\ &- \sum_{\omega\in\Omega}\alpha_{k_0}[\omega]\boldsymbol{D}_{k_0}[\omega-\omega_0](Z_{k_0}[\omega_0] - \widetilde{Z}_{k_0}^{(q)}[\omega_0]) \\ &+ \lambda(|Z_{k_0}[\omega_0]| - |\widetilde{Z}_{k_0}^{(q)}[\omega_0]|) \\ =& \frac{\|\boldsymbol{D}_{k_0}\|_2^2}{2}(Z_{k_0}[\omega_0]^2 - \widetilde{Z}_{k_0}^{(q)}[\omega_0]^2) \\ &- \underbrace{(\alpha_{k_0} * \boldsymbol{D}_{k_0}^{\curlywedge})[\omega_0]}_{\beta_{k_0}[\omega_0]}(Z_{k_0}[\omega_0] - \widetilde{Z}_{k_0}^{(q)}[\omega_0]) \\ &+ \lambda(|Z_{k_0}[\omega_0]| - |\widetilde{Z}_{k_0}^{(q)}[\omega_0]|)\end{aligned}$$

This concludes our proof. □

Using this result, we can derive the optimal value $\widetilde{Z}_{k_0}^{(q)}[\omega_0]$ to update the coordinate (k_0,ω_0) as the solution of the following optimization problem:

$$\begin{aligned}\widetilde{Z}_{k_0}^{(q)}[\omega_0] &= \arg\min_{y\in\mathbb{R}} e_{k_0,\omega_0}(u) \\ &= \arg\min_{u\in\mathbb{R}} \frac{\|\boldsymbol{D}_{k_0}\|_2^2}{2}(u-\beta_{k_0}[\omega_0])^2 + \lambda|u| \ .\end{aligned} \tag{A.1}$$

In the case where multiple coordinates (k_w,ω_w) are updated in the same iteration to values $\widetilde{Z}_{k_w}^{(q)}[\omega_w]$, we obtain the following cost variation.

Proposition A.2. *The update of the W coordinates $(k_w,\omega_w)_{w=1}^W$ with additive update $\Delta Z_{k_w}[\omega_w]$ changes the cost by:*

$$\begin{aligned}\Delta E =& \overbrace{\sum_{i=1}^{W}}^{\text{iterative steps}} \Delta E_w \\ &\underbrace{- \sum_{w\neq w'}(\boldsymbol{D}_{k_w} * \boldsymbol{D}_{k_{w'}}^{\curlywedge})[\omega_{w'}-\omega_w]\Delta Z_{k_w}[\omega_w]\Delta Z_{k_{w'}}[\omega_{w'}]}_{\text{interference}},\end{aligned}$$

Proof. We define $Z_k^{(1)}[\omega] = \begin{cases} \widetilde{Z}_{k_w}^{(q)}[\omega_w], & \text{if } (k,\omega)=(k_w,\omega_w) \\ Z_k[\omega], & \text{otherwise} \end{cases}$.
Let

$$\begin{aligned}\gamma[\omega] =& (X - Z*\boldsymbol{D})[\omega] + \sum_{w=1}^{K}\boldsymbol{D}_{k_w}[\omega-\omega_w]Z_{k_w}[\omega_w] && \text{(A.2)} \\ =& (X - Z^{(1)}*\boldsymbol{D})[\omega] + \sum_{w=1}^{K}\boldsymbol{D}_{k_w}[\omega-\omega_w]\widetilde{Z}_{k_w}^{(q)}[\omega_w] \ , && \text{(A.3)}\end{aligned}$$

and

$$\begin{aligned}\alpha_w[\omega] =& (X - Z*\boldsymbol{D})[\omega] + \boldsymbol{D}_{k_w}[\omega-\omega_w]Z_{k_w}[\omega_w] && \text{(A.4)} \\ =& \gamma[\omega] - \sum_{w'\neq w}\boldsymbol{D}_{k_{w'}}[\omega-\omega_{w'}]Z_{k_{w'}}[\omega_{w'}] \ . && \text{(A.5)}\end{aligned}$$

Then

$$\begin{aligned}\Delta E =& \frac{1}{2}\sum_{\omega\in\Omega}\Big(X[\omega] - Z*\boldsymbol{D}[\omega]\Big)^2 + \lambda\|Z\|_1 && \text{(A.6)} \\ &- \sum_{\omega\in\Omega}\Big(X[\omega] - Z^{(1)}*\boldsymbol{D}[\omega]\Big)^2 - \lambda\|Z^{(1)}\|_1 \\ =& \frac{1}{2}\sum_{\omega\in\Omega}\Big(\gamma[\omega] - \sum_{w=1}^{W}\boldsymbol{D}_{k_w}[\omega-\omega_w]Z_{k_w}[\omega_w]\Big)^2 \\ &- \frac{1}{2}\sum_{\omega\in\Omega}\Big(\gamma[\omega] - \sum_{w=1}^{W}\boldsymbol{D}_{k_w}[\omega-\omega_w]\widetilde{Z}_{k_w}^{(q)}[\omega_w]\Big)^2 && \text{(A.7)} \\ &+ \lambda\sum_{w=1}^{W}(|Z_{k_w}[\omega_w]| - |\widetilde{Z}_{k_w}^{(q)}[\omega_w]|) \\ =& \frac{1}{2}\sum_{w=1}^{W}\Big(\sum_{\omega\in\Omega}\boldsymbol{D}_{k_w}[\omega-\omega_w]^2\Big)(Z_{k_w}[\omega_w]^2 - \widetilde{Z}_{k_w}^{(q)}[\omega_w]^2) \\ &+ \lambda\sum_{w=1}^{W}(|Z_{k_w}[\omega_w]| - |\widetilde{Z}_{k_w}^{(q)}[\omega_w]|) \\ &- \sum_{w=1}^{W}\sum_{\omega\in\Omega}\Big(\gamma[\omega]\boldsymbol{D}_{k_w}[\omega-\omega_w]\Big)\Delta Z_{k_w}[\omega_w] && \text{(A.8)} \\ &- \sum_{w\neq w'}\Big[\sum_{\omega\in\Omega}\Big(\boldsymbol{D}_{k_w}[\omega-\omega_w]\boldsymbol{D}_{k_{w'}}[\omega-\omega_{w'}]\Big) \\ &\qquad \times (Z_{k_w}[\omega_w]Z_{k_{w'}}[\omega_{w'}] - \widetilde{Z}_{k_w}^{(q)}[\omega_w]\widetilde{Z}_{k_{w'}}^{(q)}[\omega_{w'}])\Big] \\ =& \sum_{w=1}^{W}\Big[\frac{\|\boldsymbol{D}_{k_w}\|_2^2}{2}(Z_{k_w}[\omega_w]^2 - \widetilde{Z}_{k_w}^{(q)}[\omega_w]^2) \\ &\qquad - \underbrace{\Big(\sum_{\omega\in\Omega}\alpha_{k_w}[\omega]\boldsymbol{D}_{k_w}[\omega-\omega_w]\Big)}_{\alpha_{k_w}*\boldsymbol{D}_{k_w}^{\curlywedge}[\omega_w]}\Delta Z_{k_w}[\omega_w] \\ &\qquad + \lambda(|Z_{k_w}[\omega_w]| - |\widetilde{Z}_{k_w}^{(q)}[\omega_w]|)\Big] && \text{(A.9)} \\ &- \sum_{w\neq w'}\Big[(\boldsymbol{D}_{k_{w'}} * \boldsymbol{D}_{k_w}^{\curlywedge})[\omega_w-\omega_{w'}] \\ &\qquad \times \Big(\Delta Z_{k_w}[\omega_w]\widetilde{Z}_{k_{w'}}^{(q)}[\omega_{w'}] + \Delta Z_{k_{w'}}[\omega_{w'}]\widetilde{Z}_{k_w}^{(q)}[\omega_w] \\ &\qquad\quad - Z_{k_w}[\omega_w]Z_{k_{w'}}[\omega_{w'}] - \widetilde{Z}_{k_w}^{(q)}[\omega_w]\widetilde{Z}_{k_{w'}}^{(q)}[\omega_{w'}]\Big)\Big] \\ =& \sum_{w=1}^{W}\Delta E_{k_w}[\omega_w] && \text{(A.10)} \\ &- \sum_{w\neq w'}(\boldsymbol{D}_{k_{w'}} * \boldsymbol{D}_{k_w}^{\curlywedge})[\omega_w-\omega_{w'}]\Delta Z_{k_w}[\omega_w]\Delta Z_{k_{w'}}[\omega_{w'}]\end{aligned}$$

□

APPENDIX B
PROBABILITY OF A COORDINATE TO BE SOFT-LOCKED

Proposition B.1. *We consider DiCoDiLe$_Z$ with W workers (W_i in each direction), run on a signal with d dimensions. We denote T_i the length of the signal and L_i the length of the atoms in each dimension i. Assuming that the updates are spread uniformly on Ω, the probability that an update candidate at random localization ω is accepted is lower bounded by*

$$P(\omega \notin \boldsymbol{SL}) \geq \prod_{i=1}^{d} \left(1 - \frac{W_i L_i}{T_i}\right) \quad .$$

Proof. We denote ω_0 the position of an update candidate, and $\mathcal{W}_{\omega_0} = \{w \in [1, W] \ \ s.t. \ \ \mathcal{S}_w \cap \mathcal{V}_\Theta(\omega) = \emptyset\}$ the set of workers that are impacted by this update. We denote $j_{\omega_0} = |\mathcal{W}_{\omega_0}|$ the number of workers in this set. With the definition of the soft-lock, there exists at least one worker w in $\mathcal{W}_{\omega_0}$ which has a coordinate that is not soft-locked. As the updates are uniformly spread across the workers, the probability of an update in ω_0 being accepted satisfies $P(\omega \notin \mathbf{SL} \mid \omega = \omega_0) \geq \frac{1}{j_{\omega_0}}$.

Since one assumes uniform update distribution, the probability to update a coordinate at location ω_0 is $P(\omega_0) = \frac{1}{|\Omega|}$. The probability for an update candidate at a random location ω to be accepted satisfies

$$\begin{aligned}
P(\omega \notin \mathbf{SL}) &= \sum_{w=1}^{W} \sum_{\omega_0 \in \mathcal{S}_w} P(\omega \notin \mathbf{SL} \mid \omega = \omega_0) P(\omega_0) && \text{(B.1)} \\
&\geq \sum_{w=1}^{W} \sum_{\omega_0 \in \mathcal{S}_w} \frac{1}{j_{\omega_0}} \frac{1}{|\Omega|} && \text{(B.2)} \\
&\geq \frac{1}{|\Omega|} \sum_{w=1}^{W} \prod_{i=1}^{d} \left(\frac{T_i}{W_i} - L_i\right) && \text{(B.3)} \\
&\geq \frac{1}{|\Omega|} \prod_{i=1}^{d} (T_i - W_i L_i) && \text{(B.4)} \\
&\geq \prod_{i=1}^{d} \left(1 - \frac{W_i L_i}{T_i}\right) \quad , && \text{(B.5)}
\end{aligned}$$

where (B.3) results from the summation on Ω. Basically, in each worker w, there is $\prod_{i=1}^{d} (T_i/W_i - 2L_i)$ coordinates for which $j_{\omega_0} = 1$. This corresponds to the coordinate in the worker that are not on the border *i.e.* $\mathcal{S}_w \setminus \mathcal{B}_\Theta(\mathcal{S}_w)$. For a coordinate at the border with $j_{\omega_0} = k > 1$, it suffices to notice that it will appear k times in the sum due to symmetry considerations. This is illustrated in Figure 10.

□

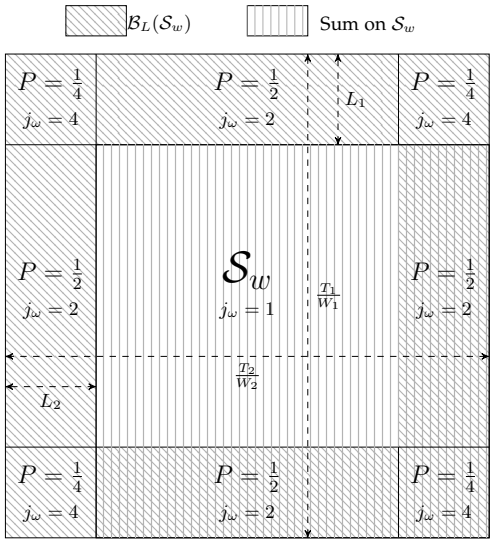


Fig. 10. Summary of the value of j_ω on a 2D example. We can see that the sum along each direction is $\frac{T_i}{W_i} - L_i$ on each workers. When multiplied by the number of workers W_i along each direction, we obtain (B.4).

ACKNOWLEDGMENTS

This work was supported by the ERC Starting Grant SLAB ERC-YStG-676943.

REFERENCES

[1] M. Aharon, M. Elad, and A. M. Bruckstein, "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation," *IEEE Transaction on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] A. Beck and M. Teboulle, "A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[3] J. Bergstra, A. Courville, and Y. Bengio, "The Statistical Inefficiency of Sparse Coding for Images (or One Gabor to Rule them All)," *preprint ArXiv*, vol. 1109.6638, 2011.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[5] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Portland, OR, USA, 2013, pp. 391–398.

[6] R. Chalasani, J. C. Principe, and N. Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *International Joint Conference on Neural Networks (IJCNN)*, Dallas, TX, USA, 2013, pp. 1–5.

[7] P. L. Combettes and H. H. Bauschke, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011, publication Title: Springer Science & Business Media.

[8] L. Dalcín, R. Paz, and M. Storti, "MPI for Python," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1108–1115, 2005.

[9] P. del Aguila Pla and J. Jaldén, "Convolutional Group-Sparse Coding and Source Localization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2776–2780.

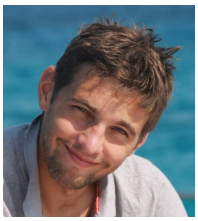
[10] T. Dupré la Tour, T. Moreau, M. Jas, and A. Gramfort, "Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals," in *Advances in Neural Information Processing Systems (NeurIPS)*, Montreal, Canada, 2018, pp. 3296–3306.

[11] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *The Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[12] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers \& Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976.

[13] M. Giavalisco and the GOODS Teams, "The Great Observatories Origins Deep Survey: Initial Results From Optical and Near-Infrared Imaging," *The Astrophysical Journal Let-*

ters, vol. 600, no. 2, p. L93, 2004.
[14] R. Grosse, R. Raina, H. Kwong, and A. Y. Ng, "Shift-Invariant Sparse Coding for Audio Classification," *Cortex*, vol. 8, p. 9, 2007.
[15] M. Jas, T. Dupré la Tour, U. Şimşekli, and A. Gramfort, "Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding," in *Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 1099–1108.
[16] S. P. Karimireddy, A. Koloskova, S. U. Stich, and M. Jaggi, "Efficient Greedy Coordinate Descent for Composite Problems," in *Artificial Intelligence and Statistics (AISTAT)*, 2019.
[17] K. Kavukcuoglu, P. Sermanet, Y.-l. Boureau, K. Gregor, and Y. Le Cun, "Learning Convolutional Feature Hierarchies for Visual Recognition," in *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2010, pp. 1090–1098.
[18] J. Mairal, F. R. Bach, J. Ponce, and G. Sapiro, "Online Learning for Matrix Factorization and Sparse Coding," *Journal of Machine Learning Research (JMLR)*, vol. 11, no. 1, pp. 19–60, 2010.
[19] T. Moreau, L. Oudre, and N. Vayatis, "DICOD: Distributed Convolutional Sparse Coding," in *International Conference on Machine Learning (ICML)*. Stockohlm, Sweden: PMLR (80), 2018, pp. 3626–3634.
[20] J. Nutini, M. Schmidt, I. H. Laradji, M. P. Friedlander, and H. Koepke, "Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection," in *International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 1632–1641.
[21] S. Osher and Y. Li, "Coordinate descent optimization for ℓ_1 minimization with application to compressed sensing; a greedy algorithm," *Inverse Problems and Imaging*, vol. 3, no. 3, pp. 487–503, 2009.
[22] V. Papyan, Y. Romano, M. Elad, and J. Sulam, "Convolutional Dictionary Learning via Local Processing," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5306–5314.
[23] M. Pereyra and A. T. Figueiredo, "Maximum-A-Posteriori Estimation with Unknown Regularization Parameters," in *European Signal Processing Conference (EUSIPCO)*, Nice, France, 2015, pp. 230–234.
[24] Python Software Fundation, "Python Language Reference, version 3.6," 2017, pages: https://python.org.
[25] E. Skau and B. Wohlberg, "A Fast Parallel Algorithm for Convolutional Sparse Coding," in *IEEE Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, 2018.
[26] J.-L. Starck, M. Elad, and D. L. Donoho, "Image Decomposition Via the Combination of Sparse Representation and a Variational Approach," *IEEE Trans.Image Process.*, vol. 14, no. 10, pp. 1570–1582, 2005.
[27] A. F. Vidal and M. Pereyra, "Maximum Likelihood Estimation of Regularization Parameters," in *IEEE International Conference on Image Processing (ICIP)*. Athens, Greece: IEEE, 2018, pp. 1742–1746.
[28] B. Wohlberg, "Efficient Algorithms for Convolutional Sparse Representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, 2016.
[29] F. Yellin, B. D. Haeffele, and R. Vidal, "Blood cell detection and counting in holographic lens-free imaging by convolutional sparse dictionary learning and coding," in *IEEE International Symposium on Biomedical Imaging (ISBI)*, Melbourne, Australia, 2017.
[30] Y. Zhang, H. W. Kuo, and J. Wright, "Structured local minima in sparse blind deconvolution," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 2018-Decem, pp. 2322–2331, 2018.
[31] E. Zisselman, J. Sulam, and M. Elad, "A Local Block Coordinate Descent Algorithm for the Convolutional Sparse Coding Model," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8208–8217.



Thomas Moreau received the graduate degree from the Ecole Polytechnique, Palaiseau, France, in 2014, and the PhD degree from the Ecole Normale Supérieure, Cachan, France, in 2017 under the supervision of Nicolas Vayatis and Laurent Oudre in the CMLA laboratory. He recently joined the Inria Parietal project team in Saclay, first as a post-doctoral researcher and then as a researcher. His research interests include unsupervised learning, image/signal processing and distributed computing.

Alexandre Gramfort graduated from Ecole Polytechnique in 2005 and Telecom ParisTech in 2006, and he holds a MS in Applied Mathematics from Ecole Normale Supérieure, Cachan, France. He is currently senior researcher in the Parietal Team at INRIA Saclay Research Center and CEA Neurospin since 2017. He was formerly Assistant Professor at Telecom ParisTech, Université Paris-Saclay, in the image and signal processing department. His field of expertise is statistical machine learning, signal processing and scientific computing applied primarily to functional brain imaging data (EEG, MEG, fMRI).