A Learned Proximal Alternating Minimization Algorithm and Its Induced Network for a Class of Two-Block Nonconvex and Nonsmooth Optimization

Yunmei Chen[1] · **Lezhi Liu**[1] · **Lei Zhang**[1]

Abstract

This work proposes a general learned proximal alternating minimization algorithm, LPAM, for solving learnable two-block nonsmooth and nonconvex optimization problems. We tackle the nonsmoothness by an appropriate smoothing technique with automatic diminishing smoothing effect. For smoothed nonconvex problems we modify the proximal alternating linearized minimization (PALM) scheme by incorporating the residual learning architecture, which has proven to be highly effective in deep network training, and employing the block coordinate decent (BCD) iterates as a safeguard for the convergence of the algorithm. We prove that there is a subsequence of the iterates generated by LPAM, which has at least one accumulation point and each accumulation point is a Clarke stationary point. Our method is widely applicable as one can employ various learning problems formulated as two-block optimizations, and is also easy to be extended for solving multi-block nonsmooth and non-convex optimization problems. The network, whose architecture follows the LPAM exactly, namely LPAM-net, inherits the convergence properties of the algorithm to make the network interpretable. As an example application of LPAM-net, we present the numerical and theoretical results on the application of LPAM-net for joint multi-modal MRI reconstruction with significantly under-sampled k-space data. The experimental results indicate the proposed LPAM-net is parameter-efficient and has favourable performance in comparison with some state-of-the-art methods.

✉ Yunmei Chen
yun@ufl.edu

Lezhi Liu
lliu1@ufl.edu

Lei Zhang
leizhang@ufl.edu

1 Department of Mathematics, University of Florida, Gainesville, FL 32611, USA

1 Introduction

Recent years have witnessed remarkable success of deep learning across various real-world applications. However, a purely data-driven approach may fail to approximate the desired functions, especially when training data are scarce. It is well known that the scarcity of data leads to overfitting and challenges in interpretation. To mitigate these issues, the unrolling/unfolding neural networks (UNNs) have been developed and have shown promising results in solving inverse problems arising from computer vision and medical imaging. The UNNs are multi-phase neural networks, in which each phase mimics one iteration of the optimization algorithms for solving the inverse problems. However, despite their promising performance in practice, most of the existing UNNs only superficially resemble steps of optimization algorithms. Consequently, their outputs do not really yield solutions to any interpretable variational models. This results is lack of theoretical justifications and convergence guarantees for their outputs.

Recently a novel class of unrolling methods known as learned optimization algorithms (LOA) [7, 10, 35] has been developed. The goal of LOA is to tackle the challenges associated with solving a class of inverse problems arising from various image reconstruction and synthesis problems. These LOAs strategically combine the learned variational model with the prior domain knowledge of underlying physical processes to enhance its interpretability. These methods also utilize optimization to guarantee the convergence.

The optimization scheme induces a highly structured deep network, aligning its architecture precisely with the iterative algorithm. Hence, the network inherits the convergence property of the algorithm, and it outputs an approximation of the solution to the variational model. As a result, it is interpretable and parameter efficient. However, the existing LOAs only consider a single block of variables.

Motivated by a wide range of applications in machine learning that involve multi-modalities or multi-domains problems, such as multi-task learning, transfer learning, multi-modal learning and fusion, dual domain image reconstruction, and image synthesis, in this work we extend the idea of LDA to deal with learned multi-block nonconvex and nonsmooth optimization problems. More specifically, we develop a novel convergent learned proximal alternating minimization (LPAM) algorithm to solve the following class of learnable optimization problems:

$$(M_1): \quad \text{minimize} \quad \Phi(\mathbf{x}_1, \mathbf{x}_2; \Theta) := H_1(\mathbf{x}_1; \theta_1) + H_2(\mathbf{x}_2; \theta_2) + H(\mathbf{x}_1, \mathbf{x}_2; \theta),$$
$$(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m, \tag{1}$$

where each learnable function is a parameterized function. $\Theta = (\theta_1, \theta_2, \theta)$ is the set of learned parameters. Each function in (1) is possibly nonconvex and nonsmooth. We shall also study the convergence and iteration complexity of the algorithm, and provide experimental results on the application on the joint multi-contrast MRI reconstruction with significantly under-sampled data.

2 Related Work

2.1 Related Work on Alternating Minimization Algorithms:

Alternating minimization (AM) or "two block" coordinate minimization (BCM) algorithm plays an indispensable role in solving a rich class of problems formulated as follows:

$$(M): \quad \text{minimize} \quad \Phi(\mathbf{x}_1, \mathbf{x}_2) := H_1(\mathbf{x}_1) + H_2(\mathbf{x}_2) + H(\mathbf{x}_1, \mathbf{x}_2), \quad (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m.$$

$$(2)$$

The standard approach to solve this problem is the so-called Gauss-Seidel iteration scheme, also known as block coordinate descent (BCD) method. This scheme alternately keeps one block of updated variable fixed and optimize the other block [6, 28]. With assumptions on convexity and smoothness, several convergence results have been established [5, 9].

Inspired by many practical problems arising from machine learning and image processing, such as non-negative matrix factorization, blind image deconvolution, low rank matrix completion [19, 21], phase retrieval problem [27], interference alignment[29], and compressed sensing using synthesis sparsity models[1] joint multi-modal image reconstruction[32], the AM algorithms for nonconvex (including biconvex) and nonsmooth minimization have attracted great interest. One of the effective approaches to solve nonconvex and nonsmooth problems (2) is the proximal alternating minimization (PAM) algorithms. The PAM algorithm developed in [3] can be viewed as a proximal regularization of a two block Gauss-Seidel method for solving (2). Under the assumption that the objective function satisfies Kurdyka-Lojasiewicz (KL) property, the convergence to a critical point of Φ of each bounded sequence generated by the algorithm has been obtained in [4]. However, PAM algorithm requires exact minimization of a nonconvex and nonsmooth problem in each step. To overcome this difficulty, the pioneer work [8] introduced a proximal alternating linearized minimization (PALM) algorithm, which can be viewed as alternating the steps of the proximal forward-backward scheme. Building on the KL property, it was proved that each bounded sequence generated by PALM globally converges to a critical point. Later, the work [30] proposed an inertial version of the PALM (iPALM) algorithm motivated from the Heavy Ball method of Polyak [31]. It is also closely related to multi-step algorithms of Nesterov's accelerated gradient method [26]. The global convergence to a critical point is also obtained for iPALM under the assumption that the objective function Φ satisfies the KL property and the same smoothness conditions as that for PALM.

2.2 Related Work on Unrolling Neural Networks Inspired by AM Algorithms:

In recent years, the unrolling neural networks inspired by AM algorithms unroll the iteration scheme of an AM algorithm into a multi-phase deep neural network. Several unrolling convolutional neural networks (CNN) based on AM algorithms have been developed for solving inverse problems in computer vision including the unrolling neural networks resembling the steps of alternating direction method of multipliers (ADMM) algorithm, and block coodinate decent (BCD) algorithm. The ADMM-net [33] with the architecture derived from ADMM has been applied to compressive sensing MRI reconstruction. The deeply aggregated alternating minimization neural network (DeepAM) [24] for image restoration uses a data-driven approach in the energy minimization framework. At each iteration, the DeepAM advances the following two steps in the conventional AM algorithm. One step uses a data-driven approach to update the auxiliary variable representing the gradient of the image by a CNN that plays the role of regularization. The other step recovers the image by minimizing the data fidelity and the disparity between the gradient of the image and updated auxiliary variable. Moreover, several versions of BCD-nets that resemble the steps of BCD algorithm have been successfully applied to solve the inverse problems for signal/image recovery/reconstruction [2, 11–14].

The common idea of those BCD-nets is combining a denoising CNN in defining the regularization as one module, namely a denoising module, and a model based image reconstruction (MBIR) as the other module into the BCD framework. The MBIR module enhances the data fidelity for the image from the denoising module. For instance, at each iteration of the BCD-net proposed in [14], it trains an image mapping CNN using identical convolutional kernels in both encoders and decoders as one module in a BCD framework for image recovery. The other module in this BCD-net minimizes an objective function consisting of the data fidelity and penalty between to be updated image and the deniosed image from image mapping CNN. This BCD-Net [14] is modified from [13], where a sparsity promoting denoising network is trained in denoising modules and the accelerated proximal gradient method using a majorizer is applied to MBIR modules with statistical CT data-fidelity for low-dose CT reconstruction. Furthermore, to achieve fast and convergent solutions for the inverse problem in image processing, the work in [12] proposes a Momentum-Net. The Momentum-Net is motivated by applying the block proximal extrapolated gradient method using a Majorizer and convolutional autoencoders to MBIR. Each iteration of Momentum-Net consists of three core modules: image refining, extrapolation and MBIR. The image refining module is a denoising module that trains an image refining CNN to remove iteration-wise artifact. The extrapolation module uses momentum from previous updates to amplify the changes in subsequent iterations and accelerate convergence. The MBIR is non-iterative, since the MBIR problem in consideration of this work has a closed-form solution. Momentum-Net guarantees convergence to a fixed-point for general differentiable nonconvex data-fit terms and convex feasible sets, under the conditions that the sequence of paired refining CNNs is asymptotically nonexpansive and the refined image from the image refining module gives an asymptotically block-coordinate minimizer.

Very recently, [15] presented a learned alternating algorithm aimed at solving a specific variational model for dual-domain sparse-view CT reconstruction. In this model, the joint term is convex and smooth, and the regularization terms are learnable in both the image and sinogram domains, respectively. In this study, we extend their work by considering a broader scenario, where each term in the objective function can be nonconvex and nonsmooth. We provide sufficient conditions for taking smoothing approximations of nonsmooth objective functions. We also provide the analysis of the iteration complexity in addition to the convergence analysis.

3 Contributions and Organization

In this work we propose a novel learned proximal alternating minimization algorithm addressing nonsmooth and nonconvex two-block variational models with provable convergence and iteration complexity. The proposed LPAM algorithm for solving (M_1) determines the architecture of the deep neural network, hence the design of the LPAM considers not only the convergence and efficiency, but also the ability to assist the training of the network parameters Θ, i.e., reducing the error in minimizing the loss function.

Our main idea for designing this algorithm is as follows: (i) We tackle the nonsmoothness by an appropriate smoothing technique with automatic diminishing smoothing effect; (ii) We modify the PALM scheme by incorporating the residual learning architecture, which has proven to be highly effective in deep network training [20]; and (iii) When some modified PALM iterates fail to satisfy certain conditions, we employ the block coordinate decent (BCD) iterates as a safeguard to ensure convergence. Moreover, we prove that a subsequence

generated by the proposed LPAM has accumulation points and all of them are Clarke stationary points of the nonsmooth nonconvex problem.This algorithm can be readily extended to multi-block nonsmooth and nonconvex optimization.

This algorithm significantly broadens its applicability, as it relaxes the strict convergence conditions of the existing algorithms. The PALM or iPALM algorithm assumes the joint term $H(\mathbf{x}_1, \mathbf{x}_2)$ to be smooth and possibly nonconvex, and the proximal points associated with $H_i(\mathbf{x}_i)$ $(i = 1, 2)$ are easy to obtain. Their global convergence results are built on KL property of the objective function. These assumptions in general cannot be met by the learnable parameterized functions representing neural networks. The proposed LPAM algorithm will not require these conditions. Each function in (1) could be nonconvex and nonsmooth.

Without those restrictions, the LPAM algorithm is flexible and applicable to deep learning problems in various applications. Moreover, the convergence of LPAM can still be established in the sense of sub-sequence convergence to Clarke stationary points (see detail in Section 3). But it cannot have global convergence to a critical point as the PALM or iPALM algorithm.

The convergence result of the BCD-net is based on the assumptions that the paired image refining neural networks are asymptotically nonexpansive, the data fidelity is differentiable, and the MBIR problem is not difficult to solve.

The proposed LPAM algorithm does not require these assumptions. The convergence of LPAM is assured by the convergence of the BCD algorithm for smooth optimization and the property of the limit of the gradient of the smoothed objective functions as smoothing factor tends zero. By removing the constraints on the model, the PALM can significantly broaden its range of applications.

The remainder of the paper is organized as follows. In Section 5 we construct the LPAM algorithm motivated by the PALM algorithm and the BCD algorithm, and introduce the three steps that constitute the LPAM algorithm. In Section 6, we prove the convergence of the LPAM algorithm and deduce the iteration complexity of this algorithm. In Section 7, we apply the LPAM algorithm to the joint reconstruction of T1 and T2 MRI images with significantly under-sampled data to show the promising performance of the proposed method. Furthermore, we provide convergence analysis and comparison with several existing methods. All the figures are listed at the end of the article.

Throughout the paper, we use the following notations without further notification.

1. $H(\mathbf{x}_1, \mathbf{x}_2; \theta)$ denotes the function H of $\mathbf{x}_1$ and $\mathbf{x}_2$ given a collection of parameters θ. $H_\epsilon(\mathbf{x}_1, \mathbf{x}_2; \theta)$ denotes the smooth approximation of $H(\mathbf{x}_1, \mathbf{x}_2; \theta)$ with a smoothing parameter ϵ.
2. For a differentiable function $f(\mathbf{x}_1, \mathbf{x}_2)$, we denote the gradient of f with respect to $\mathbf{x}_i$ by $\nabla_i f(\mathbf{x}_1, \mathbf{x}_2)$ $(i = 1, 2)$. The gradient f on the domain $(\mathbf{x}_1, \mathbf{x}_2) \in R^n \times R^m$ is represented by $\nabla_{1,2} f(\mathbf{x}_1, \mathbf{x}_2)$.
3. We use $\| \cdot \|$ to denote the standard L^2 norm in Euclidean space, and write $\langle \cdot, \cdot \rangle_{\mathbb{R}^n}$ for the standard inner product on $\mathbb{R}^n$.
4. $\chi := \mathbb{R}^n \times \mathbb{R}^m$.

4 Learned Proximal Alternating Minimization (LPAM) Algorithm

In this section we propose a learned proximal alternating minimization (LPAM) algorithm for solving the problem (1). Since θ is learned, we will omit it and simply write (1) as (2).

For H_1, H_2, H and Φ we assume

$(A1)$: H_1, H_2 and H are proper and locally Lipschitz on $\mathbb{R}^n$, $\mathbb{R}^m$ and $\mathbb{R}^n \times \mathbb{R}^m$, respectively.

Each of them is possibly non-convex and non-smooth.

$(A2)$: Φ is coercive, and $\min_{\mathbf{X} \in \chi} \Phi(\mathbf{X}) > -\infty$

The proposed LPAM algorithm consists of three stages. In the first stage we convert the nonconvex and nonsmooth problem (2) to a nonconvex smooth optimization problem by an appropriate smoothing procedure. The smoothing method is not unique, but we require the smooth approximation Φ_ϵ of $\Phi := H_1(\mathbf{x}_1) + H_2(\mathbf{x}_2) + H(\mathbf{x}_1, \mathbf{x}_2)$, which is defined as

$$\Phi_\epsilon(\mathbf{x}_1, \mathbf{x}_2) := H_{1,\epsilon}(\mathbf{x}_1) + H_{2,\epsilon}(\mathbf{x}_2) + H_\epsilon(\mathbf{x}_1, \mathbf{x}_2) \tag{3}$$

to satisfy following conditions:

- (**C1**:) $\nabla H_{1,\epsilon}$, $\nabla H_{2,\epsilon}$ and $\nabla_{1,2} H_\epsilon$ are Lipschitz continuous in $\mathbb{R}^n$, $\mathbb{R}^m$ and $\mathbb{R}^n \times \mathbb{R}^m$, respectively.
- (**C2**:) $\lim_{\epsilon \to 0^+, \mathbf{z}_i \to \mathbf{x}_i} H_{i,\epsilon}(\mathbf{z}_i) = H_i(\mathbf{x}_i)$, $i = 1, 2$, for any given $\mathbf{x}_i$ in the corresponding domain. Also for any $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m$, we assume that when $\epsilon \to 0^+$ and $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2) \to \mathbf{X}$, $\lim_{\epsilon \to 0^+, \mathbf{Z} \to \mathbf{X}} H_\epsilon(\mathbf{Z}) = H(\mathbf{X})$.
- (**C3**:) There is a continuous and non-negative function $\mathfrak{m} : [0, \infty) \to [0, \infty)$ such that $\mathfrak{m}(0) = 0$,

$$\Phi_\epsilon(\mathbf{x}_1, \mathbf{x}_2) + \mathfrak{m}(\epsilon) \le \Phi_\delta(\mathbf{x}_1, \mathbf{x}_2) + \mathfrak{m}(\delta) \text{ for all } (\mathbf{x}_1, \mathbf{x}_2) \in \chi, \text{ and } 0 < \epsilon \le \delta.$$

- (**C4**:) For any $\mathbf{X}^* = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m$ and any $\mathbf{X}^k \to \mathbf{X}^*$, $\lim_{\mathbf{X}^k \to \mathbf{X}^*, \epsilon \to 0^+} \nabla_{1,2} \Phi_\epsilon(\mathbf{X}^k)$ $\in \partial^c \Phi(\mathbf{X}^*)$, where $\partial^c \Phi(\mathbf{X}^*)$ stands for the Clarke subdifferential of Φ (see Definition 1).

Remark 1 The Lipschitz constants of $\nabla H_{1,\epsilon}$, $\nabla H_{2,\epsilon}$ and $\nabla_{1,2} H_\epsilon$ may tend to infinity as ϵ tends to 0. The condition **C3** is much weaker than the monotonicity of Φ_ϵ. The condition **C3** roughly says that as long as Φ_ϵ is not too much different from a monotonic function with respect to ϵ, our method still leads to convergence. This brings great convenience in application.

In the second stage we solve the smoothed nonconvex problem with a fixed smoothing factor ϵ, i.e.

$$\min_{\mathbf{x}_1, \mathbf{x}_2} \{ \Phi_\epsilon(\mathbf{x}_1, \mathbf{x}_2) := H_{1,\epsilon}(\mathbf{x}_1) + H_{2,\epsilon}(\mathbf{x}_2) + H_\epsilon(\mathbf{x}_1, \mathbf{x}_2) \}. \tag{4}$$

In light of the substantial improvement in practical performance by ResNet [20], we propose a modified PALM that incorporates the architecture of the ResNet for solving (4). With $\epsilon = \epsilon_k > 0$, PALM [8] generates a sequence of iterates $\{\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}\}$ by

$$\mathbf{z}_1^{k+1} = \mathbf{x}_1^k - \alpha_k \nabla H_{1,\epsilon_k}(\mathbf{x}_1^k),$$

$$\mathbf{u}_1^{k+1} = \arg\min_{\mathbf{u}} \frac{1}{2\alpha_k} \|\mathbf{u} - \mathbf{z}_1^{k+1}\|^2 + H_{\epsilon_k}(\mathbf{u}, \mathbf{x}_2^k),$$

and

$$\mathbf{z}_2^{k+1} = \mathbf{x}_2^k - \beta_k \nabla H_{2,\epsilon_k}(\mathbf{x}_2^k),$$

$$\mathbf{u}_2^{k+1} = \arg\min_{\mathbf{u}} \frac{1}{2\beta_k} \|\mathbf{u} - \mathbf{z}_2^{k+1}\|^2 + H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{u}),$$

where α_k and β_k are step sizes. In some situations, the proximal points $\mathbf{u}_1^{k+1}$ and $\mathbf{u}_2^{k+1}$ are not easy to compute, so we replace $H_{\epsilon_k}(\mathbf{u}, \mathbf{x}_2^k)$ by its linear approximation at $\mathbf{z}_1^{k+1}$ together

with a proximal term $\frac{1}{2p_k}\|\mathbf{u} - \mathbf{z}_1^{k+1}\|^2$. Similarly, we replace $H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{u})$ by its linear approximation at $\mathbf{z}_2^{k+1}$ together with a proximal term $\frac{1}{2q_k}\|\mathbf{u} - \mathbf{z}_2^{k+1}\|^2$. Thus, we have

$$\mathbf{u}_1^{k+1} = \arg\min_{\mathbf{u}} \frac{1}{2\alpha_k}\|\mathbf{u} - \mathbf{z}_1^{k+1}\|^2 + <\nabla_1 H_{\epsilon_k}(\mathbf{z}_1^{k+1}, \mathbf{x}_2^k), \mathbf{u} - \mathbf{z}_1^{k+1}> + \frac{1}{2p_k}\|\mathbf{u} - \mathbf{z}_1^{k+1}\|^2 \tag{5}$$

$$= \mathbf{z}_1^{k+1} - \tau_k \nabla_1 H_{\epsilon_k}(\mathbf{z}_1^{k+1}, \mathbf{x}_2^k), \qquad \text{with} \quad \tau_k = \frac{\alpha_k p_k}{\alpha_k + p_k}.$$

where we recall that $\nabla_1 H_{\epsilon_k}$ stands for the gradient for the first n variables.

$$\mathbf{u}_2^{k+1} = \arg\min_{\mathbf{u}} \frac{1}{2\beta_k}\|\mathbf{u} - \mathbf{z}_2^{k+1}\|^2 + <\nabla_2 H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{z}_2^{k+1}), \mathbf{u} - \mathbf{z}_2^{k+1}> + \frac{1}{2q_k}\|\mathbf{u} - \mathbf{z}_2^{k+1}\|^2 \tag{6}$$

$$= \mathbf{z}_2^{k+1} - \gamma_k \nabla_2 H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{z}_2^{k+1}), \qquad \text{with} \quad \gamma_k = \frac{\beta_k q_k}{\beta_k + q_k}.$$

In deep learning approach, the step sizes α_k, τ_k, β_k and γ_k could be learnable hyperparameters. The advantage for taking the scheme (5) and (6) to update u_1 and u_2 is that this scheme is consistent with the architecture of residual learning [22, 23], which only learns the correction needed for $\mathbf{z}_i^{k+1}$ ($i = 1, 2$) and avoids gradient vanishing in network training, and thus improves solution quality in practice. However, the convergence of the sequence $\{(\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1})\}$ is not guaranteed. Inspired by the proof of convergence in [8], we propose the following:

If $\mathbf{u}_1^{k+1}$ and $\mathbf{u}_2^{k+1}$ satisfy

$$\Phi_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1}) - \Phi_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k) \le -a(\|\mathbf{u}_1^{k+1} - \mathbf{x}_1^k\|^2 + \|\mathbf{u}_2^{k+1} - \mathbf{x}_2^k\|^2), \tag{7}$$

and

$$\|\nabla_{1,2}\Phi_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k)\| \le a^{-1}(\|\mathbf{u}_1^{k+1} - \mathbf{x}_1^k\| + \|\mathbf{u}_2^{k+1} - \mathbf{x}_2^k\|) \tag{8}$$

for some $a > 0$, we take $\mathbf{x}_1^{k+1} = \mathbf{u}_1^{k+1}$, $\mathbf{x}_2^{k+1} = \mathbf{u}_2^{k+1}$.

If one of (7) and (8) is violated, we compute $(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1})$ by BCD algorithm with a simple line-search strategy for better step sizes as follows: Let $\bar{\alpha}, \bar{\beta} \in (0, 1)$, we compute

$$\mathbf{v}_1^{k+1} = \arg\min_{\mathbf{v}} <\nabla_1 \Phi_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k), \mathbf{v} - \mathbf{x}_1^k> + \frac{1}{2\bar{\alpha}}\|\mathbf{v} - \mathbf{x}_1^k\|^2, \tag{9}$$

$$\mathbf{v}_2^{k+1} = \arg\min_{\mathbf{v}} <\nabla_2 \Phi_{\epsilon_k}(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k), \mathbf{v} - \mathbf{x}_2^k> + \frac{1}{2\bar{\beta}}\|\mathbf{v} - \mathbf{x}_2^k\|^2. \tag{10}$$

The first order optimality condition leads to

$$\mathbf{v}_1^{k+1} = \mathbf{x}_1^k - \bar{\alpha}(\nabla H_{1,\epsilon_k}(\mathbf{x}_1^k) + \nabla_1 H_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k)),$$
$$\mathbf{v}_2^{k+1} = \mathbf{x}_2^k - \bar{\beta}(\nabla H_{2,\epsilon_k}(\mathbf{x}_2^k) + \nabla_2 H_{\epsilon_k}(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)). \tag{11}$$

If for some $\delta \in (0, 1)$,

$$\Phi_{\epsilon}(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_{\epsilon}(\mathbf{x}_1^k, \mathbf{x}_2^k) \le -\delta(\|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|^2 + \|\mathbf{v}_2^{k+1} - \mathbf{x}_2^k\|^2), \tag{12}$$

we set

$$\mathbf{x}_1^{k+1} = \mathbf{v}_1^{k+1}, \quad \mathbf{x}_2^{k+1} = \mathbf{v}_2^{k+1},$$

otherwise we reduce step sizes $(\bar{\alpha}, \bar{\beta})$ by $\rho(\bar{\alpha}, \bar{\beta}) \to (\bar{\alpha}, \bar{\beta})$ where $0 < \rho < 1$, and recompute $\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}$ until the condition (12) holds. Then take $\mathbf{x}_1^{k+1} = \mathbf{v}_1^{k+1}, \mathbf{x}_2^{k+1} = \mathbf{v}_2^{k+1}$.

In the third stage, we check if the ℓ_2-norm of the gradient of the smoothed objective function has been reduced enough, so that we can perform the second stage with a reduced smoothing factor. By gradually reducing the smoothing factor, we obtain a subsequence of the iterates that converges to a Clarke stationary point of the original nonconvex and nonsmooth problem.

The scheme of LPMA is given below (For notational simplicity, we omit all learned network parameters).

Algorithm 1 Learned Proximal Alternating Minimization (LPAM) Algorithm

---

1: **Input:** Initial $\mathbf{x}_1^0, \mathbf{x}_2^0, 0 < \rho, \gamma, \delta < 1$, and $\bar{\alpha}, \bar{\beta}, \sigma, a, \epsilon_0 > 0$. Maximum iteration K or tolerance $\varepsilon_{tol} > 0$.

2: **for** $k = 0, 1, 2, \ldots, K$ **do**
3:     $\mathbf{z}_1^{k+1} = \mathbf{x}_1^k - \alpha_k \nabla H_{1,\epsilon_k}(\mathbf{x}_1^k),$
4:     $\mathbf{u}_1^{k+1} = \mathbf{z}_1^{k+1} - \tau_k \nabla_1 H_{\epsilon_k}(\mathbf{z}_1^{k+1}, \mathbf{x}_2^k),$
5:     $\mathbf{z}_2^{k+1} = \mathbf{x}_2^k - \beta_k \nabla H_{2,\epsilon_k}(\mathbf{x}_2^k)$
6:     $\mathbf{u}_2^{k+1} = \mathbf{z}_2^{k+1} - \gamma_k \nabla_2 H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{z}_2^{k+1},)$
7:     **if** conditions (7) and (8) hold, **then**
8:         set $\mathbf{x}_1^{k+1} = \mathbf{u}_1^{k+1}, \mathbf{x}_2^{k+1} = \mathbf{u}_2^{k+1},$
9:     **else**
10:        For given $\bar{\alpha}, \bar{\beta} \in (0, 1)$
11:        $\mathbf{v}_1^{k+1} = \mathbf{x}_1^k - \bar{\alpha}(\nabla H_{1,\epsilon_k}(\mathbf{x}_1^k) + \nabla_1 H_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k)),$
12:        $\mathbf{v}_2^{k+1} = \mathbf{x}_2^k - \bar{\beta}(\nabla H_{2,\epsilon_k}(\mathbf{x}_2^k) + \nabla_2 H_{\epsilon_k}(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)).$
13:        **if** condition (12) holds, **then**
14:            set $\mathbf{x}_1^{k+1} = \mathbf{v}_1^{k+1}, \mathbf{x}_2^{k+1} = \mathbf{v}_2^{k+1},$
15:        **else**
16:            update $(\bar{\alpha}, \bar{\beta}) \leftarrow \rho(\bar{\alpha}, \bar{\beta})$, then **go to** 11,
17:        **end if**
18:    **end if**
19:    **if** $\|\nabla_{1,2}\Phi_{\varepsilon_k}(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})\| < \sigma\gamma\varepsilon_k$, set $\varepsilon_{k+1} = \gamma\varepsilon_k$; **otherwise**, set $\varepsilon_{k+1} = \varepsilon_k$.
20:    **if** $\sigma\varepsilon_k < \varepsilon_{tol}$, terminate.
21: **end for**
22: **Output:** $\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}$

We would like to comment here for the LPAM algorithm. Line 2 to line 18 can be viewed as the steps of the inner iteration, which, for a fixed ε, generates a sequence $\{\mathbf{x}_1^k, \mathbf{x}_2^k\}$ such that $\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\| \to 0$ as $k \to \infty$ (see the first statement of Lemma 1 in the next section). This guarantees the reduction of $\|\nabla_{1,2}\Phi_\varepsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|$ for a fixed ε. When this value drops below a prefixed value, line 19 decreases ε, and then we repeat the procedures from line 2 to line 18 to generates a new sequence $\{\mathbf{x}_1^k, \mathbf{x}_2^k\}$ corresponding to the reduced ε. We continue these steps until the termination condition in the line 20 is met. The sequence formed by the iterates, in which the reduction criterion for ε is met, has at least one accumulation point and each accumulation point is a Clarke Stationary Point of the original problem. This will be proved in Theorem 1 in Section 6.

Remark: In order to make the LPMA algorithm match the ResNet architecture, if the function $H_\epsilon(\mathbf{x}_1, \mathbf{x}_2)$ is learnable and $H_{1,\epsilon}(\mathbf{x}_1)$ and $H_{2,\epsilon}(\mathbf{x}_2)$ are given, it is better to use the scheme above. If the functions $H_{1,\epsilon}(\mathbf{x}_1)$ and $H_{2,\epsilon}(\mathbf{x}_2)$ are learnable and $H_\epsilon(\mathbf{x}_1, \mathbf{x}_2)$ is given, it is better to change the steps 3-6 to the following:

$$\mathbf{z}_1^{k+1} = \mathbf{x}_1^k - \alpha_k \nabla_1 H_{\epsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k), \tag{13}$$

$$\mathbf{u}_1^{k+1} = \mathbf{z}_1^{k+1} - \tau_k \nabla H_{1,\epsilon_k}(\mathbf{z}_1^{k+1}); \tag{14}$$

$$\mathbf{z}_2^{k+1} = \mathbf{x}_2^k - \beta_k \nabla_2 H_{\epsilon_k}(\mathbf{u}_1^{k+1}, \mathbf{x}_2^k), \tag{15}$$

$$\mathbf{u}_2^{k+1} = \mathbf{z}_2^{k+1} - \gamma_k \nabla H_{2,\epsilon_k}(\mathbf{z}_2^{k+1}); \tag{16}$$

and keep the other steps in Algorithm 1 unchanged.

5 Convergence Analysis

Since we deal with a nonconvex and nonsmooth optimization problem, the Clarke subdifferential, which is based on the concept of generalized directional derivatives, is employed here to characterize the optimality of the solutions to (2).

Definition 1 (Clarke subdifferential). Suppose that $f : \mathbb{R}^n \times \mathbb{R}^m \to (-\infty, \infty]$ is locally Lipschitz. The Clarke subdifferential of f at $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$ is defined as

$$\partial^c f(\mathbf{X}) := \{\mathbf{W} \in \mathbb{R}^n \times \mathbb{R}^m | <\mathbf{W}, \mathbf{V}> \le \limsup_{\mathbf{Z} \to \mathbf{X}, t\downarrow 0} \frac{f(\mathbf{Z} + t\mathbf{V}) - f(\mathbf{Z})}{t}, \quad \forall \mathbf{V} \in \mathbb{R}^n \times \mathbb{R}^m\},$$

where $\mathbf{X}, \mathbf{Z} \in \mathbb{R}^n \times \mathbb{R}^m$, $<\mathbf{W}, \mathbf{V}> = <\mathbf{w}_1, \mathbf{v}_1>_{\mathbb{R}^n} + <\mathbf{w}_2, \mathbf{v}_2>_{\mathbb{R}^m}$ for $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2)$, $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2)$.

Definition 2 (Clarke stationary point) For a locally Lipschitz function f defined as in Definition 1, a point $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^n \times \mathbb{R}^m$ is called a Clarke Stationary Point of f, if $0 \in \partial^c f(\mathbf{X})$.

Lemma 1 *Assume that* $\Phi_\epsilon(\mathbf{x}_1, \mathbf{x}_2)$ *is a smooth approximation of* $\Phi(\mathbf{x}_1, \mathbf{x}_2)$ *defined in* (3) *satisfying the conditions* (**C1**)-(**C4**). *Let* $\varepsilon, \eta, a > 0$, $\rho, \bar{\alpha}, \bar{\beta} \in (0, 1)$, *and* $\mathbf{X}^0 = (\mathbf{x}_1^0, \mathbf{x}_2^0)$ *be an arbitrary initial condition. Suppose* $\{\mathbf{X}^k = (\mathbf{x}_1^k, \mathbf{x}_2^k)\}$ *is the sequence generated by repeating Lines 2–18 in Algorithm 1 with* $\varepsilon_k = \varepsilon$. *Then we have*

1. $\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\| \to 0$ *as* $k \to \infty$.
2. *Let* $(\bar{\alpha}, \bar{\beta})$ $(\bar{\alpha}, \bar{\beta} \in (0, 1))$ *be the initial step size for* $(\mathbf{v}_1^k, \mathbf{v}_2^k)$. *The maximum number of required line search steps* ℓ_{max} *is*

$$\ell_{max} = \left\lceil \frac{\log\left((\frac{L_\epsilon}{2} + \delta)\max\{\bar{\alpha}, \bar{\beta}\}\right)}{\log 1/\rho} \right\rceil + 1,$$

where $\lceil A \rceil$ *represents the largest integer less than or equal to A, and* L_ϵ *is the sum of the Lipschitz constants for* $\nabla_1 H_{1,\epsilon}$, $\nabla_2 H_{2,\epsilon}$, *and* $\nabla_{1,2} H_\epsilon$, *i.e.*

$$L_\epsilon := L_{\nabla_1 H_{1,\epsilon}} + L_{\nabla_2 H_{2,\epsilon}} + L_{\nabla_{1,2} H_\epsilon}.$$

3. *For any* $\eta > 0$,

$$\min\{k \in \mathbb{N}; \ \|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})\| < \eta\} \le (2a^{-3} + \frac{4\max(\bar{\alpha}, \bar{\beta})^2}{\delta \min(\bar{\alpha}, \bar{\beta})^2 \rho^2} L_\epsilon^2) \frac{\Phi(\mathbf{X}^0) - \Phi_* + 1}{\eta^2}. \tag{17}$$

Proof of Lemma 1 Given $(\mathbf{x}_1^k, \mathbf{x}_2^k)$, in the case that $(\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1})$ generated by Algorithm 1 satisfies the conditions (7)-(8), we take $(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) = (\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1})$. From (7)-(8), it is easy to get

$$\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \le \frac{2(\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) - \Phi_\epsilon(\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1}))}{a^3}. \tag{18}$$

If $(\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1})$ fails to satisfy the condition (7) or (8), the algorithm computes $(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1})$ by (11) with a line search strategy. Let ℓ_k be the number of the required line search steps to meet the condition (12), and then we take $(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) = (\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1})$. This gives the following

$$\mathbf{v}_1^{k+1} = \mathbf{x}_1^k - \bar{\alpha}\rho^{\ell_k}\nabla_1\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k), \qquad \mathbf{v}_2^{k+1} = \mathbf{x}_2^k - \bar{\beta}\rho^{\ell_k}\nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k). \tag{19}$$

Since $\nabla_{1,2}\Phi_\epsilon$ is L_ϵ-Lipschitz continuous, we have

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) \le \Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k) + \nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k) \cdot (\mathbf{v}_2^{k+1} - \mathbf{x}_2^k) + \frac{L_\epsilon}{2}\|\mathbf{v}_2^{k+1} - \mathbf{x}_2^k\|^2$$

$$\le \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) + \nabla_1\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \cdot (\mathbf{v}_1^{k+1} - \mathbf{x}_1^k) + \frac{L_\epsilon}{2}\|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|^2$$

$$+ \nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k) \cdot (\mathbf{v}_2^{k+1} - \mathbf{x}_2^k) + \frac{L_\epsilon}{2}\|\mathbf{v}_2^{k+1} - \mathbf{x}_2^k\|^2. \tag{20}$$

The combination of (19) and (20) yields

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) \le \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) + (-\frac{1}{\bar{\alpha}\rho^{\ell_k}} + \frac{L_\epsilon}{2})\|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|^2 + (-\frac{1}{\bar{\beta}\rho^{\ell_k}} + \frac{L_\epsilon}{2})\|\mathbf{v}_2^{k+1} - \mathbf{x}_2^k\|^2. \tag{21}$$

Hence the condition (12) is met if

$$-\frac{1}{\bar{\alpha}\rho^{\ell_k}} + \frac{L_\epsilon}{2} \le -\delta, \quad \text{and} \quad -\frac{1}{\bar{\beta}\rho^{\ell_k}} + \frac{L_\epsilon}{2} \le -\delta. \tag{22}$$

Hence, for any $k = 1, 2, \ldots$ the maximum line search steps ℓ_{max} required for (12) satisfies

$$\rho^{\ell_{max}} = (\delta + L_\varepsilon/2)^{-1}(\max\{\bar{\alpha}, \bar{\beta}\})^{-1},$$

so we abuse the notation ℓ_{max} by defining it as

$$\ell_{max} = \left\lceil \frac{\log\left((\frac{L_\epsilon}{2} + \delta)\max\{\bar{\alpha}, \bar{\beta}\}\right)}{\log 1/\rho} \right\rceil + 1. \tag{23}$$

This proves the second statement of this lemma. Moreover, the fact that $\ell_k \le \ell_{max}$ for any $k = 0, 1, 2, \ldots$, implies

$$\rho^{\ell_k} \ge \rho^{\ell_{max}} = \frac{\rho}{(\delta + L_\varepsilon/2)\max\{\bar{\alpha}, \bar{\beta}\}}. \tag{24}$$

Next we prove the other two statements. Note that from (19) the condition (12) can be rewritten as:

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) \le \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) - \delta\bar{\alpha}^2\rho^{2\ell_k}\|\nabla_1\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 - \delta\bar{\beta}^2\rho^{2\ell_k}\|\nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)\|^2. \tag{25}$$

Now we estimate the last term in (25) in terms of $\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)$. First by the Lipschitz continuity of $\nabla_2\Phi_\epsilon$ we have

$$\|\nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k) - \nabla_2\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\| \le L_\epsilon\|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|.$$

Clearly this implies

$$\|\nabla_2\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)\| \ge \|\nabla_2\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\| - L_\epsilon\|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|. \tag{26}$$

Here if the right hand side is positive, we will choose $\sigma_1 \in (0, 1)$ to be determined, and take advantage of the following inequality: $(a - b)^2 \geq \frac{\sigma_1}{2} a^2 - \sigma_1 b^2$ together with (19) to obtain

$$
\|\nabla_2 \Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)\|^2
$$
$$
\geq \frac{\sigma_1}{2} \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 - \sigma_1 L_\epsilon^2 \|\mathbf{v}_1^{k+1} - \mathbf{x}_1^k\|^2
$$
$$
= \frac{\sigma_1}{2} \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 - \sigma_1 L_\epsilon^2 \bar{\alpha}^2 \rho^{2\ell_k} \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2. \tag{27}
$$

Inserting (27) into (25), we have

$$
\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \tag{28}
$$
$$
\leq - \left(\delta \bar{\alpha}^2 \rho^{2\ell_k} - \delta \bar{\beta}^2 \rho^{2\ell_k} L_\epsilon^2 \sigma_1 \bar{\alpha}^2 \rho^{2\ell_k}\right) \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 - \delta \bar{\beta}^2 \rho^{2\ell_k} \frac{\sigma_1}{2} \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2
$$
$$
= - \delta \bar{\alpha}^2 \rho^{2\ell_k} (1 - \bar{\beta}^2 \rho^{2\ell_k} L_\epsilon^2 \sigma_1) \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 - \delta \bar{\beta}^2 \rho^{2\ell_k} \frac{\sigma_1}{2} \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2.
$$

If $\bar{\beta}^2 \rho^{2\ell_k} L_\epsilon^2 < \frac{2}{3}$, we just choose $\sigma_1 = \frac{1}{2}$. Otherwise we set σ_1 as $\sigma_1 = 1/(2\bar{\beta}^2 \rho^{2\ell_k} L_\epsilon^2)$. Since in this case, $\bar{\beta}^2 \rho^{2\ell_k} L_\epsilon^2 \geq \frac{2}{3}, \sigma_1 \in (0, 1)$.

In either case the coefficient of $\|\nabla_1 \phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$ is less than $-\frac{\delta}{2} \bar{\alpha}^2 \rho^{2\ell_k}$. For the coefficient of $\|\nabla_2 \phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$, in the first case it is less than $-\frac{\delta}{4} \bar{\beta}^2 \rho^{2\ell_k}$. In the second case it is less than $-\frac{\delta}{4L_\epsilon^2}$. Then it is easy to verify that in both cases, by (24) both coefficients are less than $-\frac{\delta \min(\bar{\alpha}, \bar{\beta})^2 \rho^2}{4 \max(\bar{\alpha}, \bar{\beta})^2 L_\epsilon^2}$ Thus we have

$$
\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \leq -D_\epsilon \|\nabla_{1,2} \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2, \tag{29}
$$

where

$$
D_\epsilon = \frac{\delta \min(\bar{\alpha}, \bar{\beta})^2 \rho^2}{4 \max(\bar{\alpha}, \bar{\beta})^2 L_\epsilon^2}. \tag{30}
$$

Consequently,

$$
\|\nabla_{1,2} \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \leq \frac{4 \max(\bar{\alpha}, \bar{\beta})^2 L_\epsilon^2}{\delta \min(\bar{\alpha}, \bar{\beta})^2 \rho^2} \{\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) - \Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1})\}. \tag{31}
$$

If the right hand side of (26) is negative, we obtain from (19) that

$$
\|\nabla_2 \Phi_\epsilon(x_1^k, x_2^k)\| \leq L_\epsilon \|v_1^{k+1} - x_1^k\| = L_\epsilon \bar{\alpha} \rho^{l_k} \|\nabla_1 \Phi_\epsilon(x_1^k, x_2^k)\|.
$$

Consequently,

$$
\|\nabla_{1,2} \Phi_\epsilon(x_1^k, x_2^k)\|^2 = \|\nabla_1 \Phi_\epsilon(x_1^k, x_2^k)\|^2 + \|\nabla_2 \Phi_\epsilon(x_1^k, x_2^k)\|^2
$$
$$
\leq (1 + L_\epsilon^2 \bar{\alpha}^2 \rho^{2l_k}) \|\nabla_1 \Phi_\epsilon(x_1^k, x_2^k)\|^2. \tag{32}
$$

Then we use (32) to estimate (25) differently:

$$
\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \tag{33}
$$
$$
\leq - \delta \bar{\alpha}^2 \rho^{2\ell_k} \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \leq -\frac{\delta \bar{\alpha}^2 \rho^{2l_k}}{1 + L_\epsilon^2 \bar{\alpha}^2 \rho^{2l_k}} \|\nabla_{1,2} \Phi_\epsilon(x_1^k, x_2^k)\|^2.
$$

If $L_\epsilon^2 \bar{\alpha}^2 \rho^{2l_k} \geq 1$, we have

$$
\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \leq -\frac{\delta}{2L_\epsilon^2} \|\nabla_{1,2} \Phi_\epsilon(x_1^k, x_2^k)\|^2. \tag{34}
$$

Otherwise,

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \le -\frac{\delta}{2}\bar{\alpha}^2 \rho^{2l_k} \|\nabla_{1,2}\Phi_\epsilon(x_1^k, x_2^k)\|^2. \tag{35}$$

In either case we see that (29) still holds. Note that we used (24) for a lower bound of ρ^{2l_k}. Thus we have established (31).

Observing (7), (18), (12) and (31) we see that in either case of $(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) = (\mathbf{u}_1^{k+1}, \mathbf{u}_2^{k+1})$ or $(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) = (\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1})$, there are two constants

$$b_1 = \min\{a, \delta\} \quad \text{and} \quad b_2 = \max\{2a^{-3}, \frac{4\max(\bar{\alpha}, \bar{\beta})^2 L_\epsilon^2}{\delta\min(\bar{\alpha}, \bar{\beta})^2 \rho^2}\}, \tag{36}$$

such that

$$\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) \le -b_1(\|\mathbf{x}_1^{k+1} - \mathbf{x}_1^k\|^2 + \|\mathbf{x}_2^{k+1} - \mathbf{x}_2^k\|^2), \tag{37}$$

and

$$\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \le b_2(\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) - \Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})). \tag{38}$$

From (37) and **C2** we get

$$\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}) < \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) < \ldots < \Phi_\epsilon(\mathbf{x}_1^0, \mathbf{x}_2^0) \le \Phi(\mathbf{x}_1^0, \mathbf{x}_2^0) + 1. \tag{39}$$

Furthermore, for any integer $K > 0$ summing up (38) for $k = 0, \ldots, K$, we have

$$\sum_{k=0}^K \|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \le b_2(\Phi_\epsilon(\mathbf{x}_1^0, \mathbf{x}_2^0) - \Phi_\epsilon(\mathbf{x}_1^{K+1}, \mathbf{x}_2^{K+1})) \le b_2(\Phi_\epsilon(\mathbf{x}_1^0, \mathbf{x}_2^0) - \Phi_* + 1).$$
$$\tag{40}$$

We use Φ_* to denote a uniform lower bound for all Φ_ϵ: $\min_{\mathbf{x}} \Phi_\epsilon(\mathbf{x}) \ge \Phi_*$ for all $\epsilon > 0$. Since b_2 is independent of K, we see that for any fixed $\epsilon > 0$, $\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \to 0$ as $k \to \infty$. This proves the first statement of Lemma 1. For the third statement of Lemma 1, we set $\kappa := \min\{k \in \mathbb{N}; \|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})\| \le \eta\}$, then we know that $\|\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})\| \ge \eta$ for all $k \le \kappa - 1$. Thus from (40) we have

$$\kappa\eta^2 \le \sum_{k=0}^{\kappa-1} \|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})\|^2 = \sum_{k=1}^{\kappa} \|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 \le b_2(\Phi_\epsilon(\mathbf{x}_1^0, \mathbf{x}_2^0) - \Phi_* + 1).$$

The third statement follows immediately. Lemma 1 is established.   □

From the first statement of Lemma 1 the reduction criterion for ϵ_k in Line 19 of Algorithm 1 must be satisfied within finitely many iterations for any $k = 1, 2, \ldots$. Hence ϵ_k will eventually be small enough to terminate the algorithm. Let k_l be the counter of iteration when the criterion in Line 19 of Algorithm 1 is met for the l-th time (we set $k_0 = -1$), then we can partition the iteration counters $k = 0, 1, 2, \ldots$, into segments accordingly, so that in the l-th segment $k = k_l + 1, \ldots, k_{l+1}$ and $\epsilon_k = \epsilon_{k_l+1} = \epsilon_0\gamma^l$. In the next theorem we will give the length of each segment, which will lead to the iteration complexity of Algorithm 1 for any $\epsilon_{\text{tol}} > 0$.

Theorem 1 *Let Φ_ϵ and Φ be described as in Lemma 1 and $(\mathbf{C1}) - (\mathbf{C4})$ hold for Φ_ϵ. If $\epsilon_{tol} = 0$, suppose $\{\mathbf{X}^k = (\mathbf{x}_1^k, \mathbf{x}_2^k)\}$ is the sequence generated by Algorithm 1 with arbitrary initial condition $\mathbf{X}^0 = (\mathbf{x}_1^0, \mathbf{x}_2^0)$. Let $\{\tilde{\mathbf{X}}^l\} = \{(\tilde{\mathbf{x}}_1^l, \tilde{\mathbf{x}}_2^l) =: (\mathbf{x}_1^{k_l+1}, \mathbf{x}_2^{k_l+1})\}$ be the subsequence of $\{\mathbf{X}^k\}$ where the reduction criterion for ϵ in the algorithm is met for $k = k_l$ and $l = 1, 2\ldots$.*

Then $\{\tilde{\mathbf{X}}^l\}$ has at least one accumulation point and each accumulation point is a Clarke Stationary Point of the original problem. Moreover, the number of iterations, $k_{l+1} - k_l$, for the $l-th$ segment is bounded by

$$k_{l+1} - k_l \le (2a^{-3} + \frac{4\min\{\bar\alpha, \bar\beta\}^2}{\delta\max\{\bar\alpha, \bar\beta\}^2} L^2_{\epsilon_0\gamma^l}) \frac{\Phi(\mathbf{X}^0) - \Phi_* + 1}{\sigma^2\epsilon_0^2\gamma^{2l+2}}. \tag{41}$$

For any $\epsilon_{tol} > 0$, the total number of iterations for Algorithm 1 to terminate with ϵ_{tol} is bounded by

$$\sum_{l=1}^{l_0-1}(2a^{-3} + \frac{4\min\{\bar\alpha, \bar\beta\}^2}{\delta\max\{\bar\alpha, \bar\beta\}^2} L^2_{\epsilon_0\gamma^l}) \frac{\Phi(\mathbf{X}^0) - \Phi_* + 1}{\sigma^2\epsilon_0^2\gamma^{2l+2}} = O(L^2_{\epsilon_{tol}}\epsilon_{tol}^{-2}), \tag{42}$$

where l_0 is number of reductions and we have $l_0 - 1 = \frac{\log \sigma\epsilon_0/\epsilon_{tol}}{\log 1/\gamma}$, $C_1 > 0$ depends only on ρ and δ.

Remark 2 In application we usually have $L_\epsilon \sim \epsilon^{-1}$, which makes the bound in (42) $O(\epsilon_{tol}^{-4})$.

Remark 3 The figure below shows the generation of the sequence $\{\tilde{\mathbf{X}}^l\}$ in Theorem 1. The iterates highlighted in red indicate the selected sequence $\{\tilde{\mathbf{X}}^l\}$, where the reduction criterion for ε_k is met at $k = k_l$ for $l = 0, 1, 2....$

Proof of Theorem 1 First we claim that the sequence $\tilde{\mathbf{X}}^l = (\tilde{\mathbf{x}}_1^l, \tilde{\mathbf{x}}_2^l)$ is compact. This is largely based on **C3**. In the first step, the initial $\mathbf{X}^0$ corresponds to ϵ_0. Then we have $\Phi_{\epsilon_0}(\mathbf{X}^1) < \Phi_{\epsilon_0}(\mathbf{X}^0)$. After this we may have $\mathbf{X}^2, .., \mathbf{X}^l$ and then we have $\epsilon_1 = \gamma\epsilon_0$ and $\Phi_{\epsilon_1}(\mathbf{X}^{l+1}) < \Phi_{\epsilon_1}(\mathbf{X}^l)$. Here we mention that we don't have $\Phi_{\epsilon_1}(\mathbf{X}^{l+1}) \le \Phi_{\epsilon_0}(\mathbf{X}^l)$, but by the third requirement of Φ_ϵ we have

$$\Phi_{\epsilon_1}(\mathbf{X}^{l+1}) + \mathfrak{m}(\epsilon_1) \le \Phi_{\epsilon_0}(\mathbf{X}^{l+1}) + \mathfrak{m}(\epsilon_0) < \Phi_{\epsilon_0}(\mathbf{X}^0) + \mathfrak{m}(\epsilon_0).$$

In a similar manner we then prove that $\Phi_{\epsilon_l}(\tilde{\mathbf{X}}^l) + \mathfrak{m}(\epsilon_l)$ is uniformly bounded for all l. Since $\mathfrak{m}$ is a positive continuous function and all $\epsilon_l \le \epsilon_0$, after removing the bound for all $\mathfrak{m}(\epsilon_l)$ we have the uniform bound for all $\Phi_{\epsilon_l}(\tilde{\mathbf{X}}^l)$.

Next we estimate the iteration times based on the previous lemma. If $\epsilon_{tol} \ne 0$, to estimate $k_{l+1} - k_l$, which is the iteration times for the inner circle, we see that $\epsilon = \epsilon_{k_l} = \epsilon_0\gamma^l$, then η in Lemma 1 is $\eta = \sigma\gamma\epsilon_{k+1} = \sigma\epsilon_0\gamma^{l+1}$, the initial step is $\mathbf{X}_{k_l+1}$. Then,

$$k_{l+1} - k_l \le (2a^{-3} + \frac{4\min\{\bar\alpha, \bar\beta\}^2}{\delta\max\{\bar\alpha, \bar\beta\}^2} L^2_{\epsilon_0\gamma^l}) \frac{\Phi(\mathbf{X}^0) - \Phi_* + 1}{\sigma^2\epsilon_0^2\gamma^{2l+2}},$$

and (41) is justified. Let l_0 be the number of reductions, then $\sigma\epsilon_0\gamma^{l_0-1} \ge \epsilon_{tol}$. Thus,

$$l_0 - 1 \le \frac{\log \sigma\epsilon_0/\epsilon_{tol}}{\log 1/\gamma}. \tag{43}$$

Based on this we have

$$\sum_{l=0}^{l_0-1}(k_{l+1} - k_l) \le \sum_{l=1}^{l_0-1}\left(2a^{-3} + \frac{4\min\{\bar\alpha, \bar\beta\}^2}{\delta\max\{\bar\alpha, \bar\beta\}^2} L^2_{\epsilon_0\gamma^l}\right) \frac{\Phi(\mathbf{X}^0) - \Phi_* + 1}{\sigma^2\epsilon_0^2\gamma^{2l+2}},$$

which yields the left hand side of (42). If we use $L_{\epsilon_{tol}}$ as an upper bound of $L_{\epsilon_0 \gamma^l}$ for all l, the summation above leads to $O(L^2_{\epsilon_{tol}} \gamma^{2l_0})$. Using the estimate of l_0 in (43) we see that this bound is $O(L^2_{\epsilon_{tol}} \epsilon_{tol}^{-2})$. Since $\|\nabla_{1,2} \Phi_{\epsilon_{k_l}}(\mathbf{X}^{k_l+1})\| \leq \sigma \gamma \epsilon_{k_l} = \sigma \epsilon_0 \gamma^{l+1} \to 0$ as $l \to \infty$, so if we choose a subsequence of $\{\mathbf{x}_1^{k_l+1}, \mathbf{x}_2^{k_l+1}\}$ that converges to an equilibrium point $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2)$, by **C4**, $0 \in \partial^c \phi$, then Definition 2 says $\bar{\mathbf{x}}$ is a Clarke Stationary Point of Φ. Theorem 1 is established. $\qquad \square$

5.1 Small Initial Steps

Finally in this subsection we prove that if the initial steps $\bar{\alpha}$ and $\bar{\beta}$ satisfy

$$\delta < \bar{\alpha} L_\epsilon < 1, \quad \delta < \bar{\beta} L_\epsilon < 1, \tag{44}$$

for some $\delta > 0$, we can improve the estimate of D_ϵ from (30) to $D_\epsilon \leq c L_\epsilon^{-1}$ for some universal constant $c > 0$.

Since under this assumption we have $-1 + \frac{L_\epsilon}{2}\bar{\alpha} < 0$ and $-1 + \frac{L_\epsilon}{2}\bar{\beta} < 0$, we can estimate the decrease of $\Phi_\epsilon(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1})$ in terms of $\|\nabla_{1,2}\Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$ using (19),(21) and $\ell_k = 0$:

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) \leq \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k) + (-\frac{1}{\bar{\alpha}} + \frac{L_\epsilon}{2})(\bar{\alpha})^2 \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$$
$$+ (-\frac{1}{\bar{\beta}} + \frac{L_\epsilon}{2})(\bar{\beta})^2 \|\nabla_2 \Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)\|^2. \tag{45}$$

To estimate $\|\nabla_{1,2}\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{x}_2^k)\|$ we use Lipschitz continuity and an elementary inequality to obtain (27) as before. Inserting (27) into (45) we have

$$\Phi_\epsilon(\mathbf{v}_1^{k+1}, \mathbf{v}_2^{k+1}) - \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)$$
$$\leq \left((-\frac{1}{\bar{\alpha}} + \frac{L_\epsilon}{2})(\bar{\alpha})^2 - (-\frac{1}{\bar{\beta}} + \frac{L_\epsilon}{2})(\bar{\beta})^2 \sigma L_\epsilon^2 (\bar{\alpha})^2\right) \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$$
$$+ \frac{\sigma}{2}(-\frac{1}{\bar{\beta}} + \frac{L_\epsilon}{2})(\bar{\beta})^2 \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2$$
$$= c_{1,\epsilon} \|\nabla_1 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2 + c_{2,\epsilon} \|\nabla_2 \Phi_\epsilon(\mathbf{x}_1^k, \mathbf{x}_2^k)\|^2, \tag{46}$$

where

$$c_{1,\epsilon} = (-\frac{1}{\bar{\alpha}} + \frac{L_\epsilon}{2})(\bar{\alpha})^2 - (-\frac{1}{\bar{\beta}} + \frac{L_\epsilon}{2})(\bar{\beta})^2 \sigma L_\epsilon^2 (\bar{\alpha})^2$$
$$= \bar{\alpha}\left((-1 + \frac{L_\epsilon}{2}\bar{\alpha}) - (-1 + \frac{L_\epsilon}{2}\bar{\beta})\sigma (L_\epsilon \bar{\beta})(L_\epsilon \bar{\alpha})\right) \tag{47}$$
$$c_{2,\epsilon} = \frac{\sigma}{2}(-\frac{1}{\bar{\beta}} + \frac{L_\epsilon}{2})(\bar{\beta})^2 = \frac{\sigma}{2}(-1 + \frac{\bar{\beta} L_\epsilon}{2})(\bar{\beta}).$$

we choose σ to be

$$\sigma = \frac{1 - \frac{L_\epsilon}{2}\bar{\alpha}}{2(L_\epsilon \bar{\beta})(L_\epsilon \bar{\alpha})(1 - \frac{L_\epsilon}{2}\bar{\beta})}.$$

Then,

$$c_{1,\epsilon} = \frac{\bar{\alpha}}{2}(-1 + \frac{L_\epsilon \bar{\alpha}}{2}), \quad c_{2,\epsilon} = \frac{-1 + L_\epsilon/2\bar{\alpha}}{4L_\epsilon(L_\epsilon \bar{\alpha})}.$$

Observing (44), we have

$$|c_{1,\epsilon}| + |c_{2,\epsilon}| \leq CL_\epsilon$$

for some $c > 0$ independent of ϵ. This clearly implies that D_ϵ is bounded by CL_ϵ^{-1} for some universal $C > 0$.

6 Experiments

In this section, we examine the performance of the proposed method for joint MRI T1 and T2 image reconstruction using significantly under-sampled data. We outline the proposed variational model, explore the convergence property of the LPAM algorithm used to solve the model, and demonstrate the efficiency of the LPAM-net.

The content is organized into five subsections. The first subsection introduces the variational model, especially the learned joint feature extractor. Then in the second subsection we verify the assumptions for the convergence of the algorithm. Then in subsection 3, we describe the dataset used in the experiments and the metrics used to evaluate the reconstruction results. Subsection 4 elucidates the complete reconstruction procedure including the initialization network and the LPAM-net. Finally, in the last subsection, we demonstrate the numerical results of the experiments via comparisons with several existing methods.

6.1 Proposed Model and the Feature Extractor

Given the under-sampled k-space data $\mathbf{f}_1, \mathbf{f}_2$ of the modalities T1 and T2, our goal is to jointly reconstruct the corresponding images ($\mathbf{x}_1$ and $\mathbf{x}_2$). We formulate the T1-T2 joint reconstruction as the following nonconvex and nonsmooth minimization problem:

$$\min_{\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2)^T} \frac{1}{2}\|\mathbf{P}F\mathbf{x}_1 - \mathbf{f}_1\|^2 + \frac{1}{2}\|\mathbf{P}F\mathbf{x}_2 - \mathbf{f}_2\|^2 + \|g_\theta(\mathbf{x}_1, \mathbf{x}_2)\|_{2,1}, \qquad (48)$$

where $\mathbf{x}_1 \in \mathbb{R}^n$, $\mathbf{x}_2 \in \mathbb{R}^n$, $\mathbf{f}_1 \in \mathbb{C}^n$ and $\mathbf{f}_2 \in \mathbb{C}^n$. $\mathbf{f}_1$ and $\mathbf{f}_2$ are the k-space data, and $\mathbf{g}$ is a learned joint feature extractor mapping from $\mathbb{C}^{n\times 2}$ to $\mathbb{C}^{n\times d}$, where n is the spatial dimension and d is the channel number of the output feature tensor. Here, F stands for the discrete Fourier transform and $\mathbf{P}$ is the binary matrix representing the k-space mask when acquiring data for $\mathbf{x}_1$ and $\mathbf{x}_2$. ($\mathbf{x}_1, \mathbf{x}_2$) means the image $\mathbf{x}_1$ and the image $\mathbf{x}_2$ are concatenated. In this model, the first two terms are data fidelity terms and the last term is the regularization term. The regularization term $\mathbf{g}_\theta$ is a learnable common feature extractor for $\mathbf{x}_1$ and $\mathbf{x}_2$ and it enhances the sparsity of the common features.

More specifically, let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)^T$ denote the vector of the concatenated two images of dimension $2n$. Then, the regularization term is represented by $\|\mathbf{g}_\theta(\mathbf{X})\|_{2,1}$. The $\mathbf{g}_{\theta,i}(\mathbf{X}) \in \mathbb{C}^d$ is the i-feature vector of $\mathbf{X}$ for $i = 1, ..., n$. Here, n represents the pixel number of the image. $\nabla_{1,2}\mathbf{g}_\theta(\mathbf{X})$ represents the gradient of $\mathbf{g}_\theta$ as a function of $2n$ variables. Then,

$$\|\mathbf{g}_\theta(\mathbf{X})\|_{2,1} = \sum_{i=1}^n \|\mathbf{g}_{\theta,i}(\mathbf{X})\|,$$

where $\mathbf{g}$ is a vanilla l-layer CNN with component-wise activation function σ:

$$\mathbf{g}_\theta(\mathbf{X}) = \mathbf{w}_l(\sigma...\sigma(\mathbf{w}_3 * \sigma(\mathbf{w}_2 * \sigma * (\mathbf{w}_1 * (\mathbf{X}))))). \qquad (49)$$

Here σ is a smoothed ReLu function defined as:

$$\sigma(x) = \begin{cases} 0, & \text{if } x \leq -\delta \\ \frac{1}{4\delta}x^2 + \frac{1}{2}x + \frac{\delta}{4}, & \text{if } -\delta < x < \delta \\ x, & \text{if } x \geq \delta. \end{cases} \tag{50}$$

We denote $\mathbf{g}_{\theta,i}(\mathbf{X})$ as $\mathbf{g}_i(\mathbf{X})$ for simplicity in the following text and in the convergence analysis.

Let $r_\epsilon(\mathbf{X})$ be the smoothed approximation of the regularization term with the parameter ϵ as follows:

$$r_\epsilon(\mathbf{X}) = \sum_{i \in I_0} \frac{1}{2\epsilon}\|\mathbf{g}_i(\mathbf{X})\|^2 + \sum_{i \in I_1}(\|\mathbf{g}_i(\mathbf{X})\| - \frac{\epsilon}{2}),$$

where $I_0 = \{i \in [n] \mid \|\mathbf{g}_i(\mathbf{X})\| \leq \epsilon\}$, $I_1 = [n] \setminus I_0$. Then, the gradient of the smoothed term is:

$$\nabla_{1,2}r_\epsilon(\mathbf{X}) = \sum_{i \in I_0} \nabla_{1,2}\mathbf{g}_i(\mathbf{X})^\top \frac{\mathbf{g}_i(\mathbf{X})}{\epsilon} + \sum_{i \in I_1} \nabla_{1,2}\mathbf{g}_i(\mathbf{X})^\top \frac{\mathbf{g}_i(\mathbf{X})}{\|\mathbf{g}_i(\mathbf{X})\|}. \tag{51}$$

Now the smoothed objective function $\Phi_\epsilon(\mathbf{X})$ of $\Phi(\mathbf{X})$ is written as

$$\Phi_\epsilon(\mathbf{X}) = \frac{1}{2}\|\mathbf{P}F\mathbf{x}_1 - \mathbf{f}_1\|^2 + \frac{1}{2}\|\mathbf{P}F\mathbf{x}_2 - \mathbf{f}_2\|^2 + r_\epsilon(\mathbf{X}).$$

6.2 Convergence Analysis

6.2.1 Theoretical Proof

In this subsection, we verify that the proposed variational model (48) and its smoothing approximation satisfy the assumptions required in Lemma 1 and Theorem 1. This verification provides a strong theoretical support for the convergence of T1-T2 joint reconstruction.

Step one: Verification of C1 and C2:

From the definition of r_ϵ, it is easy to see that both **C1** and **C2** hold.

Step two: Verification of C3:

In order to prove **C3**, We choose the continuous function $\mathfrak{m}(\epsilon) = \frac{n}{2}\epsilon$ and $\mathfrak{m}(\delta) = \frac{n}{2}\delta$. Then, we need to prove the following inequality:

$$\Phi_\epsilon(\mathbf{X}) + \frac{n}{2}\epsilon \leq \Phi_\delta(\mathbf{X}) + \frac{n}{2}\delta \text{ for all } \mathbf{X} \in \chi, \text{ and } 0 < \epsilon \leq \delta. \tag{52}$$

Note that the data fidelity term is independent of smoothing parameter, hence we only need to prove the following inequality:

$$r_{\epsilon,i}(\mathbf{X}) + \frac{\epsilon}{2} \leq r_{\delta,i}(\mathbf{X}) + \frac{\delta}{2} \text{ , for all } \mathbf{X} \in \chi, \text{ and } 0 < \epsilon < \delta. \tag{53}$$

Recall that $r_{\epsilon,i}(\mathbf{X})$ is defined as:

$$r_{\epsilon,i}(\mathbf{X}) := \begin{cases} \frac{1}{2\epsilon}\|\mathbf{g}_i(\mathbf{X})\|^2, & \text{if } \|\mathbf{g}_i(\mathbf{X})\| \leq \epsilon, \\ \|\mathbf{g}_i(\mathbf{X})\| - \frac{\epsilon}{2}, & \text{if } \|\mathbf{g}_i(\mathbf{X})\| > \epsilon. \end{cases} \tag{54}$$

The proof of (53) consists of three steps. In the first case, if $\|\mathbf{g}_i(\mathbf{X})\| \geq \delta$, it is easy to use the definition of $r_{\epsilon,i}(\mathbf{X})$ to see that both sides of (53) are equal to $\|\mathbf{g}_i(\mathbf{X})\|$. In the second case, we suppose $\epsilon < \|\mathbf{g}_i(\mathbf{X})\| < \delta$. Then

$$r_{\epsilon,i}(\mathbf{X}) + \frac{\epsilon}{2} = \|\mathbf{g}_i(\mathbf{X})\| = \frac{\|\mathbf{g}_i(\mathbf{X})\|^2}{2\|\mathbf{g}_i(\mathbf{X})\|} + \frac{\|\mathbf{g}_i(\mathbf{X})\|}{2} \leq \frac{\|\mathbf{g}_i(\mathbf{X})\|^2}{2\delta} + \frac{\delta}{2} = r_{\delta,i}(\mathbf{X}) + \frac{\delta}{2}.$$

Note that the last inequality holds due to the simple fact that the function

$$h(t) := \frac{\|\mathbf{g}_i(\mathbf{X})\|}{2t} + \frac{t}{2},$$

is nondecreasing for $t \geq \|\mathbf{g}_i(\mathbf{X})\|$.

In case three: If $\delta > \epsilon \geq \|\mathbf{g}_i(\mathbf{X})\|$,

$$r_{\epsilon,i}(\mathbf{X}) + \frac{\epsilon}{2} = \frac{1}{2\epsilon}\|\mathbf{g}_i(\mathbf{X})\|^2 + \frac{\epsilon}{2} \leq \frac{1}{2\delta}\|\mathbf{g}_i(\mathbf{X})\|^2 + \frac{\delta}{2} = r_{\delta,i}(\mathbf{X}) + \frac{\delta}{2},$$

where the inequality also holds due to the fact that the function $h(t)$ as defined above is a nondecreasing function for $\epsilon \geq \|\mathbf{g}_i(\mathbf{X})\|$. In each case, **C3** is satisfied.

Step three: Verification of C4:

Note that $\mathbf{X}_{k_l+1}$ satisfies $\|\Phi_{\epsilon_{k_l}}(\mathbf{X}_{k_l+1})\| \to 0$ as $l \to \infty$. For simplicity, we use $\{\mathbf{X}_{j+1}\}$ instead of $\mathbf{X}_{k_l+1}$. ϵ_j is the smoothed factor used in the iteration to obtain $\mathbf{X}_{j+1}$. Then, we have $\mathbf{X}_{j+1} \to \tilde{\mathbf{X}}$, $\epsilon_j \to 0$ and $\nabla_{1,2}\phi_{\epsilon_j}(\mathbf{X}_{j+1}) \to 0$.

Here we claim that $\nabla_{1,2}\Phi_{\epsilon_j}(\mathbf{X}_{j+1})$ converges to a Clarke point. Similar proof has been demonstrated in [10] that

$$\partial\Phi(\tilde{\mathbf{X}}) = \{\sum_{i \in I_0} \nabla_{1,2}\mathbf{g}_i(\tilde{\mathbf{X}})^\top \mathbf{w}_i + \sum_{i \in I_1} \nabla_{1,2}\mathbf{g}_i(\tilde{\mathbf{X}})^\top \frac{\mathbf{g}_i(\tilde{\mathbf{X}})}{\|\mathbf{g}_i(\tilde{\mathbf{X}})\|} + \nabla_{1,2}f(\tilde{\mathbf{X}}) \Big| \tag{55}$$

$$\mathbf{w}_i \in \mathbb{C}^d, \|\Pi(\mathbf{w}_i; \mathcal{C}(\nabla_{1,2}\mathbf{g}_i(\tilde{\mathbf{X}})))\| \leq 1, \forall i \in I_0\},$$

where $I_0 = \{i \in [n] \mid \|\mathbf{g}_i(\tilde{\mathbf{X}})\| = 0\}$ and $I_1 = [n] \setminus I_0$. $\Pi(\mathbf{w}; \mathcal{C}(A))$ is the projection of $\mathbf{w}$ onto $\mathcal{C}(A)$ which stands for the column space of A. Then for J sufficiently large

$$\epsilon_j < \frac{1}{2}\min\{\|\mathbf{g}_i(\tilde{\mathbf{X}})\|; i \in I_1\} \leq \frac{1}{2}\|\mathbf{g}_i(\tilde{\mathbf{X}})\| \leq \|\mathbf{g}_i(\mathbf{X}_{j+1})\|, \quad \forall j \geq J, \quad \forall i \in I_1.$$

We use the fact that $\min\{\|\mathbf{g}_i(\tilde{\mathbf{X}})\|; i \in I_1\} > 0$ and $\epsilon_j \to 0$ to prove the first inequality and we use $\mathbf{X}_{j+1} \to \tilde{\mathbf{X}}$ and the continuity of $\mathbf{g}_i$ for all i to prove the last inequality. Furthermore we denote

$$\mathbf{s}_{j,i} = \begin{cases} \frac{\mathbf{g}_i(\mathbf{X}_{j+1})}{\epsilon_j}, & \text{if} \quad \|\mathbf{g}_i(\mathbf{X}_{j+1})\| \leq \epsilon_j, \\[2mm] \frac{\mathbf{g}_i(\mathbf{X}_{j+1})}{\|\mathbf{g}_i(\mathbf{X}_{j+1})\|}, & \text{if} \quad \|\mathbf{g}_i(\mathbf{X}_{j+1})\| > \epsilon_j. \end{cases}$$

Then, we have

$$\nabla_{1,2}\Phi_{\epsilon_j}(\mathbf{X}_{j+1}) = \sum_{i \in I_0} \nabla_{1,2}\mathbf{g}_i(\mathbf{X}_{j+1})^\top \mathbf{s}_{j,i} + \sum_{i \in I_1} \nabla_{1,2}\mathbf{g}_i(\mathbf{X}_{j+1})^\top \frac{\mathbf{g}_i(\mathbf{X}_{j+1})}{\|\mathbf{g}_i(\mathbf{X}_{j+1})\|} + \nabla_{1,2}f(\mathbf{X}_{j+1}).$$

As $j \to \infty$, obviously $\nabla_{1,2}\Phi_{\epsilon_j}(\mathbf{X}_{j+1})$ converges to the three terms in (55). Since $\nabla_{1,2}\Phi_{\epsilon_j}(\mathbf{X}_{j+1}) \to 0$ and $\partial\Phi(\tilde{\mathbf{X}})$ is closed, $\mathbf{X}_{j+1}$ converges to a Clarke stationary point. We have validated the four conditions required for the proposed algorithm to be convergent in regards of the application of joint reconstruction of T1 and T2 MRI images. Therefore, by Lemma 1 and Theorem 1, this joint reconstruction will provide a convergent solution.

6.3 Experiment Settings

The dataset we used in all experiments are from *Multi-modal Brain Tumor Segmentation Challenge 2018*.[25] Both the training set and the validation set in Multi-modal Brain Tumor Segmentation Challenge 2018 contain four modalities (T1,T2, FLAIR and $T1_c$). The training set is scanned from 285 patients and the validation set is obtained from 66 patients. Each image has a volume size of $240 \times 240 \times 155$. Each modality has two types of gliomas: 75 volumes of low-grade gliomas(LGG) and 210 volumes of high-grade gliomas(HGG).

In our experiments, we used HGG MRI images from two modalities (T1 and T2). For each modality, we chose images scanned from 50 patients as our training set and we randomly chose images scanned from 6 patients as our testing set. We cropped the 2D image size to be 160×180 in the center region, resulting in a overall number of 500 images as our training set and 60 images as our testing set.

To obtain the k-space images, We used Matlab to apply 2D fast Fourier transform to every ground truth image in the training set and the testing set. Then, we shifted the zero-frequency components to the central area for each image and we applied 10%, 20% radial mask to every image. After having all the operations on Matlab, we had the k-space under-sampled images, denoted as $\mathbf{f}_1$ and $\mathbf{f}_2$.

The Initialization network was implemented in Python using the TensorFlow framework and the LPAM-net was implemented in Python using the PyTorch framework. Our experiments were run on a Linux server with an NVIDIA A100 Tensor Core GPU available on HiPerGator. The batch size was set to be 1 in all experiments due to the consideration of the GPU memory, data volume and the operation speed.

Our reconstruction results are assessed using four evaluation matrices: peak-to-noise ratio (PSNR), structural similarity (SSIM), normalized mean squared error (NMSE) and root mean square error (RMSE). PSNR, SSIM, NMSE and RMSE can be computed using the following equations.

Suppose we have a reconstructed image $\mathbf{X}$ and a ground truth image $\mathbf{Y}$, each with a dimension of $m \times n$. We define the mean square error and peak-to-noise ratio as

$$MSE\,(\mathbf{X}, \mathbf{Y}) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [\mathbf{X}(i, j) - \mathbf{Y}(i, j)]^2 = \frac{1}{mn} ||\mathbf{X} - \mathbf{Y}||_2^2, \tag{56}$$

and

$$PSNR\,(\mathbf{X}, \mathbf{Y}) = 10 \cdot log_{10} \left(\frac{MAX(\mathbf{Y})}{MSE(\mathbf{X}, \mathbf{Y})} \right).$$

$MAX(\mathbf{Y})$ is the maximum possible pixel value of the ground truth image. PSNR measures the quality of a compressed image in comparison to its ground truth image. A higher PSNR indicates a better quality of the compressed or reconstructed image.

$$SSIM\,(\mathbf{X}, \mathbf{Y}) = \frac{(2\mu_X \mu_Y + c_1)(2\sigma_{XY} + c_2)}{(\mu_X^2 + \mu_Y^2 + c_1)(\sigma_X^2 + \sigma_Y^2 + c_2)},$$

where μ_X, μ_Y is the pixel mean of X and Y, respectively; σ_X^2, σ_Y^2 are the variance of X, Y respectively. $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ are two variables to avoid zero denominator and L is the dynamic range of the pixel values. The SSIM, ranging from 0 to 1, is used to measure the similarity between two images. A higher SSIM value indicates a greater similarity between the two images.

$$NMSE\,(\mathbf{X}, \mathbf{Y}) = \frac{||\mathbf{X} - \mathbf{Y}||_2^2}{||\mathbf{Y}||_2^2}.$$

Table 1 The evaluation results (mean $\pm$ standard deviation) for noiseless $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$ after the initialization network

Image	PSNR	SSIM	NMSE
10% T1	23.146 ± 0.66	0.617 ± 0.034	0.0196 ± 0.0035
10% T2	23.65 ± 1.64	0.619 ± 0.04	0.063 ± 0.026
20% T1	26.517 ± 0.518	0.735 ± 0.031	0.009 ± 0.0014
20% T2	27.032 ± 1.633	0.735 ± 0.031	0.029 ± 0.012

NMSE measures the mean relative error.

$$RMSE\,(\mathbf{X}, \mathbf{Y}) = \sqrt{MSE(\mathbf{X}, \mathbf{Y})} = \sqrt{\frac{1}{mn}||\mathbf{X} - \mathbf{Y}||_2^2}.$$

RMSE is utilized to measure the difference between the reconstructed image and the ground truth.

6.4 The Architecture of the Reconstruction Process

The complete reconstruction procedure consists of the Initialization network and the LPAM-net. The architecture of the multi-phase LPAM-net follows the LPAM algorithm exactly in the way that each phase of the network corresponds to each itration of the algorithm. The details are given in the following subsections.

6.4.1 Initialization Network

To have a better input for LPAM-net, we construct an initialization network, the major component of which is the initialization blocks. It contains a 4-layer convolutional neural network(CNN) following the residual structure, as outlined in [20]. Subsequently, an inverse Fourier transform is applied to the output of the initialization block. The initialization block performs interpolation on the inputs $\mathbf{f}_1$ and $\mathbf{f}_2$, which helps to fill up the missing components of the under-sampled images in the k-space domain. We learned the weights of the kernels of the Initialization network by minimizing the following loss function $L = Loss1 + Loss2$ using Adam Optimizer implemented using Tensorflow:

$$Loss1 = \frac{1}{mn}||\mathbf{x}_1^0 - \mathbf{x}_1^{truth}||_2^2,$$

$$Loss2 = \frac{1}{mn}||\mathbf{x}_2^0 - \mathbf{x}_2^{truth}||_2^2;$$

where $m \times n$ is the dimension of the image.

A learning rate of 10^{-3} was set, complemented by a decay rate of 0.95 for every 100 steps. The Xavier initialization was utilized to initialize the kernels. Table 1 displays the quantitative assessment of the Initialization network for the 60 images in the testing set. This includes the average and the standard deviation for the outputs $\mathbf{x}_1^0$, $\mathbf{x}_2^0$ under noiseless condition, corresponding to radial masks with an under-sampling ratios of 10% and 20%.

6.4.2 LPAM-net for Joint Reconstruction

The architecture of the proposed LPAM-net is shown in the Fig. 1, which follows the LPAM algorithm exactly. The inputs for the LPAM-net are the outputs of the Initialization network,
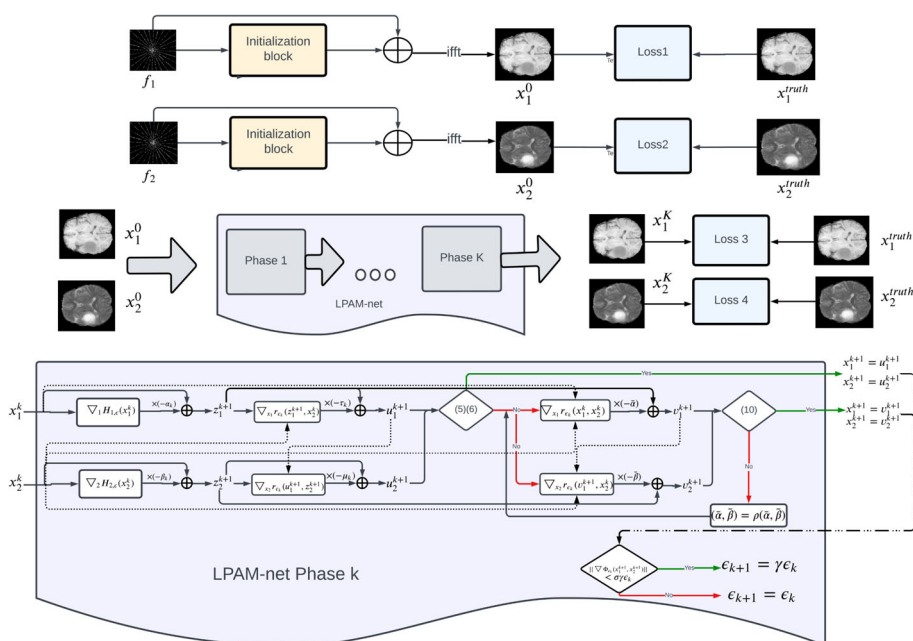
Fig. 1 The architecture of the propsoed Initialization network and the LPAM-net. Top: The proposed Initialization network; Middle: The architecture of the LPAM-net for joint reconstruction of T1 and T2 images; Bottom: The detailed illustration of the kth–phase of the LPAM-net

denoted as $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$. The outputs of the final phase of the LPAM-net are the resulting reconstructed images.

Throughout our experiments, the regularization we used is the $\ell_{2,1}$-norm of the joint feature extractor, which is described in the Section 7.1. The formula for the regularization term is $r(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{g}_\theta(\mathbf{x}_1, \mathbf{x}_2)||_{2,1}$, and the expressions of $\nabla_{1,2} r_\epsilon(\mathbf{x}_1, \mathbf{x}_2)$ and $r_\epsilon(\mathbf{x}_1, \mathbf{x}_2)$ are given in 51 and 54, respectively. We denote the feature extractor $\mathbf{g}_\theta(\mathbf{x}_1, \mathbf{x}_2)$ as $\mathbf{g}(\mathbf{x}_1, \mathbf{x}_2)$ or $\mathbf{g}$ for simplicity in the following text. Regarding the joint feature extractor $\mathbf{g}$, we parameterize it as 49 with component-wise activation σ as 50. The prefixed parameter δ in the activation function σ was set to be 0.01. The default configuration of the feature extraction operator $\mathbf{g}$ was set as follows: it consisted of $l = 4$ complex-kernel convolution layers. Every kernel in each layer has a dimension of $3\times3\times32$ (except for the first layer), where 32 is the depth of the kernel. In the first layer, the kernel size was $3\times3\times2$, with a depth of 2. For each layer, we had 32 kernels. We set the stride to be 1 and used zero-padding to preserve image size.

For joint reconstruction using the LPAM-net, we need to learn the step sizes α_k, τ_k, β_k and γ_k, the threshold ϵ_k, and the weights of the kernels of the feature extractor $\mathbf{g}$. We consolidated all the learned parameters into a unified symbol denoted as θ. The total number of the phases k is set to be 15. To enhance the stability and precision of the parameters before increasing the iteration count, we initially trained LPAM-net with phase number K=3 for 100 epochs. After completing the training of the phase K=3, an incremental strategy involving adding 2 more phases each time was employed and for K>3, we conducted training of 30 epochs during each iteration. Throughout the training, the batch size was set to be 1 and parameters were updated after the training for every batch.
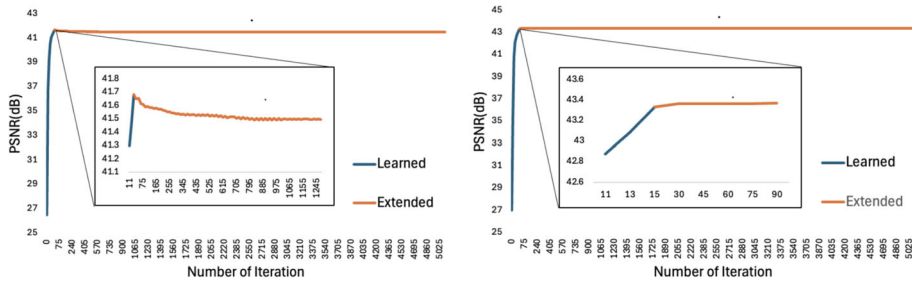
Fig. 2 From Left to Right: The PSNR values for the reconstruction results of the T1 and T2 image with a 20% under-sampling ratio at various phases. In both figures, the LPAM-net learns the network parameters in the first 15 phases. Starting from $\mathbf{X}_{16}$ the reconstruction results are obtained from the same model (48) with fixed **g** learned from the 15-phase LPAM-net. The PSNR values for the reconstruction results of the first 15 phases are plotted in blue, and since the 16th phase, that are plotted in orange

We updated the parameters θ by minimizing the loss function L' using Adam Optimizer implemented by Pytorch with $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate was set at 10^{-4}, and the initial step sizes were established as $\alpha_0 = \beta_0 = 0.5$. For τ_0 and μ_0, we have three initial step sizes for different phases. For phase 1 to phase 3, $\tau_0 = \mu_0 = 2$; for phase 4 to phase 12, $\tau_0 = \mu_0 = 1$; for phase 13 to phase 15, $\tau_0 = \mu_0 = 0.1$. The Xavier initialization was applied to initialize the kernels. The outputs for the algorithm were denoted as $\mathbf{x}_1^k$ and $\mathbf{x}_2^k$ and the ground truth were represented by $\mathbf{x}_1^{truth}$ and $\mathbf{x}_2^{truth}$. To calculate the loss, we converted the $m \times n$ image into a vector of length mn. Suppose the total pixel number of an image is mn, the loss function we minimized is expressed as $L' = Loss3 + Loss4$ where $Loss3$ and $Loss4$ are defined as follows respectively:

$$Loss3 = \frac{1}{mn}||\mathbf{x}_1^k - \mathbf{x}_1^{truth}||_2^2 + 0.1 \times (1 - SSIM(\mathbf{x}_1^k, \mathbf{x}_1^{truth}));$$

$$Loss4 = \frac{1}{mn}||\mathbf{x}_2^k - \mathbf{x}_2^{truth}||_2^2 + 0.1 \times (1 - SSIM(\mathbf{x}_2^k, \mathbf{x}_2^{truth})).$$

6.5 Convergence Behavior of the LPAM-net

In the previous subsection, we proved that there is at least a subsequence of the iterates generated by the LPAM algorithm converges to a Clarke stationary point. It is expected that LPAM would perform stably even beyond the trained phases. To demonstrate its stability empirically, we set the parameters as follows: $\epsilon_0 = 0.01$, $\gamma = 0.9$, $\sigma = 60000$ and the number of phases $K = 15$. We used the k-space data of a 20% under-sampling ratio for the joint reconstruction of MR T1 and T2 images. The ℓ_2-norm of the gradient of the objective function value ($\|\nabla_{1,2}\Phi_{\varepsilon_k}(\mathbf{x}_1^k, \mathbf{x}_2^k)\|$) decreases as the number of phase increases. According to the ϵ reduction criteria specified in the line 19 in the algorithm, this leads to a reduction in the smoothing factor from 0.01 to 0.0043 by phase 15. During the 15 phases, the smoothing factor steadily decreases as the number of phases increases, indicating that the smoothed objective function is progressively converging to the original problem.

Furthermore, we extend the algorithm beyond 15 phases and observe its performance on the values of the objective function and PSNR for the reconstruction. To this end, after 15 phases, we set $\epsilon_{tol} = 0$ and ran the algorithm for up to 5010 iterations with fixed g that is learned at the 15th phase. Figures 2 and 3 illustrate the changes in the values of PSNR
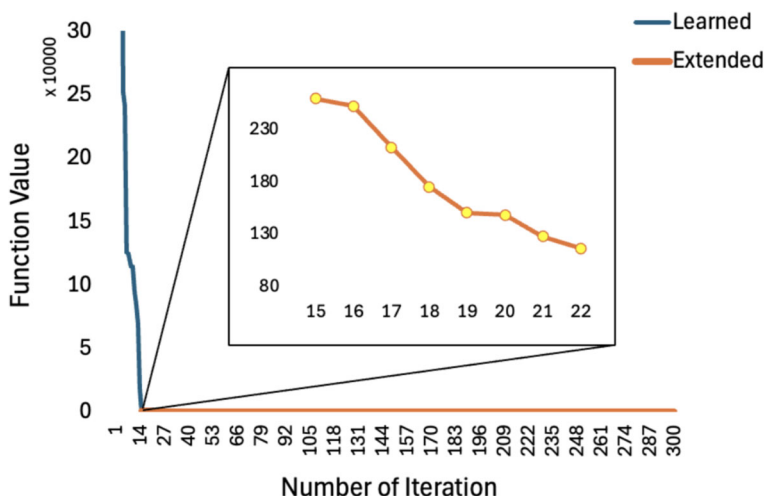
Fig. 3 Objective function value ($\Phi(\mathbf{x}_1^k, \mathbf{x}_2^k)$) of images reconstruction with a 20% under-sampling ratio across various phase number K. The results of the first 15 phases are plotted in blue and since the 16th phase, that are plotted in orange (Color figure online)

and objective function as the number of iterations increases, respectively. Figure 3 is drawn based on the objective function with 0.0093 as the coefficient of the regularization part. This coefficient does not affect the convergence based on Lemma 1. As we can tell, LPAM-net reduces the objective function value steadily in the first 15 phases. After that, the value of the objective function continue to decrease, but not as fast as the first 15 phases. The PSNR for both T1 and T2 images after the 15th phase maintain stable with slightly decrease in the first a few iterations. The Figure 4 visually shows the reconstructed images obtained after 15, 150, 1005, 5010 iterations. We can see that the reconstructed images are almost identical and free of visual artifacts, with very similar PSNR for both T1 and T2 contrasts. These evidence shows that LPAM-net maintains stable performance over extended iterations.

6.6 Results

In this subsection, we provide numerical results to showcase the enhanced image quality achieved by the proposed LPAM-net and the variational model 48. First, in order to show the advantage of using the joint features as the regularization term in the model, we make comparison between the LPAM-net and the Individual-modality Reconstruction Network. Next, we compare the LPAM-net with the network induced by BCD algorithm, which is equivalent to run only the lines 10-17 and skip the lines 3-9 in LPAM algorithm. Finally, we present the comparison of the LPAM-net with four state-of-art methods for joint reconstruction of MRI T1 and T1 images.

6.6.1 Comparison with Individual-modality Reconstruction Network

This experiment is designed to verify that using the joint features in the regularization term enhances the accuracy and efficiency of the reconstructions of the MRI T1 and T2 images. Since the LPAM-net takes advantage of the common features from both T1 and T2 contrasts,
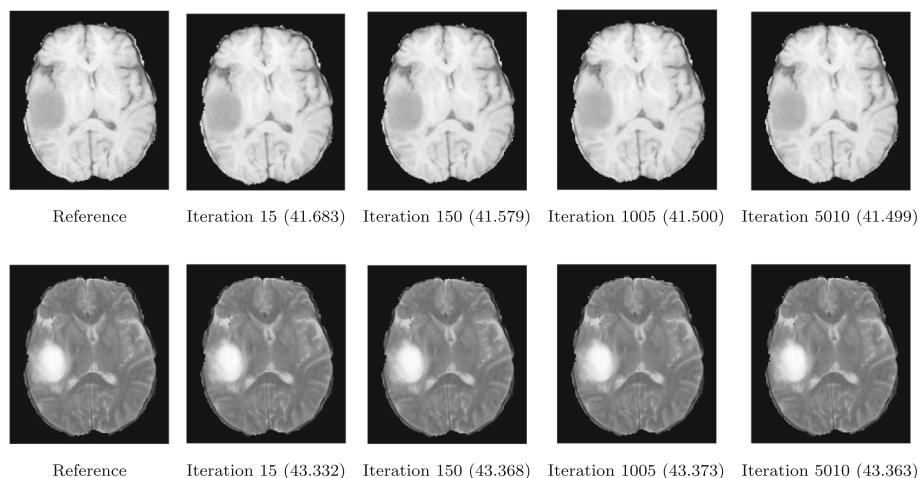
Fig. 4 Top row: Representation brain T1 MRI images reconstructed by the LPAM-net with a 20% undersampling ratio in the radial mask after 15, 150, 1005 and 5010 iterations; Bottom row: Representation brain T2 MRI images reconstructed by the LPAM-net with a 20% under-sampling ratio in radial mask after 15, 150, 1005 and 5010 iterations. PSNRs (dB) are shown in the parentheses

the proposed reconstructed images should preserve more details. To validate this advantage, we designed an Individual-modality Reconstruction Network which extracts the individual features of the T1 and T2 contracts separately. The Individual-modality Reconstruction Network precisely adheres the algorithm in [10] for minimizing the following problem:

$$\min_{\mathbf{x}_i} \|\mathbf{PF}\mathbf{x}_i - \mathbf{f}_i\|^2 + \|\mathbf{g}_{\theta_i}(\mathbf{x}_i)\|_{2,1},$$

where $\mathbf{x}_i \in \mathbb{R}^n$, $\mathbf{f}_i$ is the under-sampled k-space images and $i = 1, 2$. $\mathbf{g}_{\theta_i}$ is the individual feature extractor from $\mathbb{C}^n$ to $\mathbb{C}^{n \times d}$ and it is designed to be a 4-layer CNN. Unlike the joint reconstruction, in which we learn a joint feature extraction operator $\mathbf{g}_\theta$, in this individual reconstruction, we learn two feature extraction operators $\mathbf{g}_{\theta_1}$ and $\mathbf{g}_{\theta_2}$ separately. We learned the weights of the kernels in the CNNs by minimizing the loss function using Adam Optimizer implemented by Pytorch. Suppose the total pixel number of an image is mn, the loss function we minimized is expressed as follows:

$$Loss = \frac{1}{mn}\|\mathbf{x}_i^k - \mathbf{x}_i^{truth}\|_2^2 + 0.1 \times (1 - SSIM(\mathbf{x}_i^k, \mathbf{x}_i^{truth})),$$

where $\mathbf{x}_i^k$ is the output and $\mathbf{x}_i^{truth}$ is the ground truth image.

We compare T1 and T2 image reconstruction results with 10% under-sampling ratio k-space noiseless data resulting from LPAM-net and the Individual-modality reconstruction Network, respectively. Figure 5 shows the PSNR value at the phase number $K = 3, 5, ..., 15$ of both deep neural networks. Both networks were trained using Adam Optimizer implemented in Pytorch with $\beta_1 = 0.9$, $\beta_2 = 0.999$. The learning rate was set at 10^{-4} and the number of phase is set at 15. The average PSNR and its standard deviation are depicted in Figure 5 for each phase. For phase 3, Individual-modality Reconstruction Network outperforms, but as the phase number increases, the LPAM-net produces better reconstructions. At phase 15, the average PSNR of the LPAM-net improves 0.40 dB for the T1 images and improves 1.49 dB for the T2 images, respectively comparing to the Individual-modality Reconstruc-
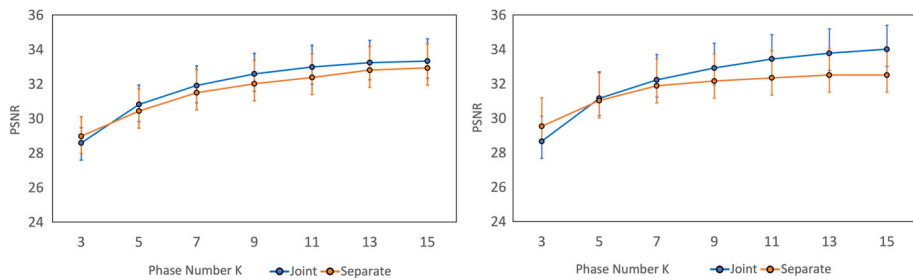
Fig. 5 PSNR for the reconstruction result of images with a 10% under-sampling ratio across various phase number K. Left: T1 images. Right: T2 images. In both figures, the blue line represents results for joint reconstruction via LPAM-net and the orange line represents the results from the Individual-modality Reconstruction Network. The vertical line represents the standard deviation of PSNR

tion Network. In addition, the standard deviation decreases 0.10 dB for the T1 images and decreases 0.21 dB for the T2 images.

Although PSNR is a crucial measure for comparing the two networks, it is hard for us to visually discern the impact that the LPAM-net has in the reconstruction process. In order to visualize the differences, we added Gaussian white noise to the complex-valued under-sampled k-space images. The real noise and the imaginary noise are both normally distributed with the same mean and the same standard deviation, but they maintain independence from each other. Then, we acquired three groups of T1, T2 images: one set is noiseless; one set has complex-valued noise with a mean of zero and a standard deviation of 3 on both the real part and imaginary part, and the third set has complex-valued noise with a mean of zero and a standard deviation of 7 on both the real part and imaginary part. Next, we trained and tested these three groups of T1,T2 images using both the LPAM-net and the Individual-modality Reconstruction Network. The resulting reconstructed T1 images from the three groups are presented in Figure 6 with the ground truth reference. Figure 7 illustrates the pointwise absolute error for each group of the images compared to the ground truth reference. Likewise, the reconstructed T2 images and the associated absolute errors are shown in Figures 8 and 9.

Figures 6 and 8 reveal that, for under-sampled images with any levels of Gaussian white noise, the edges of the tissues in the images reconstructed by the LPAM-net are sharper than the edges of the tissue reconstructed by the Individual-modality Reconstruction Network. Furthermore, the absolute errors are less in the images reconstructed from the LPAM-net in Figures 7 and 9. Tables 2 and 3 provide a comprehensive summary for the quantitative comparison between the LPAM-net and the Individual-modality Reconstruction Network across various levels of noise. The PSNR, SSIM, NMSE and RMSE are provided for images with each level of noise. We can conclude that the LPAM-net generates more accurate images than the Individual-modality Reconstruction Network regardless of the noise level in the under-sampled images. The results show the advantage of using the joint feature operator as the regularization term in the reconstruction process. Additionally, using joint feature operator demonstrates greater parameter efficiency.

6.6.2 Comparison with the BCD Algorithm

This experiment aims to demonstrate the effectiveness of the proposed LPAM algorithm by comparing to the standard BCD algorithm. The LPAM algorithm uses $\mathbf{u}_{k+1}$ to update the scheme to match the Res-net structure for better training, and $\mathbf{v}_{k+1}$ as a safeguard to ensure

Table 2 The results (mean ± standard deviation) of T1 image reconstruction using LPAM-net and Individual-modality Reconstruction Network, with the radial mask at a 10% under-sampling ratio, are presented for different variances of Gaussian white noises added to the k-space data

Method	SD of noise	PSNR	SSIM	NMSE	RMSE	#Par
Individual-modality	7	29.1356 ± 0.9087	0.8688 ± 0.0108	0.0051 ± 0.0016	0.0351 ± 0.0036	55903
LPAM-net	7	29.9429 ± 0.7945	0.8852 ± 0.0097	0.0042 ± 0.0011	0.032 ± 0.0028	56510
Individual-modality	3	31.2754 ± 1.1333	0.9044 ± 0.0098	0.0031 ± 0.0011	0.0275 ± 0.0034	55903
LPAM-net	3	31.994 ± 1.01	0.9175 ± 0.0091	0.0026 ± 0.0008	0.0253 ± 0.0028	56510
Individual-modality	none	33.0252 ± 1.3921	0.9251 ± 0.0094	0.0021 ± 0.0008	0.0226 ± 0.0034	55903
LPAM-net	none	33.3269 ± 1.2853	0.927 ± 0.0091	0.002 ± 0.0007	0.0218 ± 0.003	56510

Table 3 The results (mean ± standard deviation) of T2 image reconstruction using LPAM-net and Individual-modality Reconstruction Network, respectively, with the radial mask at a 10% under-sampling ratio, are presented for different variances of Gaussian white noises added to the k-space data

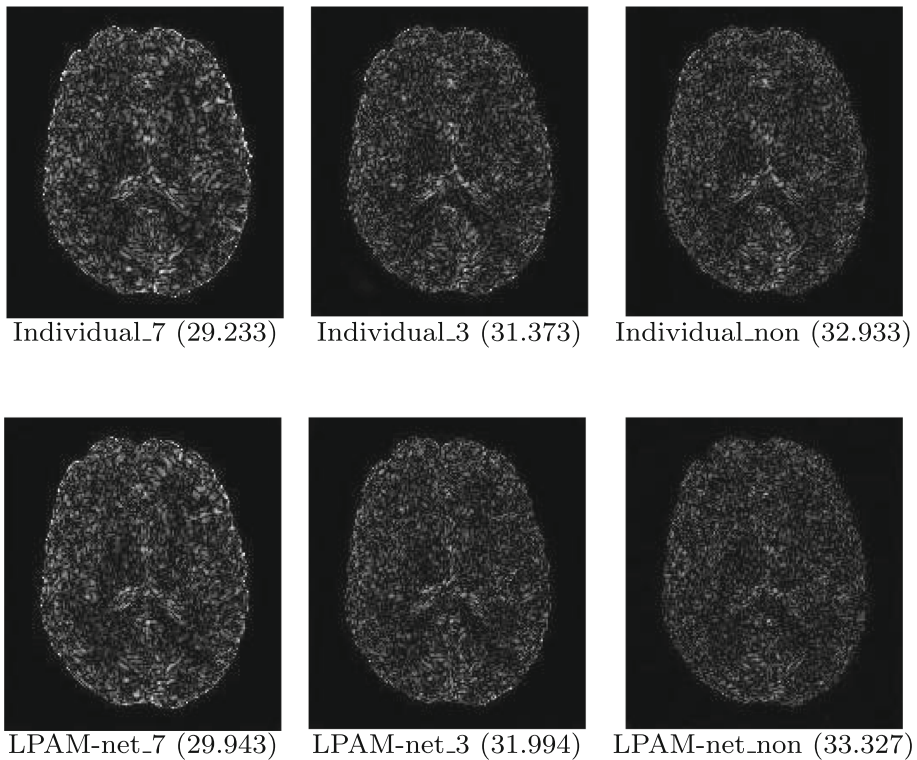| Method	SD of noise	PSNR	SSIM	NMSE	RMSE	#Par
Individual-modality	7	28.6758 ± 1.0697	0.8632 ± 0.016	0.0193 ± 0.0067	0.0371 ± 0.0046	55903
LPAM-net	7	29.8219 ± 1.107	0.88 ± 0.0178	0.015 ± 0.0055	0.0325 ± 0.0041	56510
Individual-modality	3	30.972 ± 1.2514	0.9049 ± 0.0149	0.0114 ± 0.0040	0.0286 ± 0.0041	55903
LPAM-net	3	32.1759 ± 1.0952	0.9209 ± 0.0147	0.0087 ± 0.0031	0.0248 ± 0.0031	56510
Individual-modality	none	32.621 ± 1.6011	0.9209 ± 0.0151	0.0078 ± 0.0028	0.0238 ± 0.0044	55903
LPAM-net	none	34.011 ± 1.3859	0.9392 ± 0.0132	0.0057 ± 0.0021	0.0202 ± 0.0032	56510
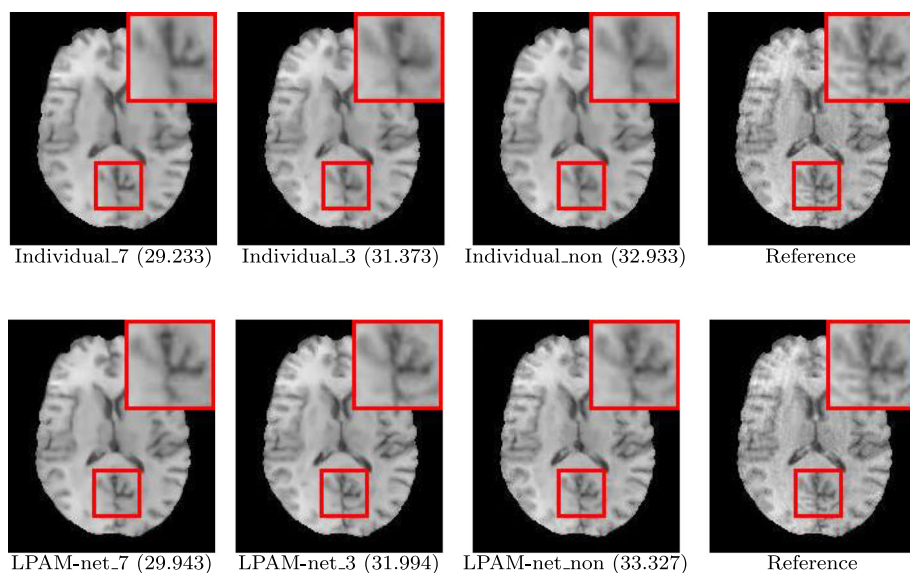
Fig. 6 Reconstruction results of T1 brain images by the LPAM-net and the Individual-modality Reconstruction Network employing the radial mask with a 10% under-sampling ratio. Top to bottom rows: Reconstruction results obtained by the Individual-modality Reconstruction Network and the LPAM-net, respectively. The first to the third columns: Images reconstructed from the k-spaces, where a complex noise with a standard deviation of 7, 3, and 0 (noiseless) is added, respectively. The last column: Ground truth reference. PSNRs (dB) are shown in the parentheses. The regions of interest are magnified in red boxes for better visualization

convergence. The standard BCD algorithm is equivalent to remove the iterations for updating $\mathbf{u}_{k+1}$ and only performing $\mathbf{v}_{k+1}$ in the LPAM. Although it converges for nonconvex smooth optimization, but the networks induced by BCD algorithm do not have Resnet structure. We conducted experiments comparing these two methods on under-sampled T1 and T2 images using the same variational model 48. Figures 10 and 11 illustrate the average PSNR and SSIM versus phase number with respect to the radial mask with a under-sampling ratio of 10% and 20%. As we can tell, both the average PSNR and the average SSIM of the two algorithms ascend, indicating an increase in image quality as phase number rises. For each phase, the values of the PSNR and the SSIM of the LPAM algorithm outperforms the values of the BCD algorithm in each graph, highlighting the efficiency and accuracy of the LPAM algorithm. The compared networks were trained using Adam Optimizer implemented in Pytorch with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and the learning rate was set at 10^{-4}.

### 6.6.3 Comparison with the State-of-the-Art

We compare the proposed algorithm LPAM-net with five representative methods, including the X-net[16], the Joint Group Sparsity-based Network (JGSN)[17], ReconFormer[18] and the Joint Cross-Attention Network (jCAN)[34]. X-net, originating from the U-net, introduces the capability to accommodate two inputs and generates two outputs and it is a non model-based method. JGSN, jCAN and ReconFormer are three model-based methods. JGSN unrolls the iterative process of the joint sparsity algorithm. jCAN deploys Vision Transformer and CNN in the image and k-space domains, respectively. ReconFormer designs Recurrent

Individual_7 (29.233)   Individual_3 (31.373)   Individual_non (32.933)

LPAM-net_7 (29.943)   LPAM-net_3 (31.994)   LPAM-net_non (33.327)

Fig. 7 The corresponding pointwise absolute errors between the reconstructed brain T1 images in Fig. 6 and the ground truth reference. The errors are scaled up by a factor of 4 for better visualization and the bright parts indicate large value

Pyramid Transformer at each architecture unit. Since the original codes for X-net and JGSN are not available, we implemented X-net and JGSN according to their original paper.

Similar to LPAM-net, X-net and JGSN update the two under-sampled modalities jointly. ReconFormer is a single-modality reconstruction network and we reconstructed T1 and T2 images respectively for comparison. jCAN restores the under-sampled target modality with guidance from the full-sampled auxiliary modality; thus, we used full-sampled T1 images to restore under-sampled T2 images and vice versa for comparison. Since some methods incorporate transformers, we adjusted the patch size and the window size to fit our image sizes. To ensure a fair evaluation, we trained each of the state-of-the-art method for 100 epoches and the quantitative analysis of the reconstructed images is presented in the Tables 4 and 5. Compared with the other methods, the proposed LPAM-net achieves competitive performance on both T1 and T2 images with a under-sampling ratio of 20%.
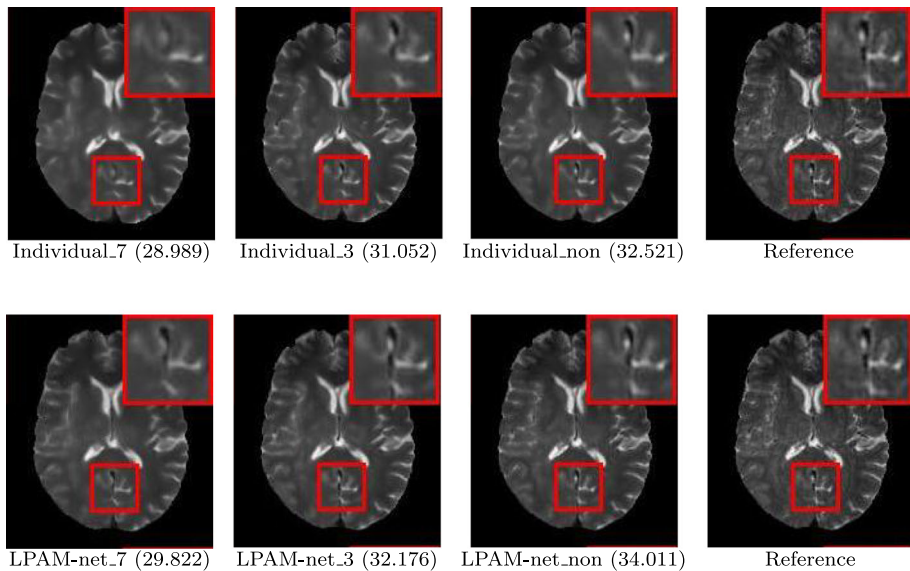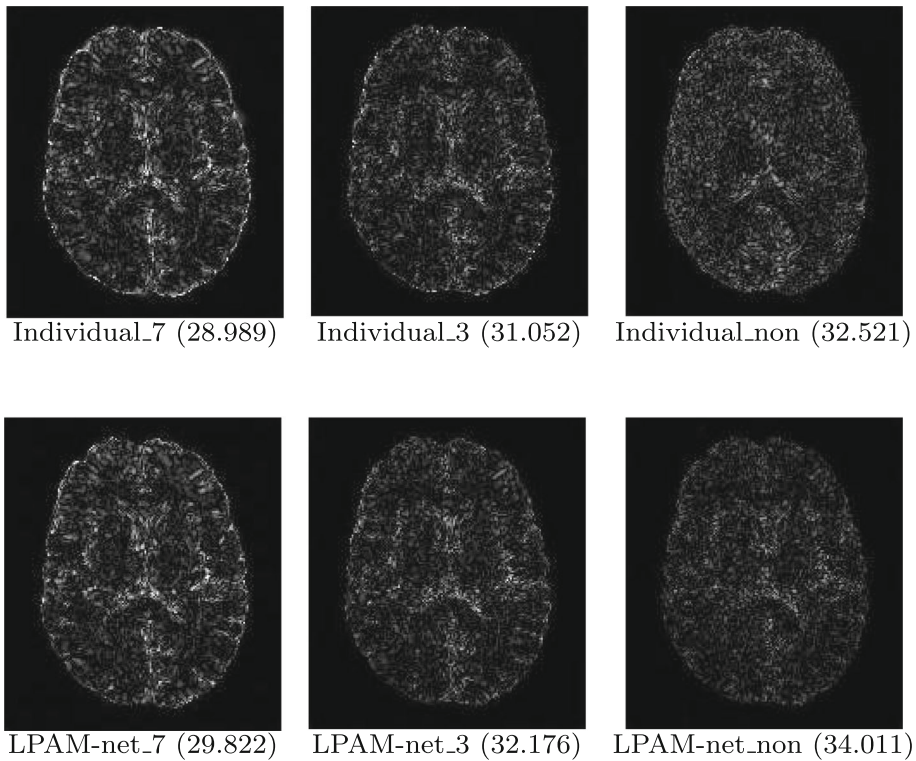
Fig. 8 Reconstruction results of T2 brain images by the LPAM-net and the Individual-modality Reconstruction Network employing the radial mask with a 10% under-sampling ratio. Top to bottom rows: Reconstruction results obtained by the Individual-modality Reconstruction Network and the LPAM-net, respectively. The first to the third columns: Images reconstructed from the k-spaces, where a complex noise with a standard deviation of 7, 3, and 0 (noiseless) is added, respectively. The last column: Ground truth reference. PSNRs (dB) are shown in the parentheses. The regions of interest are magnified in red boxes for better visualization

Table 4 The comparison (mean $\pm$ standard deviation) between LPAM-net and the state-of-the-art methods for T1 images with an under-sampling ratio of 20%

Method	PSNR	SSIM	NMSE	RMSE	#Par
X-net	32.713$\pm$ 0.93	0.93$\pm$ 0.006	0.002$\pm$ 0.0006	0.024$\pm$ 0.024	42.816M
JGSN	38.617$\pm$ 1.22	0.972$\pm$ 0.0048	0.0006$\pm$ 0.0002	0.012$\pm$ 0.0016	21192
jCAN	35.367$\pm$ 1.006	0.927$\pm$ 0.0087	0.0012$\pm$ 0.0004	0.017$\pm$ 0.0019	45.1M
ReconFormer	39.124$\pm$ 1.47	0.977$\pm$ 0.0047	0.0005$\pm$ 0.0002	0.011$\pm$ 0.0019	1.1M
LPAM-net	**40.66**$\pm$ 1.508	**0.983**$\pm$ 0.004	**0.0004**$\pm$ 0.0002	**0.0094**$\pm$ 0.0016	56510

Table 5 The comparison (mean $\pm$ standard deviation) between LPAM-net and the state-of-the-art methods for T2 images with an under-sampling ratio of 20%

Method	PSNR	SSIM	NMSE	RMSE	#Par
X-net	32.65$\pm$ 1.634	0.923$\pm$ 0.0106	0.0081$\pm$ 0.0033	0.024$\pm$ 0.005	42.816M
JGSN	39.3$\pm$ 1.413	0.975$\pm$ 0.0046	0.0017$\pm$ 0.0006	0.011$\pm$ 0.0017	21192
jCAN	37.583$\pm$ 1.513	0.964$\pm$ 0.0054	0.0025$\pm$ 0.001	0.013$\pm$ 0.0023	45.1M
ReconFormer	40.58$\pm$ 1.706	0.982$\pm$ 0.0051	0.0012$\pm$ 0.0004	0.0095$\pm$ 0.0018	1.1M
LPAM-net	**42.536**$\pm$ 1.527	**0.987**$\pm$ 0.0043	**0.0008**$\pm$ 0.0003	**0.0076**$\pm$ 0.0013	56510

Individual_7 (28.989)   Individual_3 (31.052)   Individual_non (32.521)

LPAM-net_7 (29.822)   LPAM-net_3 (32.176)   LPAM-net_non (34.011)

Fig. 9 The corresponding pointwise absolute errors between the reconstructed brain T2 images in Fig. 8 and the ground truth reference. The errors are scaled up by a factor of 4 for better visualization and the bright parts indicate large value
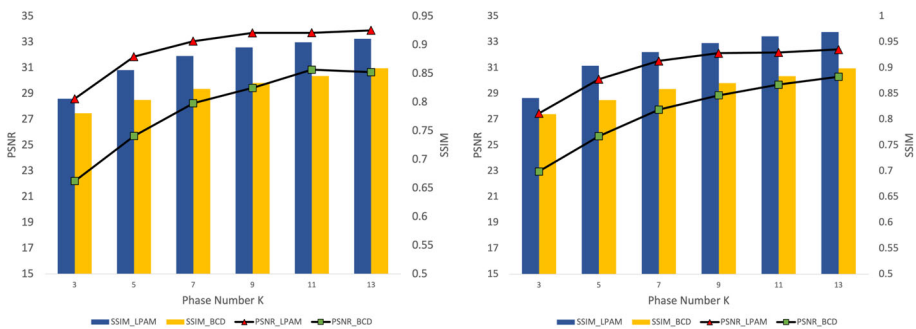
Fig. 10 Average PSNR and SSIM of the reconstruction results for images with a 10% under-sampling ratio obtained by the LPAM algorithm and the BCD algorithm across various phase number K. Left: T1 images. Right: T2 images
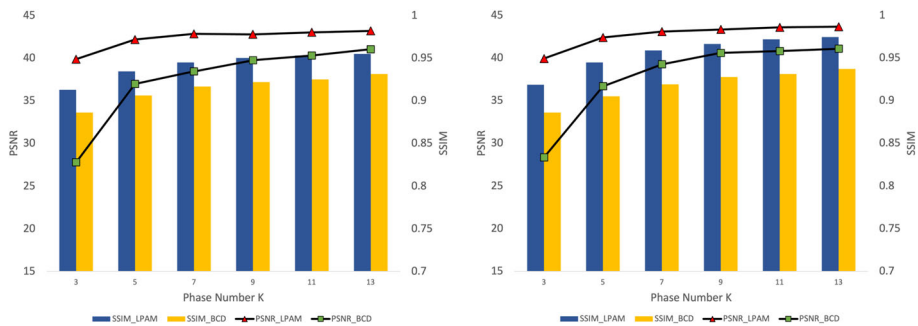
Fig. 11 Average PSNR and SSIM of the reconstruction results for images with a 20% under-sampling ratio obtained by the LPAM algorithm and the BCD algorithm across various phase number K. Left: T1 images. Right: T2 images

Data Availability The original dataset we used in all experiments are from *Multi-modal Brain Tumor Segmentation Challenge 2018*.[25]

Declarations

Conflict of interest The authors declare that they have no Conflict of interest.

References

1. Abolghasemi, V., Ferdowsi, S., Sanei, S.: A gradient-based alternating minimization approach for optimization of the measurement matrix in compressive sensing. Signal Process. **92**(4), 999–1009 (2012)
2. Aggarwal, H.K., Mani, M.P., Jacob, M.: Modl: model-based deep learning architecture for inverse problems. IEEE Trans. Med. Imaging **38**(2), 394–405 (2018)
3. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the kurdyka-łojasiewicz inequality. Math. Oper. Res. **35**(2), 438–457 (2010)
4. Attouch, H., Bolte, J., Svaiter, B.F.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized gauss-seidel methods. Math. Program. **137**(1–2), 91–129 (2013)
5. Beck, A., Tetruashvili, L.: On the convergence of block coordinate descent type methods. SIAM J. Optim. **23**(4), 2037–2060 (2013)
6. Bertsekas, D., Tsitsiklis, J.: Parallel and distributed computation: numerical methods. Athena Scientific (2015)
7. Bian, W., Zhang, Q., Ye, X., Chen, Y.: A learnable variational model for joint multimodal mri reconstruction and synthesis. Lect. Notes Comput. Sci. **13436**, 354–364 (2022). https://doi.org/10.1007/978-3-031-16446-0_34
8. Bolte, J., Sabach, S., Teboulle, M.: Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math. Program. **146**(1–2), 459–494 (2014)
9. Charbonnier, P., Blanc-Féraud, L., Aubert, G., Barlaud, M.: Deterministic edge-preserving regularization in computed imaging. IEEE Trans. Image Process. **6**(2), 298–311 (1997)
10. Chen, Y., Liu, H., Ye, X., Zhang, Q.: Learnable descent algorithm for nonsmooth nonconvex image reconstruction. SIAM J. Imag. Sci. **14**(4), 1532–1564 (2021). https://doi.org/10.1137/20M1353368
11. Chun, I.Y., Fessler, J.A.: Convolutional analysis operator learning: acceleration and convergence. IEEE Trans. Image Process. **29**, 2108–2122 (2020)
12. Chun, I.Y., Huang, Z., Lim, H., Fessler, J.: Momentum-net: Fast and convergent iterative neural network for inverse problems. IEEE transactions on pattern analysis and machine intelligence (2020)

13. Chun, I.Y., Zheng, X., Long, Y., Fessler, J.A.: Bcd-net for low-dose ct reconstruction: Acceleration, convergence, and generalization. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part VI 22, pp. 31–40. Springer (2019)

14. Chun, Y., Fessler, J.A.: Deep bcd-net using identical encoding-decoding cnn structures for iterative image recovery. In: 2018 IEEE 13th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), pp. 1–5. IEEE (2018)

15. Ding, C., Zhang, Q., Wang, G., Ye, X., Chen, Y.: Learned alternating minimization algorithm for dual-domain sparse-view ct reconstruction. arXiv preprint arXiv:2306.02644 (2023)

16. Do, W.J., Seo, S., Han, Y., Ye, J.C., Choi, S.H., Park, S.H.: Reconstruction of multicontrast mr images through deep learning. Med. Phys. **47**(3), 983–997 (2020)

17. Guo, D., Zeng, G., Fu, H., Wang, Z., Yang, Y., Qu, X.: A joint group sparsity-based deep learning for multi-contrast mri reconstruction. J. Magn. Reson. **346**, 107354 (2023)

18. Guo, P., Mei, Y., Zhou, J., Jiang, S., Patel, V.M.: Reconformer: Accelerated mri reconstruction using recurrent transformer. IEEE transactions on medical imaging (2023)

19. Hardt, M.: Understanding alternating minimization for matrix completion. In: 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, pp. 651–660. IEEE (2014)

20. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)

21. Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pp. 665–674 (2013)

22. Kaiming He Xiangyu Zhang, S.R., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition p. 770-778 (2016)

23. Kaiming He Xiangyu Zhang, S.R., Sun, J.: Deep residual learning for image recognition. European conference on computer vision pp. 630–645 (2016). https://doi.org/10.1007/978-3-319-46493-0_38

24. Kim, Y., Jung, H., Min, D., Sohn, K.: Deeply aggregated alternating minimization for image restoration. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 6419–6427 (2017)

25. Menze, B.H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al.: The multimodal brain tumor image segmentation benchmark (brats). IEEE Trans. Med. Imaging **34**(10), 1993–2024 (2014)

26. Nesterov, Y., et al.: Lectures on convex optimization, vol. 137. Springer, Berlin (2018)

27. Netrapalli, P., Jain, P., Sanghavi, S.: Phase retrieval using alternating minimization. Advances in Neural Information Processing Systems **26** (2013)

28. Palomar, D.P., Eldar, Y.C.: Convex optimization in signal processing and communications. Cambridge University Press, Cambridge (2010)

29. Peters, S.W., Heath, R.W.: Interference alignment via alternating minimization. In: 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 2445–2448. IEEE (2009)

30. Pock, T., Sabach, S.: Inertial proximal alternating linearized minimization (ipalm) for nonconvex and nonsmooth problems. SIAM J. Imag. Sci. **9**(4), 1756–1787 (2016). https://doi.org/10.1137/16M1064064

31. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. **4**(5), 1–17 (1964)

32. Sroubek, F., Milanfar, P.: Robust multichannel blind deconvolution via fast alternating minimization. IEEE Trans. Image Process. **21**(4), 1687–1700 (2011)

33. Sun, J., Li, H., Xu, Z., et al.: Deep admm-net for compressive sensing mri. Advances in neural information processing systems **29** (2016)

34. Sun, K., Wang, Q., Shen, D.: Joint cross-attention network with deep modality prior for fast mri reconstruction. IEEE Transactions on Medical Imaging (2023)

35. Zhang, Q., Alvandipour, M., Xia, W., Zhang, Y., Ye, X., Chen, Y.: Provably convergent learned inexact descent algorithm for low-dose ct reconstruction (2021). https://doi.org/10.48550/ARXIV.2104.12939. arXiv:2104.12939