electronics

MDPI

Article

IPGM: Inertial Proximal Gradient Method for Convolutional Dictionary Learning

Jing Li [1], Xiao Wei [1], Fengpin Wang [2] and Jinjia Wang [2,3,*]

[1] School of Science, Yanshan University, Qinhuangdao 066004, China; lijing1977@ysu.edu.cn (J.L.); sjr@stumail.ysu.edu.cn (X.W.)
[2] School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China; wangfengpin@stumail.ysu.edu.cn
[3] Hebei Key Laboratory of Information Transmission and Signal Processing, Yanshan University, Qinhuangdao 066004, China
* Correspondence: wjj@ysu.edu.cn

Abstract: Inspired by the recent success of the proximal gradient method (PGM) and recent efforts to develop an inertial algorithm, we propose an inertial PGM (IPGM) for convolutional dictionary learning (CDL) by jointly optimizing both an ℓ_2-norm data fidelity term and a sparsity term that enforces an ℓ_1 penalty. Contrary to other CDL methods, in the proposed approach, the dictionary and needles are updated with an inertial force by the PGM. We obtain a novel derivative formula for the needles and dictionary with respect to the data fidelity term. At the same time, a gradient descent step is designed to add an inertial term. The proximal operation uses the thresholding operation for needles and projects the dictionary to a unit-norm sphere. We prove the convergence property of the proposed IPGM algorithm in a backtracking case. Simulation results show that the proposed IPGM achieves better performance than the PGM and slice-based methods that possess the same structure and are optimized using the alternating-direction method of multipliers (ADMM).

Keywords: convolutional sparse representation; needle; convolutional dictionary learning; inertia term; proximal gradient descent; convergence

Citation: Li, J.; Wei, X.; Wang, F.; Wang, J. IPGM: Inertial Proximal Gradient Method for Convolutional Dictionary Learning. *Electronics* **2021**, *10*, 3021. https://doi.org/10.3390/electronics10233021

Academic Editor: Bhanu Prakash KN

Received: 14 October 2021
Accepted: 1 December 2021
Published: 3 December 2021

1. Introduction

Sparse representation is a popular a priori mathematical modeling approach in various signal and image processing applications. Inspired by deep learning-based convolutional operations, convolutional sparse representation is a hot topic at present. In the convolutional sparse representation model, the convolutional dictionary learning (CDL) process plays an important role, but the associated algorithm and convergence proof are difficult problems [1]. CDL involves training dictionaries and estimating codes from multiple signals. In the past decade, the convolutional sparse representation model has achieved impressive results in various applications. The resulting model can achieve excellent performance in image denoising and repair [2], image decomposition [3], image reconstruction [4], medical imaging [5,6], trajectory reconstruction [7], image segmentation [8], superresolution [9], audio processing [10], and other applications. CDL research is a typical interdisciplinary research subject combining mathematics and artificial intelligence. The research results in this field have important theoretical and practical value for a variety of signal and image processing applications. In such a case, a typical assumption is that a signal $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{y} = \mathbf{D\Gamma}$ is a linear combination of columns, also known as atoms. The coefficient $\mathbf{\Gamma} \in \mathbb{R}^M$ is a sparse vector. A matrix $\mathbf{D} \in \mathbb{R}^{N \times M}$ is called a dictionary. Given $\mathbf{y}$ and $\mathbf{D}$, this task of finding its sparsest representation is equivalent to solving the following problem:

$$\min_{\mathbf{\Gamma}} \|\mathbf{\Gamma}\|_0 \text{ s.t. } \|\mathbf{y}\text{-}\mathbf{D\Gamma}\|_2 \leq \varepsilon \quad (1)$$

where ε represents the degree of model mismatch or the additive noise intensity. The solutions to such problems can be approximated by greedy algorithms (such as orthogonal matching pursuit (OMP) [11]) or convex formulas (such as basis pursuit (BP) [12]). The task of the developed learning model is to identify the dictionary $\mathbf{D}$ that can best represent a set of training signals, called dictionary learning. Several processing methods have been proposed, including K-singular value decomposition (K-SVD) [13], the method of optimal directions (MOD) [14], online dictionary learning [15], and trainlet learning [16]. Unfortunately, many real-world signals, such as images and audio, are high-dimensional signals, making sparse coding computationally challenging. To avoid the curse of dimensionality, high-dimensional signals are decomposed into low-dimensional overlapping signal patches, and sparse coding is performed independently on each signal block [17]. Because of its simplicity and high performance, this method has been widely utilized in a successful manner. However, this method has some limitations that its patch-based process ignores the correlations between patches.

Another option that contrasts with this local paradigm is a global model [18] called the convolutional sparse representation model. The input signal is represented as the superposition of the convolutional results of some local atoms and sparse feature mappings. The problem is solved by applying a specific structure to the global dictionary involved. In particular, the dictionary in this model is constrained to a banded circulant matrix formed by atomic cascades, which is called convolutional sparse representation.

The convolutional sparse representation model is as follows:

$$\mathbf{y} = \mathbf{D}\boldsymbol{\Gamma} = \sum_{m} \mathbf{d}_m * \boldsymbol{\gamma}_m \tag{2}$$

where a signal $\mathbf{y} \in \mathbb{R}^N$, the banded global convolutional dictionary $\mathbf{D} \in \mathbb{R}^{N \times NM}$ is composed of a local convolutional dictionary filter, $\{\mathbf{d}_m\}_{m=1}^{M} \in \mathbb{R}^n$, a cyclic shift composed of the atom $\mathbf{D}_L \in \mathbb{R}^{n \times M}$, and the coding coefficient $\boldsymbol{\Gamma} \in \mathbb{R}^{NM}$ is composed of $\{\boldsymbol{\gamma}_i\}_{i=1}^{M}$, where $\boldsymbol{\gamma}_i \in \mathbb{R}^N$.

This work proposes a theoretical analysis of a new global convolutional sparse representation model, which guarantees the local sparsity metric. The convolutional dictionary directly determines translation invariance to compensate for the problem that the correlations between adjacent signal patches are ignored. By using this convolutional dictionary structure, convolutional sparse coding (CSC) and CDL have arisen. For the convenience of calculation, the ℓ_1 norm of the coefficient is used instead of the ℓ_0 norm, and a spherical constraint is used for the dictionary instead of the sphere constraint. Several convex and relaxed CDL algorithms have been proposed [19–22]. They mainly utilize the alternating-direction method of multipliers (ADMM) solver based on the Fourier domain. However, the ADMM loses its connection to the patch-based processing paradigm that is widely used in many signal and image processing applications [23].

Papyan proposed a new convex relaxation algorithm [23] via slice-based dictionary learning (SBDL), where the sum of the slices forms the signal patches. The developed algorithm is an ADMM solver based on the signal domain. It adopts a local point of view and trains a convolutional dictionary only according to the local calculation of the signal domain. It runs on local slices and faithfully solves the problem of global convolutional sparse coding. This local–global method and its resulting decomposition follow a recent work [18] that compared it with the Fourier-based method. The SBDL algorithm is easy to implement, is intuitive, achieves the most advanced performance, and converges faster than other approaches. Moreover, it provides a better model that can naturally allow a different number of nonzero values to be contained in each spatial location according to the local signal complexity. The ADMM parameters strongly depend on the given problem. Although its corresponding pursuit algorithm can be proven to be convergent, when the convolutional dictionary is iteratively updated in the ADMM algorithm, it is difficult to prove the convergence of the tracking algorithm [24].

The convex relaxation and ADMM-based optimization algorithm produce a nonsparse coding solution, and it is challenging to prove the algorithmic convergence [25]. In addition, in nonconvex optimization, the greedy algorithm for CDL problems has a high computational cost, poor performance, and convergence proof difficulty [26]. Chun and Fessler [27] recently proposed an algorithm that could achieve full convergence. However, the method involves approximate convex constraints, and its overall performance is slightly better than that of the ADMM. The CDL problem is essentially a nonconvex and nonsmooth optimization problem, as shown in the following formula, making it difficult to propose an optimization algorithm and prove its convergence.

$$\underset{\{\mathbf{d}_m\},\{\boldsymbol{\gamma}_{l,m}\}}{\operatorname{argmin}} \frac{1}{2}\sum_{l}\left\|\sum_{m=1}^{M}\mathbf{d}_m * \boldsymbol{\gamma}_{l,m} - \mathbf{y}_l\right\|_2^2 + \lambda_1\sum_{l}\sum_{m=1}^{M}\Omega_1\left(\boldsymbol{\gamma}_{l,m}\right) + \lambda_2\sum_{m=1}^{M}\Omega_2(\mathbf{d}_m) \tag{3}$$

where $\lambda_i, i = 1, 2$ is the equilibrium coefficient and $\Omega_i, i = 1, 2$ is the nonconvex constraint of the coefficient and the convolutional dictionary. For example, in a case with typical nonconvex constraints, $\Omega_1(\mathbf{x}) = \|\mathbf{x}\|_0$ is a zero norm, and $\Omega_2(d_m) = \|\mathbf{d}_m\|_2^2 = 1$ is a spherical constraint.

Peng [28] realized the joint and direct optimization of the CDL problem under a nonconvex and nonsmooth optimization scheme and proposed a forward–backward splitting algorithm based on the Fourier domain. The developed approach is better than the ADMM algorithm. To be more precise, the forward step involves estimating the smooth part of the objective function through a partial gradient. In contrast, the backward step counts the degree of nonsmoothness of the objective function through its proximal operator. Peng proved the convergence of the solution sequence of the proposed algorithm by using the semialgebraic property of reference [29] and the Kurdyka–Lojasiewicz (KL) property. Peng [28] used the gradient descent algorithm in the dictionary and coding process. Although this dramatically reduces the computational complexity of the overall algorithm, in theory, the gradient descent algorithm can only guarantee that it will reach the lowest local point, not the lowest global point. Many minimum points are contained in many complex functions. In many cases, we can only obtain the optimal local solution by using the gradient descent method and not the optimal global solution. In addition, when the sample size of the given dataset is large, the convergence speed of the gradient descent algorithm is slow.

Polyak [30] proposed the heavy-ball method that an inertia term is added to the standard gradient descent method. This method has a faster convergence rate than the standard gradient method based on an unchanging number of required calculations. Peter Ochs [31] applied this method to a convex optimization scheme. He proposed an iPiano algorithm combining an inertia term and a forward–backward splitting frame to address minimization problems consisting of differentiable (possibly nonconvex) and convex (possibly nondifferentiable) functions. These problems were strictly analyzed, and the global convergence of the function values and parameters was determined. Simultaneously, the efficiency of convergence was greatly improved. We will apply this inertia term algorithm to the CDL problem.

In this paper, an inertial forward–backward splitting algorithm is proposed for the CDL problem. The convergence of the algorithm is given and proven. The optimal convergence rate is derived. Finally, a large number of experiments show that the proposed algorithm has high efficiency and effectiveness.

The rest of this article is organized as follows. In Section 2, we introduce some related knowledge. In Section 3, the inertia forward and backward splitting algorithm is proposed to restate the CDL problem. In Section 4, the complexity of the proposed algorithm is analyzed. In Section 5, the convergence of the proposed algorithm is analyzed and proven. The convergence rate of the proposed algorithm is derived in Section 6. In Section 7, the performance of the proposed algorithm is evaluated through experiments and compared with other existing methods. Section 8 summarizes the full text.

2. Related Knowledge

2.1. Convex CDL in the Time Domain via Local Processing

The CSC model [19] assumes that a global signal $\mathbf{y} \in \mathbb{R}^N$ can be decomposed as

$$\mathbf{y} = \mathbf{D}\boldsymbol{\Gamma} = \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_i \tag{4}$$

The matrix $\mathbf{D} \in \mathbb{R}^{N \times NM}$ is a banded convolutional dictionary. This matrix consists of all shifted versions of a local dictionary $\mathbf{D}_L \in \mathbb{R}^{n \times m}$ whose columns are atoms. L represents the initial of the word "local." The global sparse vector $\boldsymbol{\Gamma} \in \mathbb{R}^{NM}$ can be decomposed into N non-overlapping, m-dimensional local sparse vectors $\{\boldsymbol{\alpha}_i\}_{i=1}^N$, where $\boldsymbol{\alpha}_i \in \mathbb{R}^M$ are called needles. The operator $\mathbf{P}_i^T$ places $\mathbf{D}_L \boldsymbol{\alpha}_i$ in the ith position of the signal. The above formula can be used to obtain sparse coefficients using the basic pursuit problem, as shown below:

$$\min_{\{\boldsymbol{\alpha}_i\}_{i=1}^N} \frac{1}{2} \left\| \mathbf{y} - \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_i \right\|_2^2 + \lambda \sum_{i=1}^{N} \|\boldsymbol{\alpha}_i\|_1 \tag{5}$$

Papyan et al. [23] proposed slice-based local processing and defined $\mathbf{s}_i = \mathbf{D}_L \boldsymbol{\alpha}_i$ as the ith slice. Unlike other existing works in signal and image processing, which train a dictionary in the Fourier domain, they defined the learning problem based on the constructed slices, and CDL was carried out in the signal domain via local processing. Through local processing in the original signal domain, the global problem was solved completely. The global signal $\mathbf{y}_l$ can be rewritten as $\mathbf{y} = \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{s}_i$.

The ADMM algorithm is used to solve the problem of minimizing the following augmented Lagrangian problem:

$$\min_{\mathbf{D}_L, \{\boldsymbol{\alpha}_i\}_{i=1}^N, \{\mathbf{s}_i\}_{i=1}^N, \{\mathbf{u}_i\}_{i=1}^N} \frac{1}{2} \left\| \mathbf{y} - \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{s}_i \right\|_2^2 + \sum_{i=1}^{N} \left(\lambda \|\boldsymbol{\alpha}_i\|_1 + \frac{\rho}{2} \|\mathbf{s}_i - \mathbf{D}_L \boldsymbol{\alpha}_i + \mathbf{u}_i\|_2^2 \right) \tag{6}$$

where $\{\mathbf{u}_i\}_{i=1}^N$ are the dual variables that satisfy the given constraint. ρ is the Lagrangian coefficient. References [23,24] call this method the SBDL algorithm.

In the SBDL algorithm, the CDL problem is transformed into a traditional dictionary learning problem solved by the K-SVD algorithm [13] or other dictionary learning algorithms. The sparse coding process is based on the ADMM algorithm for updating, and each coding update is regarded as a least absolute shrinkage and selection operator (LASSO) problem, which is solved by the least-angle regression and shrinkage (LARS) algorithm. The ADMM is used to solve the coding problem, which increases the numbers of auxiliary variables, calculations and redundant iterations. In addition, different solving methods are used to update dictionary and codes, which makes it difficult to prove the convergence of the algorithm.

2.2. Forward–Backward Splitting

Splitting algorithms for convex optimization problems usually originate from the proximal point algorithm [32]. The proximal point algorithm is very general, and the results regarding its convergence affect many other algorithms. However, in practice, an iteration of the computational algorithm can be as tricky as the original problem. The strategy to solve this problem involves splitting approaches such as the Douglas–Rachford method, several primal-dual algorithms, and forward–backward splitting.

Forward–backward splitting schemes have been used to solve a variety of problems in recent years. For example, forward–backward splitting algorithms are used to solve normal problems [33], to find generalized Nash equilibrium [34], to solve linear constraint problems [35], or to analyze related function problems in Banach space [36,37]. In particular, it is appealing to generalize forward–backward splitting schemes to nonconvex problems. This is due to their simplicity and simpler formulations in some exceptional cases, such

as the gradient projection method, where the backward step is a projection onto a set. The backward step of the forward–backward algorithm studied in [28] is the solution of a proximal term of a nonconvex function.

The goal of a forward–backward splitting framework is to solve the following forms of an optimization problem:

$$\underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x}) + g(\mathbf{x}) \tag{7}$$

However, when a large amount of data is processed, even if misestimation is not considered during processing, the processing result of the forward–backward splitting algorithm becomes inaccurate. In this paper, the original algorithm is improved to increase the accuracy and reduce the induced error.

2.3. Inertia Item

Polyak studied a multistep scheme for the accelerated gradient method in [30] and proposed the heavy-ball method for the first time. Unlike the usual gradient method, this approach adds an inertia term, which is calculated by the difference between the values obtained during the previous two iterations. Compared with the standard gradient method, this method can accelerate the convergence speed of the algorithm while keeping the cost of each iteration unchanged. In addition, this method can obtain the optimal convergence rate without additional knowledge.

The inertia term of the heavy-ball method was first applied to the minimization of differentiable convex functions, and then it was extended to subdifferential convex functions. Later, the heavy-ball method was used for forward–backward splitting [37,38]. Recently, it has been frequently applied to the minimization of nonconvex functions. In [39,40], the inertia term was introduced into the nonconvex optimization problem to improve the convergence speed. Refs. [31,41] used an inertia term to develop a nonconvex optimization CDL scheme. The authors of [31] aimed to minimize problems composed of differentiable (possibly nonconvex) and convex (possibly nondifferentiable) functions. The iPiano algorithm was proposed by combining forward–backward splitting with an inertia term. The global convergence of the function values and parameters was determined. In [41], an inertial version of the proximal alternating linearization minimization (PALM) algorithm was proposed, and its global convergence to the critical point of the input objective function was proven.

3. Proposed IPGM Algorithm

The CDL problem to be solved via local processing is given as follows:

$$\underset{\{\boldsymbol{\alpha}_i\}_{i=1}^{N}, \mathbf{D}_L}{\operatorname{argmin}} \frac{1}{2} \sum_{l} \left\| \sum_{i=1}^{N} \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_{l,i} - \mathbf{y}_l \right\|^2 + \lambda_1 \sum_{l} \left(\sum_{i=1}^{N} \Omega_1(\boldsymbol{\alpha}_{l,i}) \right) + \lambda_2 \Omega_2(\mathbf{D}_L) \tag{8}$$

$\mathbf{D}_L$ is the local convolutional dictionary, which has n rows and m columns. $\boldsymbol{\alpha}_{l,i}$, which has m rows, is the sparse coding of each component i of each sample l. $\mathbf{P}_i^T$, which has N rows and n columns, is the operator that puts $\mathbf{D}_L \boldsymbol{\alpha}_{l,i}$ in the ith position and pads the rest of the entries with zeros. $\mathbf{y}_l$ is the observed signal. $\|\cdot\|$ has different meanings; when the interior is a vector, $\|\cdot\|$ represents the ℓ_2-norm, and when the interior is a matrix, $\|\cdot\|$ represents the Frobenius norm. λ_1 and λ_2 are hyperparameters. Ω_1 is the sparse constraint imposed on the column vectors with the ℓ_0 norm or ℓ_1 norm, which is defined as follows: $\Omega_1(\mathbf{x}) = \|\mathbf{x}\|_0$ or $\Omega_1(\mathbf{x}) = \|\mathbf{x}\|_1$. Ω_2 is the indicator function of unit-norm sphere.

To solve the above convex or nonconvex CDL optimization problem (8) via local processing, we use the inertial forward-backward splitting framework. An inertial forward-backward splitting algorithm via local processing called the inertial proximal gradient method (IPGM) algorithm is proposed.

To form the objective function of the proposed CDL optimization problem via local processing with the developed framework, based on the proposed Equation (8), supposing

that $\mathbf{x} = \left(\mathbf{D}_L^t, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right)$, $\{\mathbf{x}^t\}$ can be generated by iteratively implementing the following equation:

$$\mathbf{x}^{t+1} \leftarrow \ prox\Big((\mathbf{x}^t - \eta_t \nabla f(\mathbf{x}^t) + \xi_t(\mathbf{x}^t - \mathbf{x}^{t-1})\Big) \tag{9}$$

where prox refers to the proximal mapping operation.

f and g are, respectively defined as follows:

$$f(\mathbf{D}_L, \{\boldsymbol{\alpha}_{l,i}\}) = \frac{1}{2}\sum_l \left\| \sum_{i=1}^N \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_{l,i} - \mathbf{y}_l \right\|^2 \tag{10}$$

$$g(\mathbf{D}_L, \{\boldsymbol{\alpha}_{l,i}\}) = \sum_l \left(\sum_{i=1}^N \lambda_1 \Omega_1(\boldsymbol{\alpha}_{l,\mathrm{i}}) \right) + \lambda_2 \Omega_2(\mathbf{D}_L) \tag{11}$$

To generate the sequence $\{\mathbf{x}^t\} = \left\{\mathbf{D}_L^t, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right\}$ using iterative Equation (9) based on inertial forward-backward splitting, we first need to derive the gradient of f and the proximal mapping of g.

Since f is a function of the composite variable $\mathbf{x}$ or $\left(\mathbf{D}_L, \left\{\boldsymbol{\alpha}_{l,i}\right\}\right)$, we define the gradient of f as follows:

$$\nabla f := \left\{\nabla_{\mathbf{D}_L} f, \left\{\nabla_{\boldsymbol{\alpha}_{l,i}} f\right\}\right\} \tag{12}$$

$\nabla_{\mathbf{D}_L} f$ and $\nabla_{\boldsymbol{\alpha}_{l,i}} f$ can be computed as follows:

$$\nabla_{\mathbf{D}_L} f = \sum_l \sum_{i=1}^N \left(\mathbf{P}_i \left(\sum_{i=1}^N \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_{l,i} - \mathbf{y}_l \right) \boldsymbol{\alpha}_{l,i}^T \right) \tag{13}$$

$$\nabla_{\boldsymbol{\alpha}_{a,b}} f = \mathbf{D}_L^T \mathbf{P}_b \left(\sum_{i=1}^N \mathbf{P}_i^T \mathbf{D}_L \boldsymbol{\alpha}_{a,i} - \mathbf{y}_a \right) \tag{14}$$

To show the result of the descent process, we use an intermediate variable $\left(\mathbf{D}_L^{t+1/2}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1/2}\right\}\right)$:

$$\mathbf{D}_L^{t+1/2} := \mathbf{D}_L^t - \eta_t \nabla_{\mathbf{D}_L} f + \xi_t \left(\mathbf{D}_L^t - \mathbf{D}_L^{t-1} \right) \tag{15}$$

$$\boldsymbol{\alpha}_{l,i}^{t+1/2} := \boldsymbol{\alpha}_{l,i}^t - \eta_t \nabla_{\boldsymbol{\alpha}_{l,i}} f + \xi_t \left(\boldsymbol{\alpha}_{l,i}^t - \boldsymbol{\alpha}_{l,i}^{t-1} \right) \tag{16}$$

where η_t is a step size or descent parameter and ξ_t is an inertial parameter. Therefore, we compute the proximal mapping of g at $\left(\left\{\mathbf{D}_L^{t+1/2}\right\}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1/2}\right\}\right)$ as follows:

$$\left(\mathbf{D}_L^{t+1}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1}\right\}\right) = prox_{\eta_t g}\left(\mathbf{D}_L^{t+1/2}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1/2}\right\}\right) = \left(\left\{prox_{\eta_t \lambda_2 \Omega_2}\left(\mathbf{D}_L^{t+1/2}\right)\right\}, \left\{prox_{\eta_t \lambda_1 \|\cdot\|_0}\left(\boldsymbol{\alpha}_{l,\mathrm{i}}^{t+1/2}\right)\right\}\right) \tag{17}$$

The function prox refers to the proximal mapping operation, which is defined with the following form:

$$prox_{\eta p}(\mathbf{u}) = \underset{v}{\operatorname{argmin}} \frac{1}{2\eta} \|\mathbf{u} - \mathbf{v}\|^2 + p(\mathbf{v}) \tag{18}$$

Furthermore, in the inertial forward-backward splitting framework, to accelerate the convergence rate, we propose using η_t to fit the local curvature of f in each iteration and capturing the local curvature f by estimating the local Lipschitz constant of the CDL problem via local processing. This constant is defined as follows:

$$L_t = \frac{\sqrt{\|\nabla f(\mathbf{x}^t) - \nabla f(\mathbf{x}^{t-1})\|^2}}{\sqrt{\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2}} \tag{19}$$

However, the direct derivation of η_t from L_t does not satisfy the convergence requirement. It is suggested to insert a backtracking scheme in the inertial forward-backward splitting framework to restore its convergence.

We introduce a sequence $\{\tau_t\}$, where $\tau_t > 1$ holds for all t, to represent the adaptive parameter such that the sequence $\{\mathbf{x}^t\}$ satisfies the following equation:

$$f\left(\mathbf{x}^{t+1}\right) \leq f\left(\mathbf{x}^t\right) + \left\langle \nabla f\left(\mathbf{x}^t\right), \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \frac{\tau_t L_t}{2} \left\| \mathbf{x}^{t+1} - \mathbf{x}^t \right\|^2 \tag{20}$$

We assume that each step size η_t is maintained by τ_t using the inequality $0 < \eta_t < (1/(\tau_t L_t))$ and that (20) can be reformulated as follows:

$$\begin{aligned} f\left(\mathbf{x}^{t+1}\right) &= f\left(\mathbf{D}_L^{t+1}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1}\right\}\right) \leq f\left(\mathbf{x}^t\right) + \left\langle \nabla f\left(\mathbf{x}^t\right), \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \tfrac{1}{2\eta_t} \left\| \mathbf{x}^{t+1} - \mathbf{x}^t \right\|^2 \\ &= f\left(\mathbf{D}_L^t, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right) + \sum_{p,q} \left[\boldsymbol{\alpha}_{p,q}^{t+1} \odot \left(\nabla_{\boldsymbol{\alpha}_{p,q}} f\right)\left(\mathbf{D}_L^t, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right) + \tfrac{1}{2\eta_t} \Gamma_r \left(\left(\boldsymbol{\alpha}_{p,q}^{t+1} - \boldsymbol{\alpha}_{p,q}^t\right)^{\mathbf{T}} \left(\boldsymbol{\alpha}_{p,q}^{t+1} - \boldsymbol{\alpha}_{p,q}^t\right) \right) \right] \\ &+ \mathbf{D}_L^{t+1} \odot \left(\nabla_{\mathbf{D}_L} f\right)\left(\mathbf{D}_L^t, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right) + \tfrac{1}{2\eta_t} \Gamma_r \left(\left(\mathbf{D}_L^{t+1} - \mathbf{D}_L^t\right)^{\mathbf{T}} \left(\mathbf{D}_L^{t+1} - \mathbf{D}_L^t\right) \right) \end{aligned} \tag{21}$$

where $\odot$ stands for the Hadamard product.

We present the IPGM algorithm that solves the proposed CDL problem via local processing using inertial forward–backward splitting in Algorithm 1.

Algorithm 1 The IPGM algorithm for solving the proposed CDL problem via local processing

Input: initial dictionary $\mathbf{D}_L^0$; initial dictionary $\mathbf{D}_L^{-1} = \mathbf{D}_L^0$; initial coefficients map $\left\{\boldsymbol{\alpha}_{l,i}^0\right\}$; initial coefficient maps $\left\{\boldsymbol{\alpha}_{l,i}^{-1}\right\} = \left\{\boldsymbol{\alpha}_{l,i}^0\right\}$; adaptive parameter $\tau > 1$; choose $c_1 > 0$ close to 0; inertial parameter $\xi_0 \in \left[0, \frac{1}{2}\right)$; and descent parameter $1 < \eta_0 < (1 - 2\xi_0)/\tau L$, where L is the Lipschitz constant of ∇f.

Output: learned dictionary $\mathbf{D}_L^t$; learned coefficients $\left\{\boldsymbol{\alpha}_{l,i}^t\right\}$;

1: **Initialization:** $t \leftarrow 0$
2: **repeat**
3:
if $t < 2$, then
4:
$\eta_t \leftarrow \eta_0$, $\xi_t \leftarrow \xi_0$
5:
else
6:
compute L_t using Equation (19)
7:
$\eta_t \leftarrow \frac{1}{L_t}$
8:
end if
9:
$k \leftarrow 0$
10: **repeat**
11:
compute $\nabla_{\mathbf{D}_L} f$ and $\nabla_{\boldsymbol{\alpha}_{l,i}} f$ using Equations (13) and (14)
12:
compute $\left(\mathbf{D}_L^{t+1/2}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1/2}\right\}\right)$ using Equations (15) and (16)
13:
update $\left(\mathbf{D}_L^{t+1}, \left\{\boldsymbol{\alpha}_{l,i}^{t+1}\right\}\right)$ using Equation (17)

Algorithm 1 *Cont.*

14:
$\tau_t \leftarrow \tau^k$
15:
$\eta_t \leftarrow \frac{\eta_t}{\tau}$
16:
$k \leftarrow k+1$
17:
until Equation (21) is satisfied
18:
$\eta_t \geq c_1,\ \xi_t \geq 0$
19:
$\delta_t := \frac{1}{2\eta_t} - \frac{\tau_t L_t}{2} - \frac{\xi_t}{2\eta_t}$
20:
$\gamma_t := \frac{1}{2\eta_t} - \frac{\tau_t L_t}{2} - \frac{\xi_t}{\eta_t}$
21:
until $\delta_t \geq \gamma_t \geq c_1$ and $(\delta_t)_{t=0}^{\infty}$ is monotonically decreasing
22:
$t \leftarrow t+1$
23: **until** convergence
24: **return** $\left\{\{\mathbf{D}_L^t\}, \left\{\boldsymbol{\alpha}_{l,i}^t\right\}\right\}$

4. Computational Complexity Analysis

The computational complexity of our algorithms is discussed as follows. We assume the following. N is the signal dimension. I is the number of signals. n is the patch size. m is the number of filters. k is the maximal number of nonzeros in a needle $\boldsymbol{\alpha}_{l,i}$, which is very sparse $k \ll m$. C is the number of backtracking loops, which is usually very small. The convolution is performed by the local processing operation. The operation $\mathbf{P}_i^{\mathbf{T}}$ is only an operator of a column to an image.

The dominant computational complexity of the signal reconstruction performed by the local processing operation is approximately $O(INnk)$. The computation of the signal residual requires $O(IN)$ operations. Neglecting some minor factors, the computational complexity of the gradient of needles and dictionary performed by the local processing operation is effectively $O(INnm)$ and $O(INnk)$, respectively. In addition, the inertia term includes an addition and a multiplication for each iteration which is negligible.

5. Proof of the Convergence of Algorithm 1

Before describing the convergence theorem, let us analyze the relevant lemmas and hypotheses.

Lemma 1. *(The abstract convergence result for KL functions) This convergence result is based on three abstract conditions for a sequence* $\left(\mathbf{z}^t\right)_{t\in\mathbb{N}} := \left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right)_{t\in\mathbb{N}} \in \mathbb{R}^{2N}, \mathbf{x}^t \in \mathbb{R}^N,\ \mathbf{x}^{t-1} \in \mathbb{R}^N$. *We fix two positive constants a > 0 and b > 0 and consider a proper lower semicontinuous function* $F:\ \mathbb{R}^{2N} \rightarrow\ \mathbb{R} \cup \{\infty\}$. *Then, the conditions for* $\left(\mathbf{z}^t\right)_{t\in\mathbb{N}}$ *are as follows:*

(**H1**) *For each* $k \in \mathbb{N}$, *it holds that*

$$F\left(\mathbf{z}^{k+1}\right) + a\|\mathbf{x}^k - \mathbf{x}^{k-1}\| \leq F\left(\mathbf{z}^k\right)$$

(**H2**) *For each* $k \in \mathbb{N}$, *there exists a* $\mathbf{w}^{k+1} \in \partial F\left(\mathbf{w}^{k+1}\right)$ *such that*

$$\|\mathbf{w}^{k+1}\| \leq \frac{b}{2}\left(\|\mathbf{x}^k - \mathbf{x}^{k-1}\| + \|\mathbf{x}^{k+1} - \mathbf{x}^k\|\right)$$

(**H3**) *There exists a subsequence* $\left(\mathbf{z}^{k_j}\right)_{j\in\mathbb{N}}$ *such that*

$$\mathbf{z}^{k_j} \to \tilde{\mathbf{z}} \text{ and } F\left(\mathbf{z}^{k_j}\right) \to F\left(\tilde{\mathbf{z}}\right) \text{ as } j \to \infty$$

When F is a KL function and H1, H2, and H3 are satisfied, F satisfies the convergence result.

Lemma 2. *For all* $n \geq 0$, $\delta_t \geq \gamma_t$, $\xi_t \in \left[0, \frac{1}{2}\right)$, *and* $\eta_t \leq (1-2\xi_t)/\tau_t L_t$. *Furthermore, given* $L_t > 0$, *there exists a pair of parameters* η_t *and* ξ_t *such that* $(\delta_t)_{t=0}^{\infty}$ *is monotonically decreasing.*

Proof of Lemma 2. By the algorithmic requirements,

$$\delta_t = \frac{1}{2\eta_t} - \frac{\tau_t L_t}{2} - \frac{\xi_t}{2\eta_t} \geq \frac{1}{2\eta_t} - \frac{\tau_t L_t}{2} - \frac{\xi_t}{\eta_t} = \gamma_t \geq 0 \tag{22}$$

The upper bounds for ξ_t and η_t are obtained by rearranging $\gamma_t \geq c_1$ to $\xi_t \leq (1-\eta_t\tau_t L_t - 2\eta_t c_1)/2$ and $\eta_t \leq (1-2\xi_t)/(2c_1+\tau_t L_t)$, respectively.

The last statement follows by incorporating the descent property of δ_t . Let $\delta_{-1} \geq c_1$ be chosen initially. Then, the decent property of $(\delta_t)_{t=0}^{\infty}$ requires one of the equivalent statements below

$$\delta_{t-1} \geq \delta_t \Leftrightarrow \delta_{t-1} \geq \frac{1}{2\eta_t} - \frac{\tau_t L_t}{2} - \frac{\xi_t}{2\eta_t} \Leftrightarrow \eta_t \geq (1-\xi_t)/(2\delta_{t-1}+\tau_t L_t) \tag{23}$$

to be true. An upper bound on η_t is obtained by

$$\gamma_t \geq c_1 \Leftrightarrow \eta_t \leq \frac{1-2\xi_t}{2c_1+\tau_t L_t} \tag{24}$$

Consider the condition for a nonnegative gap between the upper and lower bounds of η_t:

$$\frac{1-2\xi_t}{2c_1+\tau_t L_t} - \frac{1-\xi_t}{2\delta_{n-1}+\tau_t L_t} \geq 0 \Rightarrow \frac{2\delta_{n-1}+\tau_t L_t}{2c_1+\tau_t L_t} \geq \frac{1-\xi_t}{1-2\xi_t} \tag{25}$$

Defining $b := (2\delta+\tau_t L_t)/(2c_1+\tau_t L_t) \geq 1$, it is easily verified that there exists $\xi_t \in \left[0, \frac{1}{2}\right)$ satisfying the equivalent condition

$$\frac{b-1}{2b-1} \geq \xi_t \tag{26}$$

As a consequence, the existence of a feasible η_t follows, and the decent property for δ_t holds. □

Proposition 1. *(a) The sequence* $\left(H_{\delta_t}\left(\mathbf{x}^t, \mathbf{x}^{t-1}\right)\right)_{n=0}^{\infty}$ *is monotonically decreasing and thus convergent. In particular, it holds that*

$$H_{\delta_{t+1}}\left(\mathbf{x}^{t+1}, x^t\right) \leq H_{\delta_t}\left(\mathbf{x}^t, \mathbf{x}^{t-1}\right) - \gamma_t \boldsymbol{\Delta}_t^2 \tag{27}$$

(b) It holds that $\sum_{n=0}^{\infty} \boldsymbol{\Delta}_t^2 < \infty$, *and thus,* $\lim_{t\to\infty} \boldsymbol{\Delta}_t = 0$.

Proof of Proposition 1. (a) For a more convenient notation, we abbreviate $h = f + g$ in the IPGM algorithm for the nonconvex nonsmooth CDL problem via local processing; this does not mean that the value of the function drops, so we construct a function $H_\delta(\mathbf{x},\ \mathbf{y}) = h(\mathbf{x}) + \delta\|\mathbf{x}-\mathbf{y}\|^2,\ \delta \in \mathbb{R}$. Note that for $\mathbf{x} = \mathbf{y}$, $H_\delta(\mathbf{x},\ \mathbf{y}) = h(\mathbf{x})$.

We show below that $H_\delta(\mathbf{x},\ \mathbf{y})$ satisfies the convergence requirement. In the inertial forward–backward splitting framework, the sequence $\{\mathbf{x}^t\}$ is generated by iteratively implementing Equation (9).

Notably, in the IPGM algorithm, the function f satisfies the adaptive descent formula in (20).

The function g is a nonconvex, nonsmooth, and proper closed function. Combining the iterative mapping of the sequence $\{\mathbf{x}^t\}$ in the inertial forward–backward splitting framework and the definition of the proximal operator, we can obtain

$$\begin{aligned}\tfrac{1}{2\eta_t}\|\mathbf{x}^{t+1}-\left(\mathbf{x}^t-\eta_t\nabla f\left(\mathbf{x}^t\right)+\xi_t\left(\mathbf{x}^t-\mathbf{x}^{t-1}\right)\right)\|^2+g\left(\mathbf{x}^{t+1}\right)\\ \leq \tfrac{1}{2\eta_t}\|\mathbf{x}^t-\left(\mathbf{x}^t-\eta_t\nabla f\left(\mathbf{x}^t\right)+\xi_t\left(\mathbf{x}^t-\mathbf{x}^{t-1}\right)\right)\|^2+g\left(\mathbf{x}^t\right)\end{aligned} \tag{28}$$

Thus, this simplifies to

$$g\left(\mathbf{x}^{t+1}\right)+\frac{1}{2\eta_t}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+\frac{1}{\eta_t}\left\langle\mathbf{x}^{t+1}-\mathbf{x}^t,\eta_t\nabla f\left(\mathbf{x}^t\right)-\xi_t\left(\mathbf{x}^t-\mathbf{x}^{t-1}\right)\right\rangle\leq g\left(\mathbf{x}^t\right) \tag{29}$$

Now, using (20) and (29) by summing both inequalities, it follows that

$$\begin{aligned}h(\mathbf{x}^{t+1})&\leq h(\mathbf{x}^t)-\left(\tfrac{1}{2\eta_t}-\tfrac{\tau_t L_t}{2}\right)\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+\tfrac{\xi_t}{\eta_t}\left\langle\mathbf{x}^{t+1}-\mathbf{x}^t,\mathbf{x}^t-\mathbf{x}^{t-1}\right\rangle\\ &\leq h(\mathbf{x}^t)-\left(\tfrac{1}{2\eta_t}-\tfrac{\tau_t L_t}{2}-\tfrac{\xi_t}{2\eta_t}\right)\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+\tfrac{\xi_t}{2\eta_t}\|\mathbf{x}^t-\mathbf{x}^{t-1}\|^2\end{aligned} \tag{30}$$

where the second line follows from $2\langle\mathbf{A},\mathbf{B}\rangle\leq\|\mathbf{A}\|_2^2+\|\mathbf{B}\|_2^2$ for vectors $\mathbf{A},\mathbf{B}\in\mathbb{R}^N$. Then, a simple rearrangement of the terms yields:

$$h\left(\mathbf{x}^{t+1}\right)+\delta_t\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2\leq h\left(\mathbf{x}^t\right)+\delta_t\|\mathbf{x}^t-\mathbf{x}^{t-1}\|^2-\gamma_t\|\mathbf{x}^t-\mathbf{x}^{t-1}\|^2 \tag{31}$$

which establishes (27) as δ_t is monotonically decreasing. The sequence $\left(H_{\delta_t}\left(\mathbf{x}^t,\mathbf{x}^{t-1}\right)\right)_{n=0}^{\infty}$ monotonically decreases if and only if $\gamma_t\geq 0$, which is confirmed by the algorithmic requirements. By assumption, h is bounded from below by some constant $\underline{h}>-\infty$; hence, $\left(H_{\delta_t}\left(\mathbf{x}^t,\mathbf{x}^{t-1}\right)\right)_{n=0}^{\infty}$ converges.

(b) Summing (27) from $t=0,\cdots,T$ yields (note that $H_{\delta t}\left(\mathbf{x}^0,\mathbf{x}^{-1}\right)=h(\mathbf{x}^0)$)

$$\sum_{t=0}^{T}r_t\boldsymbol{\Delta}_t^2\leq\sum_{t=0}^{T}H_{\delta_t}\left(\mathbf{x}^t,\mathbf{x}^{t-1}\right)-H_{\delta_{t+1}}\left(\mathbf{x}^{t+1},\mathbf{x}^t\right)=h\left(\mathbf{x}^0\right)-H_{\delta_{T+1}}\left(\mathbf{x}^{T+1},\mathbf{x}^T\right)\leq h\left(\mathbf{x}^0\right)-\underline{h}<\infty \tag{32}$$

Letting T tend to ∞, it can be seen from (32) that $\lim_{t\to T}\gamma_t\boldsymbol{\Delta}_t^2=0$ since $\gamma_T\geq c_1>0$ implies the above statement. □

Proposition 2. *(a) The sequence $(h(\mathbf{x}^t))_{t=0}^{\infty}$ converges.*

(b) There exists a convergent subsequence $(\mathbf{x}^{t_k})_{k=0}^{\infty}$.

(c) Any limit point $\mathbf{x}^*:=lim_{k\to\infty}\mathbf{x}^{t_k}$ is a critical point of $h(\mathbf{x})$, and $h(\mathbf{x}^{t_k})\to h(\mathbf{x}^*)$ as $k\to\infty$.

Proof of Proposition 2. (a) This follows from the squeeze theorem as for all $t\geq 0$, the following holds:

$$H_{-\delta_t}\left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right)\ \leq\ h\left(\mathbf{x}^t\right)\ \leq\ H_{\delta_t}\left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right) \tag{33}$$

and due to Propositions 1(a) and (b),

$$\lim_{t\to\infty}H_{-\delta_t}\left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right)=\lim_{t\to\infty}H_{\delta_t}\left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right)-2\delta_t\boldsymbol{\Delta}_n^2=\lim_{t\to\infty}H_{\delta_t}\left(\mathbf{x}^t,\ \mathbf{x}^{t-1}\right) \tag{34}$$

(b) By Proposition 1(a) and the fact that $H_{\delta_0}\left(\mathbf{x}^0,\ \mathbf{x}^{-1}\right)=h\left(\mathbf{x}^0\right)$, it is clear that the whole sequence $(\mathbf{x}^t)_{t=0}^{\infty}$ is contained in the level set $\left\{\mathbf{x}\in\mathbb{R}^N\ :\ h\ \leq\ h(\mathbf{x})\ \leq\ h\left(\mathbf{x}^0\right)\right\}$, which is

bounded due to the coercivity of h and because $h = \inf_{x\in\mathbb{R}^N} h(\mathbf{x}) > -\infty$. Using the Bolzano-Weierstrass theorem, we deduce the existence of a convergent subsequence $(\mathbf{x}^{t_k})_{k=0}^{\infty}$.

(c) To show that each limit point $\mathbf{x}^* = lim_{j\to\infty}\mathbf{x}^{t_j}$ is a critical point of $h(\mathbf{x})$, we recall that the subdifferential is closed. We define

$$\varsigma_j := \frac{\mathbf{x}^{t_j} - \mathbf{x}^{t_j+1}}{\eta_{t_j}} - \nabla f\left(\mathbf{x}^{t_j}\right) + \frac{\xi_{t_j}}{\eta_{t_j}}\left(\mathbf{x}^{t_j} - \mathbf{x}^{t_j-1}\right) + \nabla f\left(\mathbf{x}^{t_j+1}\right). \tag{35}$$

Then, the sequence $\left(\mathbf{x}^{t_j},\ \varsigma_j\right) \in Graph(\partial h)\ := \left\{(\mathbf{x},\ \varsigma) \in \mathbb{R}^N \times \mathbb{R}^N | \varsigma \in \partial h(\mathbf{x})\right\}$. Furthermore, it holds that $\mathbf{x}^* = lim_{j\to\infty}\mathbf{x}^{t_j}$, and due to Proposition 1(b), ∇f is Lipschitz-continuous, and

$$\|\varsigma^j - 0\| \leq \frac{1}{\eta_{t_j}}\mathbf{\Delta}_{n_j+1} + \frac{\xi_{t_j}}{\eta_{t_j}}\mathbf{\Delta}_{n_j} + \|\nabla f\left(\mathbf{x}^{t_j+1}\right) - \nabla f\left(\mathbf{x}^{t_j}\right)\| \tag{36}$$

It holds that $lim_{j\to\infty}\varsigma^j = 0$. It remains to be shown that $lim_{j\to\infty}h\left(\mathbf{x}^{t_j}\right) = h(\mathbf{x}^*)$. By the closure property of the subdifferential ∂h, $(\mathbf{x}^*, 0) \in Graph(\partial h)$, which means that $\mathbf{x}^*$ is a critical point of h.

According to the iterative mapping (Equation (9)) of the sequence $\{\mathbf{x}^t\}$ in the inertial forward-backward splitting framework, we can obtain:

$$\begin{aligned} \tfrac{1}{2\eta_t}\|\mathbf{x}^{t+1} - \left(\mathbf{x}^t - \eta_t\nabla f\left(\mathbf{x}^t\right) + \xi_t\left(\mathbf{x}^t - \mathbf{x}^{t-1}\right)\right)\|^2 + g\left(\mathbf{x}^{t+1}\right) \\ \leq \tfrac{1}{2\eta_t}\|\mathbf{x} - \left(\mathbf{x}^t - \eta_t\nabla f\left(\mathbf{x}^t\right) + \xi_t\left(\mathbf{x}^t - \mathbf{x}^{t-1}\right)\right)\|^2 + g(\mathbf{x}) \end{aligned} \tag{37}$$

which implies that

$$\begin{aligned} g\left(\mathbf{x}^{t_j+1}\right) + \tfrac{1}{\eta_{t_j}}\left\langle \eta_{t_j}\nabla f\left(\mathbf{x}^{t_j}\right) - \xi_t\left(\mathbf{x}^{t_j} - \mathbf{x}^{t_j-1}\right),\ \mathbf{x}^{t_j+1} - \mathbf{x}\right\rangle \\ + \tfrac{1}{2\eta_{t_j}}\left(\|\mathbf{x}^{t_j+1} - \mathbf{x}^{t_j}\|^2 - \|\mathbf{x} - \mathbf{x}^{t_j}\|^2\right) \leq\ g(\mathbf{x}) \end{aligned} \tag{38}$$

Proposition 1(b) and the boundedness of $\eta_{t_j}\nabla f\left(\mathbf{x}^{t_j}\right) - \xi_t\left(\mathbf{x}^{t_j} - \mathbf{x}^{t_j-1}\right)$ yield that $lim\ sup_{j\to\infty}g\left(\mathbf{x}^{t_j}\right) \leq g(\mathbf{x})$. Invoking the lower semicontinuity of g yields $lim_{j\to\infty}g\left(\mathbf{x}^{t_j}\right) = g(\mathbf{x}^*)$.

Moreover, f is differentiable and continuous, thus, $lim_{j\to\infty}f\left(\mathbf{x}^{t_j}\right) = f(\mathbf{x}^*)$. We imply that $lim_{j\to\infty}h\left(\mathbf{x}^{t_j}\right) = h(\mathbf{x}^*)$.

Now, using Lemma 1, we can verify the convergence of the sequence $\left(\mathbf{x}^t\right)_{t\in\mathbb{N}}$ generated by Algorithm 1. □

Theorem 1 *(Convergence of the IPGM algorithm to a critical point). Let $\left(\mathbf{x}^t\right)_{t\in\mathbb{N}}$ be generated by Algorithm 1, and let $\delta_t = \delta$ for all $t \in \mathbb{N}$. Then, the sequence $\left(\mathbf{x}^{t+1}, \mathbf{x}^t\right)_{t\in\mathbb{N}}$ satisfies H1, H2, and H3 for the function $H_\delta : \mathbb{R}^{2N} \to \mathbb{R}\cup\{\infty\}$, $(\mathbf{x},\ \mathbf{y}) \mapsto h(\mathbf{x}) + \delta\|\mathbf{x} - \mathbf{y}\|_2^2$. Moreover, if the sequence possesses the KL property at a cluster point, then the sequence has a finite length and is a critical point H_δ; hence, $\mathbf{x}^*$ is a critical point of h.*

Proof of Proposition Theorem 1. First, we prove that the function has the KL property.h is a semialgebraic function. If we consider that $\|\mathbf{x} - \mathbf{y}\|^2 = \sum_{i=1}^{N}[\mathbf{x}(i) - \mathbf{y}(i)]^2$, then $\|\mathbf{x} - \mathbf{y}\|^2$ is a polynomial function. Therefore, it is semialgebraic, so $\delta\|\mathbf{x} - \mathbf{y}\|^2$ is a semialgebraic function. $H_\delta(\mathbf{x},\ \mathbf{y})$ is semialgebraic and has the KL property.

Next, we verify that Assumptions H1, H2, and H3 are satisfied.

Condition H1 is proven in Proposition 1(a) with $a = c_2 \leq \gamma_t$. To prove Condition H2, consider $\mathbf{w}^{t+1} := (\mathbf{w}_x^{t+1}, \mathbf{w}_y^{t+1})^T \in \partial H_\delta(\mathbf{x}^{t+1}, \mathbf{x}^t)$ with $\mathbf{w}_x^{t+1} \in \partial g(\mathbf{x}^{t+1}) + \nabla f(\mathbf{x}^{t+1}) + 2\delta(\mathbf{x}^{t+1} - \mathbf{x}^t)$ and $\mathbf{w}_y^{t+1} = -2\delta(\mathbf{x}^{t+1} - \mathbf{x}^t)$. The Lipschitz continuity of ∇f and the use of (9) to specify an element from $\partial g(\mathbf{x}^{t+1})$ imply that

$$\begin{aligned} \|\mathbf{w}^{t+1}\| &\leq \|\mathbf{w}_x^{t+1}\| + \|\mathbf{w}_y^{t+1}\| \\ &\leq \|\nabla f(\mathbf{x}^{t+1}) - \nabla f(\mathbf{x}^t)\| + \left(\tfrac{1}{\eta_t} + 4\delta\right)\|\mathbf{x}^{t+1} - \mathbf{x}^t\| + \tfrac{\xi_t}{\eta_t}\|\mathbf{x}^t - \mathbf{x}^{t-1}\| \\ &\leq \tfrac{1}{\eta_t}(\eta_t\tau_t L_t + 1 + 4\eta_t\delta)\|\mathbf{x}^{t+1} - \mathbf{x}^t\| + \tfrac{1}{\eta_t}\xi_t\|\mathbf{x}^t - \mathbf{x}^{t-1}\| \end{aligned} \tag{39}$$

As $\delta_t \geq \gamma_t \geq c_1 > 0$, we can obtain $\eta_t\tau_t L_t \leq 1 - 2\xi_t \leq 1$, and $\delta\eta_t = \frac{1}{2} - \frac{\eta_t\tau_t L_t}{2} - \frac{\xi_t}{2} \leq \frac{1}{2}$. Setting $b = \frac{4}{c_1}$ verifies condition H2, i.e., $\|\mathbf{w}^{t+1}\| \leq b(\|\mathbf{x}^t - \mathbf{x}^{t-1}\| + \|\mathbf{x}^{t+1} - \mathbf{x}^t\|)$. Condition H2 is proven.

In Proposition 2(c), it is proven that there exists a subsequence $\left(\mathbf{x}^{t_j+1}\right)_{j\in N}$ of $(\mathbf{x}^t)_{t\in N}$ such that $\lim_{j\to\infty} h\left(\mathbf{x}^{t_j+1}\right) = h(\mathbf{x}^*)$. The following corollary uses the fact that semialgebraic functions have the KL property. Proposition 1(b) shows that $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \to 0$ as $t \to \infty$; hence, $lim_{j\to\infty}\, \mathbf{x}_{j+1}^t = \mathbf{x}^*$. As the term $\delta\|\mathbf{x} - \mathbf{y}\|^2$ is continuous in x and y, we deduce that

$$\lim_{j\to\infty} H(\mathbf{x}^{t_{j+1}}, \mathbf{x}^{t_j}) = \lim_{j\to\infty} h(\mathbf{x}^{t_j+1}) + \delta\|\mathbf{x}^{t_j+1} - \mathbf{x}^{t_j}\| = H(\mathbf{x}^*, \mathbf{x}^*) = h(\mathbf{x}^*) \tag{40}$$

Therefore, Condition H3 is proven.
Now, Theorem 1 concludes the proof. □

Remark 1. *IPGM algorithm is convergent under nonconvex optimization. It is easy to prove the convergence under convex optimization constraints, which is equivalent to a special case of the iPiano algorithm for the CDL problem.*

6. Convergence Rate

We prove a global $O(1/k)$ convergence rate for $\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2$. We first define the error u_N to be the smallest squared ℓ_2 norm value of successive iterations:

$$u_N := \min_{0\leq n\leq N} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 \tag{41}$$

Theorem 2. *Algorithm 1 guarantees that for all $N \geq 0$,*

$$u_N \leq c_1^{-1}\frac{h(\mathbf{x}^0) - h}{N+1} \tag{42}$$

Proof of Proposition Theorem 2. In view of Proposition 1(a) and the definition of γ_N in (21), summing both sides of (26) for $n = 0, \ldots, N$ and using the fact that $\delta_N > 0$ from (21), we obtain

$$h \leq h\left(\mathbf{x}^0\right) - \sum_{n=0}^{N} \gamma_n\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 \leq h\left(\mathbf{x}^0\right) - (N+1)\min_{0\leq n\leq N}\gamma_n\mu_N. \tag{43}$$

As $\gamma_n > c_1$, a simple rearrangement concludes the proof. □

7. Experimental Results and Analysis

In this section, we compare the performance of the proposed IPGM with that of various existing methods with respect to solving CDL problems.

7.1. Experimental Setup

7.1.1. List of Compared Methods

1. SBDL denotes the method proposed in [23] based on convex optimization with an ℓ_1 norm-based sparsity-inducing function that uses slices via local processing.
2. Local block coordinate descent (LoBCoD) denotes the method proposed in [24] based on convex optimization with an ℓ_1 norm-based sparsity-inducing function that utilizes needle-based representation via local processing.
3. The PGM denotes the method proposed in [1] based on convex optimization with an ℓ_1 norm-based sparsity-inducing function that uses the Fast Fourier transform (FFT) operation.
4. The IPGM denotes the method proposed in Algorithm 1 in this paper uses needle-based representation via local processing.

7.1.2. Parameter Settings

The parameter settings used for the comparison methods are described as follows:

1. In SBDL [23], the model parameter is λ, and its initial value is 1. The maximum nonzero value of the LARS algorithm is 5. The filter size of the dictionary is 11*11*100. In addition, special techniques are used in the first few iterations of the dictionary learning process.
2. In LoBCoD [24], the model parameter λ is initialized to 1. The maximum nonzero value of the LARS algorithm is 5. The filter format of the dictionary is 11*11*100. In addition, the dictionary learning process first carries out 40 iterations and then uses the proximal gradient descent algorithm.
3. In the PGM [1] model, the Lipschitz constant L is set to 1000 for fruit, and $\lambda = 1$.
4. In the IPGM, the model parameter L_t is set at 1000 for fruit, and $\lambda = 1$.

7.1.3. Implementation Details

The computations of the SBDL, LoBCoD, and PGM algorithms, as well as the IPGM algorithm, are performed using a PC with an Intel i5 CPU and 12 GB of memory.

7.2. Efficiency of Dictionary Learning Algorithms

7.2.1. Motivation and Evaluation Metrics

The efficiency and stability of objective function are the most important criteria for evaluating a numerical optimization algorithm. The sequence generated by an efficient dictionary learning algorithm converges quickly to the corresponding clustering point with a lower function value. The relevant evaluation indicators are listed below.

1. Final Value of $f + g$ and the Algorithmic Convergence Properties: In the $f + g$ minimization-based analysis method, we use the function value of the generated sequence to evaluate its optimization efficiency and convergence.
2. Computing Time: We compare the computing times of the SBDL, LoBCoD, PGM, and IPGM algorithms and compare the computing efficiency.

7.2.2. Training Data and Experimental Settings

A set of generic grayscale natural images is used as the training data in this experiment. The set includes 10 fruit images (100×100 pixels). It is decomposed into high- and low-frequency components. The means are subtracted. These images are normalized to a range of 0 to 1. The dictionary includes 100 elements, the sizes of which are 11×11. One thousand iterations are carried out for each method. The dataset comes from SBDL [23] and LOBCOD [24], and the experimental process refers to the experimental process in

references SBDL and LOBCOD. Similar to references [23,24], there is a similar trend in the experimental results under different atomic numbers or different sizes. We give the final objection function value representing the convergence properties and computing time representing convergence efficiency using only the conditions of 100 atoms with a size of 11 × 11 from the fruit grayscale natural images. Each of the average results represents the four trials using different initial dictionaries, from which the initial coefficients were derived.

7.2.3. Results

The experimental results are described as follows. The set of training data described above is used to derive the results. Figure 1 shows the functional values of all compared methods in each iteration of the dictionary learning procedure. In addition, Table 1 shows the experimental data yielded by all compared methods over 1000 iterations. Figures 2 and 3 correspond to the trained dictionaries and reconstructed images of all compared methods after 1000 iterations, respectively.

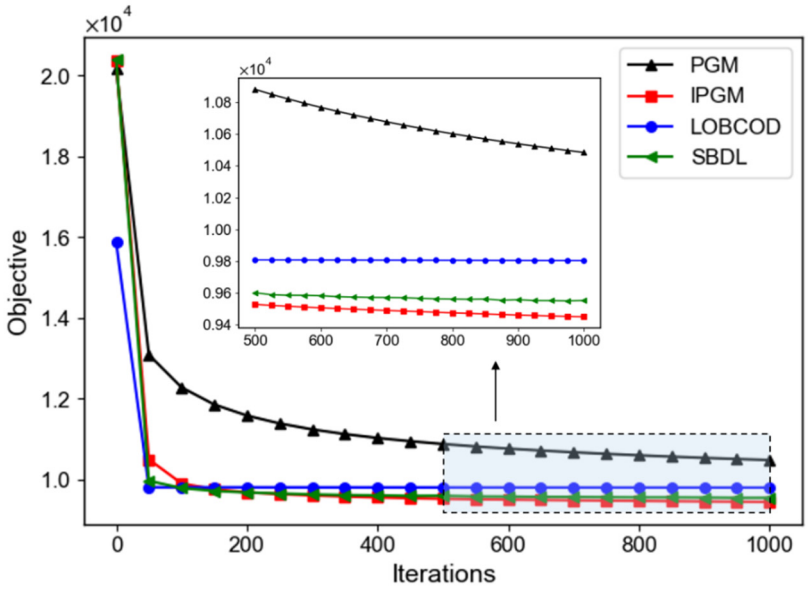
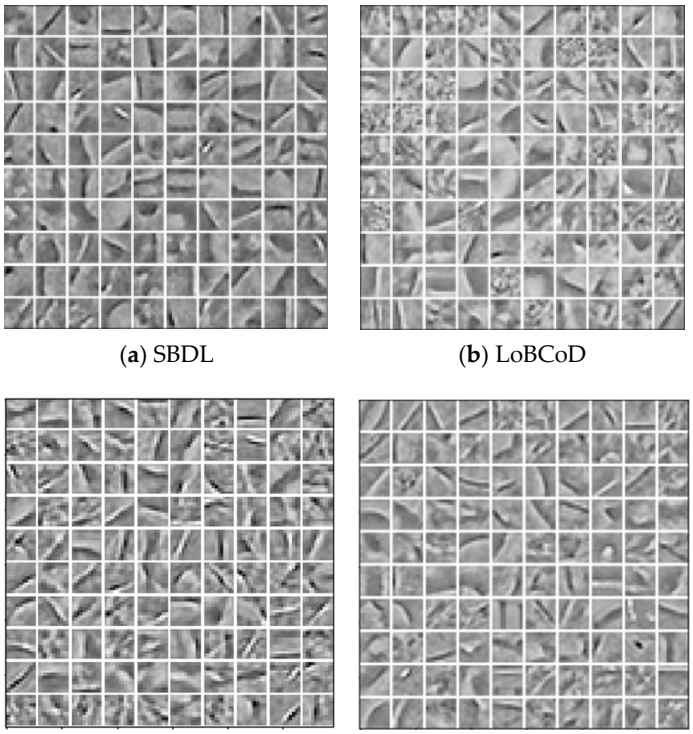


Figure 1. Comparison of the average function values obtained over four trials versus the number of iterations for learning 100 atoms with sizes of 11 × 11 from the grayscale natural fruit images.

Table 1. The experimental results obtained by all compared methods after 1000 iterations with the grayscale natural fruit images.

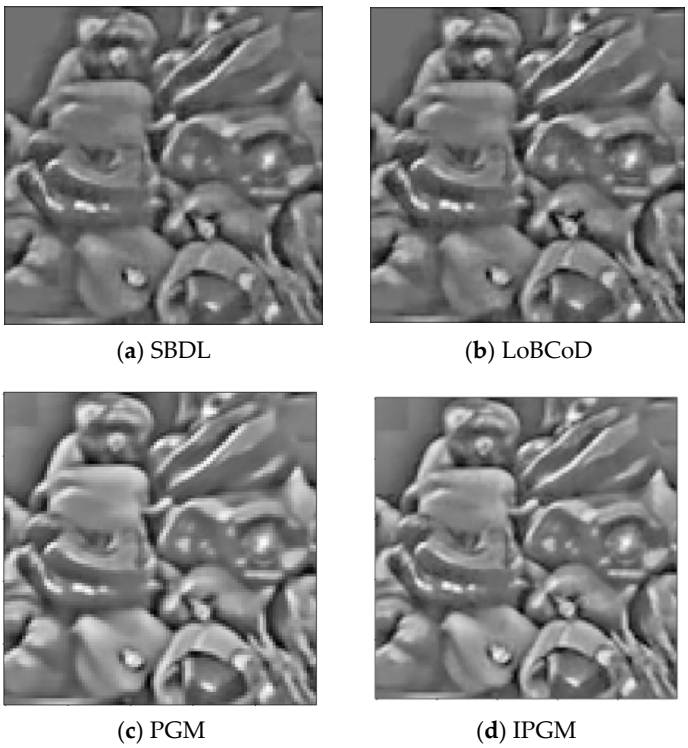
Fruit	SBDL	LoBCoD	PGM	IPGM
Obj	9.545×10^3	9.857×10^3	1.048×10^4	9.449×10^3
Data consistent	3.067×10^3	3.136×10^3	3.528×10^3	3.035×10^3
Regularization	6.478×10^3	6.722×10^3	6.957×10^3	6.415×10^3
Sparsity	0.146%	0.141%	1.696%	0.145%
Time (s)	1110.134	1879.400	1955.365	1222.722
PSNR (dB)	29.348	29.252	28.799	29.438



(**a**) SBDL (**b**) LoBCoD

(**c**) PGM (**d**) IPGM

Figure 2. Dictionaries obtained by the four algorithms after 1000 iterations from the grayscale natural fruit images. (**a**–**d**) is the dictionary obtained by SBDL, LOBCOD, PGM, IPGM, respectively.



(**a**) SBDL (**b**) LoBCoD

(**c**) PGM (**d**) IPGM

Figure 3. Reconstruction images yielded by all compared methods after 1000 iterations from the grayscale natural fruit images. (**a**–**d**) is the reconstruction images yielded by SBDL, LOBCOD, PGM, IPGM, respectively.

7.2.4. Discussion

A discussion of the experimental results is as follows. Table 1 shows the experimental results produced by the four algorithms for the fruit dataset, which shows that the IPGM method yields better results in terms of performances. In terms of the objective function values of the four algorithms, the IPGM algorithm is lowest with 9.449×10^3. In terms of sparsity, except for the poor sparsity of the PGM, there is little difference among the other algorithms. The time consumption of our algorithm is 122.722 s, which is better than that of LoBCoD and the PGM but worse than that of SBDL. The SBDL algorithm calls for the C++ function in the MATLAB program, so it requires less time than the IPGM method. The peak signal-to-noise ratio (PSNR) of the IPGM algorithm is best with 29.438 dB, which is the highest value among the four algorithms. After analyzing Figure 3 above, we see that compared with other algorithms, the IPGM algorithm provides the reconstructed image with the clearest texture details. Generally, the IPGM algorithm has the best performance and the best effect when compared with other algorithms.

7.3. Ablation Experiment

7.3.1. Motivation and Evaluation Metrics

In the proposed IPGM algorithm, the inertia parameters are assigned directly, and they are equivalent to fixed values. Nevertheless, the magnitudes of the inertia parameters have specific impacts on the performance of the algorithm. Therefore, this section uses ablation experiments to analyze the performance of IPGM under different inertia parameters and the performance of IPGM under different optimization conditions.

7.3.2. Training Data and Experimental Settings

The experimental setting of the ablation experiment is consistent with that of the experiment in the previous section. The fruit dataset is also used as the ablation experiment dataset. The dataset comes from SBDL [23] and LOBCOD [24], and the experimental process refers to the experimental process in references SBDL and LOBCOD. We experiment on IPGM algorithms with different inertia parameters under convex optimization constraints and nonconvex optimization constraints.

7.3.3. Results

A discussion of the experimental results is as follows. Table 2 shows the objective function and PSNR values obtained with different inertia parameters under the constraint of nonconvex optimization. Table 3 shows the objective function and PSNR values obtained with different inertia parameters under the constraint of convex optimization.

Table 2. Objective function values and PSNR values of IPGM models with different inertia parameters under the constraint of nonconvex optimization.

	0.1	**0.2**	**0.3**	**0.4**
Obj	1.072×10^4	1.059×10^4	1.097×10^4	1.167×10^4
PSNR	31.960	32.139	32.335	32.734

Table 3. Objective function values and PSNR values of IPGM models with different inertia parameters under the constraint of convex optimization.

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
Obj	9.737×10^3	9.642×10^3	9.604×10^3	9.569×10^3	9.565×10^3	9.543×10^3	9.516×10^3	9.484×10^3	9.449×10^3
PSNR	29.350	29.394	29.401	29.427	29.414	29.422	29.433	29.431	29.438

7.3.4. Discussion

1. The above table summarizes the performance of the IPGM algorithm under different inertia parameter settings. Under nonconvex optimization, with the increase in the inertia parameter, the objective function value decreases, and the PSNR value increases. The effect is the best when the inertia parameter is 0.4. The objective function value is 1.167×10^4, and the PSNR value is 32.734. Under convex optimization, with the increase in the inertia parameter, the value of the objective function decreases, and the value of PSNR increases. The effect is the best when the inertia parameter is 0.9. At this time, the value of the objective function is 9.449×10^3, and the value of PSNR is 29.438. The results show that the larger the inertia parameter setting is, the better the performance of IPGM is under both convex and nonconvex constraints.
2. According to the ranges of inertia parameters allowed under different constraints, the results in the above table are obtained. By comparing Tables 2 and 3, it can be found that with the increase in inertia parameters, the PSNR of the IPGM algorithm under convex optimization constraints is higher, the objective function value is low, and the performance improves. Under the constraint of nonconvex optimization, although the PSNR of the IPGM algorithm is high, the value of the objective function is also high, and the performance is not ideal. This result shows that the simple utilized inertia term cannot achieve the perfect effect under the nonconvex constraint. In the future, we will continue to study the IPGM algorithm with dynamic inertia parameters under nonconvex optimization.

8. Conclusions

For the CDL problem, an IPGM algorithm based on a forward–backward splitting algorithm with an inertial term is proposed. The complexity of the algorithm is analyzed, and the convergence of the algorithm is proven. Finally, the IPGM algorithm is compared with other algorithms in terms of performance and effect through experiments. The results show that the IPGM algorithm produces a lower objective function value, lower sparse value, and higher reconstruction PSNR. In summary, the IPGM algorithm has many good theoretical properties, and it is efficient and straightforward.

In addition, according to the allowable range of inertia parameters under different constraints, the ablation experiment of the IPGM algorithm is carried out. According to the experimental results, it can be found that the IPGM algorithm achieves better performance under convex optimization. However, under the nonconvex optimization constraint, the performance of the IPGM algorithm does not reach the ideal value, indicating that the perfect effect cannot be achieved by simply using the inertia term under the nonconvex constraint. In the future, we will further study the IPGM algorithm with dynamic inertia parameters under nonconvex optimization.

Author Contributions: Supervision, Conceptualization, J.L.; Formal analysis, Methodology, Writing, X.W.; Mathematical aspects, F.W.; Review, Funding acquisition, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Special Projects on Basic Research Cooperation of Beijing, Tianjin and Hebei, grant number 19JCZDJC65600Z, & F2019203583; Central Funds Guiding the Local Science and Technology Development, grant number 206Z5001G; National Natural Science Foundation of China grant number 61473339.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wohlberg, B. Efficient algorithms for convolutional sparse representations. *IEEE Trans. Image Process.* **2016**, *25*, 301–315. [CrossRef] [PubMed]
2. Simon, D.; Elad, M. Rethinking the CSC Model for Natural Images. In Proceedings of the Thirty-third Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 1–8.

3. Zhang, H.; Patel, V.M. Convolutional Sparse and Low-Rank Coding-Based Image Decomposition. *IEEE Trans. Image Process.* **2018**, *27*, 2121–2133. [CrossRef] [PubMed]
4. Yang, L.; Li, C.; Han, J.; Chen, C.; Ye, Q.; Zhang, B.; Cao, X.; Liu, W. Image Reconstruction via Manifold Constrained Convolutional Sparse Coding for Image Sets. *IEEE J. Sel. Top. Signal Process.* **2017**, *11*, 1072–1081. [CrossRef]
5. Bao, P.; Sun, H.; Wang, Z.; Zhang, Y.; Xia, W.; Yang, K.; Chen, W.; Chen, M.; Xi, Y.; Niu, S.; et al. Convolutional Sparse Coding for Compressed Sensing CT Reconstruction. *IEEE Trans. Med. Imaging* **2019**, *38*, 2607–2619. [CrossRef] [PubMed]
6. Hu, X.-M.; Heide, F.; Dai, Q.; Wetzstein, G. Convolutional Sparse Coding for RGB+NIR Imaging. *IEEE Trans. Image Process.* **2018**, *27*, 1611–1625. [CrossRef] [PubMed]
7. Zhu, Y.; Lucey, S. Convolutional Sparse Coding for Trajectory Reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 529–540. [CrossRef]
8. Annunziata, R.; Trucco, E. Accelerating Convolutional Sparse Coding for Curvilinear Structures Segmentation by Refining SCIRD-TS Filter Banks. *IEEE Trans. Med Imaging* **2016**, *35*, 2381–2392. [CrossRef]
9. Gu, S.; Zuo, W.; Xie, Q.; Meng, D.; Feng, X.; Zhang, L. Convolutional sparse coding for image super-resolution. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1823–1831.
10. Wang, J.; Xia, J.; Yang, Q.; Zhang, Y. Research on Semi-Supervised Sound Event Detection Based on Mean Teacher Models Using ML-LoBCoD-NET. *IEEE Access* **2020**, *8*, 38032–38044. [CrossRef]
11. Chen, S.; Billings, S.A.; Luo, W. Orthogonal least squares methods and their application to non-linear system identification. *Int. J. Control* **1989**, *50*, 1873–1896. [CrossRef]
12. Chen, S.S.; Donoho, D.L.; Saunders, M.A. Atomic Decomposition by Basis Pursuit. *SIAM Rev.* **2001**, *43*, 129–159. [CrossRef]
13. Aharon, M.; Elad, M.; Bruckstein, A. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process.* **2006**, *54*, 4311–4322. [CrossRef]
14. Engan, K.; Aase, S.O.; Husoy, J.H. Method of optimal directions for frame design. In Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, Phoenix, AZ, USA, 15–19 March 1999; Volume 5, pp. 2443–2446.
15. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G. Online dictionary learning for sparse coding. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 689–696.
16. Sulam, J.; Ophir, B.; Zibulevsky, M.; Elad, M. Trainlets: Dictionary learning in high dimensions. *IEEE Trans. Signal Process.* **2016**, *64*, 3180–3193. [CrossRef]
17. Wright, J.; Yang, A.Y.; Ganesh, A.; Sastry, S.S.; Ma, Y. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 210–227. [CrossRef] [PubMed]
18. Papyan, V.; Sulam, J.; Elad, M. Working locally thinking globally: Theoretical guarantees for convolutional sparse coding. *IEEE Trans. Signal Process.* **2017**, *65*, 5687–5701. [CrossRef]
19. Garcia-Cardona, C.; Wohlberg, B. Convolutional dictionary learning: A comparative review and new algorithms. *IEEE Trans. Comput. Imaging* **2018**, *4*, 366–381. [CrossRef]
20. Bristow, H.; Eriksson, A.; Lucey, S. Fast convolutional sparse coding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 13–28 June 2013; pp. 391–398.
21. Heide, F.; Heidrich, W.; Wetzstein, G. Fast and flexible convolutional sparse coding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 513–5143.
22. Rey-Otero, I.; Sulam, J.; Elad, M. Variations on the Convolutional Sparse Coding Model. *IEEE Trans. Signal Process.* **2020**, *68*, 519–528. [CrossRef]
23. Papyan, V.; Romano, Y.; Sulam, J.; Elad, M. Convolutional dictionary learning via local processing. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5306–5314.
24. Zisselman, E.; Sulam, J.; Elad, M. A Local Block Coordinate Descent Algorithm for the Convolutional Sparse Coding Model. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 8200–8209.
25. Peng, G. Adaptive ADMM for Dictionary Learning in Convolutional Sparse Representation. *IEEE Trans. Image Process.* **2019**, *28*, 3408–3422. [CrossRef] [PubMed]
26. Elad, P.; Raja, G. Matching Pursuit Based Convolutional Sparse Coding. In Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 6847–6851.
27. Chun, I.I.Y.; Fessler, J. Convolutional dictionary learning: Acceleration and convergence. *IEEE Trans. Image Process.* **2017**, *27*, 1697–1712. [CrossRef] [PubMed]
28. Peng, G. Joint and Direct Optimization for Dictionary Learning in Convolutional Sparse Representation. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 559–573. [CrossRef]
29. Attouch, H.; Bolte, J.; Svaiter, B.F. Convergence of descentmethods for semi-algebraic and tame problems: Proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Math. Program.* **2013**, *137*, 91–129. [CrossRef]
30. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *USSR Comput. Math. Math. Phys.* **1964**, *4*, 1–17. [CrossRef]
31. Ochs, P.; Chen, Y.; Brox, T.; Pock, T. iPiano: Inertial proximal algorithm for nonconvex optimization. *SIAM J. Imaging Sci.* **2014**, *7*, 1388–1419. [CrossRef]
32. Rockafellar, R.T. Monotone Operators and the Proximal Point Algorithm. *SIAM J. Appl. Math.* **1976**, *14*, 877–898. [CrossRef]

33. Moursi, W.M. The Forward-Backward Algorithm and the Normal Problem. *J. Optim. Theory Appl.* **2018**, *176*, 605–624. [CrossRef]
34. Franci, B.; Staudigl, M.; Grammatico, S. Distributed forward-backward (half) forward algorithms for generalized Nash equilibrium seeking. In Proceedings of the 2020 European Control Conference (ECC), Saint-Petersburg, Russia, 12–15 May 2020; pp. 1274–1279.
35. Molinari, C.; Peypouquet, J.; Roldan, F. Alternating forward–backward splitting for linearly constrained optimization problems. *Optim. Lett.* **2020**, *14*, 1071–1088. [CrossRef]
36. Guan, W.B.; Song, W. The forward–backward splitting method and its convergence rate for the minimization of the sum of two functions in Banach spaces. *Optim. Lett.* **2021**, *15*, 1735–1758. [CrossRef]
37. Abass, H.A.; Izuchukwu, C.; Mewomo, O.T.; Dong, Q.L. Strong convergence of an inertial forward-backward splitting method for accretive operators in real Hilbert space. *Fixed Point Theory* **2020**, *21*, 397–412. [CrossRef]
38. Bot, R.I.; Grad, S.M. Inertial forward–backward methods for solving vector optimization problems. *Optimization* **2018**, *67*, 1–16. [CrossRef] [PubMed]
39. Ahookhosh, M.; Hien, L.T.K.; Gillis, N.; Patrinos, P. A block inertial Bregman proximal algorithm for nonsmooth nonconvex problems with application to symmetric nonnegative matrix tri-factorization. *J. Optim. Theory Appl.* **2020**, *190*, 234–258. [CrossRef]
40. Xu, J.; Chao, M. An inertial Bregman generalized alternating direction method of multipliers for nonconvex optimization. *J. Appl. Math. Comput.* **2021**, 1–27. [CrossRef]
41. Pock, T.; Sabach, S. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems. *SIAM J. Imaging Sci.* **2016**, *9*, 1756–1787. [CrossRef]