remote sensing

MDPI

Article

Self-Supervised Deep Hyperspectral Inpainting with Plug-and-Play and Deep Image Prior Models

Shuo Li [1,*] and Mehrdad Yaghoobi [2]

[1] School of Engineering, University of Edinburgh, Edinburgh EH8 9YL, UK
[2] Institute of Imaging, Data and Communications, University of Edinburgh, Edinburgh EH8 9YL, UK; m.yaghoobi-vaighan@ed.ac.uk
* Correspondence: s1809498@ed.ac.uk

Abstract: Hyperspectral images are typically composed of hundreds of narrow and contiguous spectral bands, each containing information regarding the material composition of the imaged scene. However, these images can be affected by various sources of noise, distortions, or data loss, which can significantly degrade their quality and usefulness. This paper introduces a convergent guaranteed algorithm, LRS-PnP-DIP(1-Lip), which successfully addresses the instability issue of DHP that has been reported before. The proposed algorithm extends the successful joint low-rank and sparse model to further exploit the underlying data structures beyond the conventional and sometimes restrictive unions of subspace models. A stability analysis guarantees the convergence of the proposed algorithm under mild assumptions, which is crucial for its application in real-world scenarios. Extensive experiments demonstrate that the proposed solution consistently delivers visually and quantitatively superior inpainting results, establishing state-of-the-art performance.

Keywords: low rank; sparsity; hyperspectral image inpainting; self-supervised learning; fixed-point convergence

check for updates

Academic Editor: Salah Bourennane

Received: 6 December 2024
Revised: 9 January 2025
Accepted: 10 January 2025
Published: 15 January 2025

Citation: Li, S.; Yaghoobi, M. Self-Supervised Deep Hyperspectral Inpainting with Plug-and-Play and Deep Image Prior Models. *Remote Sens.* **2025**, *17*, 288. https://doi.org/10.3390/rs17020288

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Hyperspectral remote sensing has found widespread application in numerous fields such as astronomy, agriculture, environmental monitoring, and Earth observation. Hyperspectral images (HSIs) are often captured via satellite or airborne sensors, each presenting samples at different time slots, using push-broom strategies along the flying pathway [1]. The nature of the HSI acquisition system makes the HSI a high-resolution 3D data cube that covers hundreds or thousands of narrow spectral bands, conveying a wealth of spatio-spectral information. However, due to instrumental errors, imperfect navigation and atmospheric variations, practical HSIs can suffer from noise, and missing pixels or even entire lines of pixels (an example of this type of missing data can be found in the recently released Earth Surface Mineral Dust Source Investigation (EMIT) hyperspectral dataset, developed by NASA's Jet Propulsion Laboratory and launched on 14 July 2022 [2]). These issues can severely impact the accuracy and reliability of subsequent analyses, making HSI inpainting a critical task in the field of remote sensing and Earth observation.

1.1. Hyperspectral Image Inpainting

Hyperspectral image inpainting refers to the restoration of missing or corrupted data in acquired HS images. Unlike RGB images, for which inpainting involves filling a single pixel value (red, green, and blue channels), HSI inpainting requires filling in a complex vector that contains extensive spectral information. This additional complexity makes the

already challenging task of HSI inpainting even more difficult. The primary objective of HSI inpainting is to create visually convincing structures while ensuring that the texture of the missing regions is spectrally coherent. The inpainting of HS images typically involves two main steps: (a) estimating the missing or corrupted data by leveraging information from the neighboring spectral bands and spatially adjacent pixels and (b) refining the estimated data to ensure consistency with the overall spectral and spatial properties of the HS image. Traditional approaches, such as those proposed by [3], assume that the spectral vectors of the HSI always exist in some unknown low-dimensional subspaces, and the missing regions are estimated through projections. However, the performance of such techniques deteriorates significantly when large portions of the image are missing, and they may even fail when all spectral bands are absent. Another line of research involves the tensor completion process to restore incomplete observations [4–6]. These methods leverage the spatio-spectral correlations of the hyperspectral data cube, offering potential improvements over subspace-based techniques.

Over the past decade, the success of deep learning has brought new opportunities for solving HSI inpainting tasks. In [7,8], the authors proposed transformer-based frameworks to effectively capture both local and global contextual information in 3D images, demonstrating the potential of attention modules in exploring relationships between missing pixels and background pixels. In [9], the authors showed that missing pixels can be predicted using generative neural networks. However, this approach has two key limitations: (1) it requires training the networks on extensive datasets to achieve the desired performance, and (2) the inpainted areas often suffer from over-smoothing, resulting in the loss of critical information, such as abrupt changes in the surface materials. More recently, researchers discovered that the network's structure itself may serve as a good inductive bias for regularizing image reconstruction tasks, such as image inpainting [10]. A deep image prior (DIP) enables the learning of priors directly from the image without the need for extensive training datasets, which was further developed and successfully applied to hyperspectral (HS) images [11]. Following [11], subsequent works [12–18] have shown that certain trained or untrained neural networks can be directly incorporated into iterative solvers to improve the reconstruction accuracy, i.e., as the regularizer. More recently, the authors of [19] introduced a self-supervised framework that leverages the powerful learning capabilities of diffusion models and sets new state-of-the-art benchmarks across several HSI reconstruction tasks. However, as it was reported in our preliminary report [20], as well as in a series of DIP-related works [13,21–23], the DIP may not always be the ideal solution, despite promising empirical results. On the practical side, this is because a DIP works by directly fitting a neural network to the corrupted HS image; in the absence of regularization or explicit denoising mechanisms, a DIP tends to overfit, which will eventually capture the noise in the image. On the theoretical side, a DIP lacks a strong theoretical foundations to explain its performance, and its convergence is highly sensitive to random initialization of the neural network. Different initializations could potentially lead to poor results. Although subsequent works [12,13,24] have attempted to mitigate this issue by introducing a plug-and-play (PnP) denoiser into the training of a DIP, they require either a suitable early stopping criterion or the manual control of the learning rate. As a result, there remains a need for a new HSI inpainting algorithm that combines the strengths of both PnP and DIP methods while offering convergence guarantees.

1.2. Sparsity and Low Rankness in HSIs

Despite the high dimensionality, HS images exhibit intrinsic sparse and low-rank structures due to the high correlation between spatial pixels and spectral channels. This property has made sparse representation—(SR) and low rank—(LR) based methods popular

for the processing of HSIs [25–27]. SR relies on the key assumption that the spectral signatures of the pixels approximately lie in a low-dimensional subspace spanned by representative pixels from the same class. With a given dictionary, SR allows for sparse decomposition of HSIs into a linear combination of several atoms, thus exploiting the spatial similarity of HSIs (It is worth mentioning that, when the dictionary is not given, we can manually use the end-members of some pixels with pure spectra and build the dictionary, or learn the dictionary [28] in an unsupervised manner, using sparsity as a regularizer). This assumption about the underlying structures has been effectively applied in various HSI tasks, including HSI classification [29], denoising [30], and unmixing [31].

LR is another widely used regularizer in HSI processing, which assumes that pixels in the clean HSI image have high correlations in the spectral domain, thus capturing both the spectral similarity and global structure of HSIs. LR allows for the modeling of the HSI as a low-rank matrix; thus, the missing and corrupted information in the HSIs can be effectively reconstructed by leveraging the underlying low-rank structure. In many state-of-the-art HSI reconstruction algorithms, LR is often used in parallel with SR to achieve better reconstruction accuracy [30,32]. By combining the two priors together, algorithms can better handle missing data, noise, and corruption while maintaining the overall structure and spectral integrity of the image. For more detailed information on LR and SR of hyperspectral images, we refer readers to the review work in [33].

However, a notable limitation with such an LR or SR model is that the real HS data do not exactly follow the sparse and/or low-dimension assumptions.; e.g., there are long trails (numerous small but non-zero elements) in the descending sorted absolute sparse coefficients and singular values, where overly-restrictive assumptions on these coefficients could lead to under-fitting. In conclusion, both LR and SR rely on some predefined mathematical models that assume certain properties of the HS image, such as low-dimensional manifolds (low-rankness) or sparse representations in specific domains. While these models are useful, they struggle to capture more complex patterns in the real HS data, such as non-linear relationships and complex spatio-spectral dependencies. Moreover, when a significant percentage of the data are corrupted or entire spectral bands of pixels are missing, these methods often fail [3]. The question of how deep neural models can address this issue by exploiting more data-adaptive, low-dimensional models in a self-supervised learning framework is the main topic of this study.

In this paper, we start by giving a brief overview of the LRS-PnP-DIP algorithm [20]; we replaced the denoising/thresholding step with the PnP denoiser, and we used a DIP to replace the low-rank component of the formulation. Since there remains a lack of comprehensive theoretical analysis of the DIP and its extensions that can guarantee the convergence, we demonstrate both theoretically and empirically that small modifications can ensure the convergence of LRS-PnP and LRS-PnP-DIP, while the original algorithms in [20] may diverge. We propose a variant of the LRS-PnP-DIP algorithm dubbed LRS-PnP-DIP(1-Lip), which not only outperforms existing learning-based methods but also resolves the instability issue of a DIP in an iterative algorithm.

1.3. Contributions

This paper presents a novel HS inpainting algorithm that leverages the learning capability of deep networks while providing insights into the convergence behavior of such complex algorithms. Through rigorous mathematical analysis, we derive sufficient conditions for stability and fixed-point convergence and show that these conditions are satisfied by slightly modified algorithms. Our approach offers several key contributions, including the following:

- This paper explores the LRS-PnP and LRS-PnP-DIP algorithm in more depth by showing the following: (1) The sparsity and low-rank constraints are both important to the success of the methods in [20]. (2) DIP can better explore the low-rank subspace compared to conventional subspace models such as singular value thresholding (SVT).
- Under some mild assumptions, a fixed-point convergence proof is provided for the LRS-PnP-DIP algorithm (see Theorem 1). We introduce a variant to the LRS-PnP-DIP called LRS-PnP-DIP(1-Lip), which effectively resolves the instability issue of the algorithm by slightly modifying both the DIP and PnP denoiser.
- To the best of our knowledge, this is the first time the theoretical convergence of PnP with DIP replaced the low-rank prior being analyzed under an iterative framework.
- Extensive experiments were conducted on real-world data to validate the effectiveness of the enhanced LRS-PnP-DIP(1-Lip) algorithm with a convergence certificate. The results demonstrate the superiority of the proposed solution over existing learning-based methods in both stability and performance.

The remainder of this paper is organized as follows: Section 2 provides an introduction to our approach, and Section 3 describes our main convergence theory for the proposed LRS-PnP-DIP(1-Lip) algorithm. Section 4 provides the implementation details and discusses the experimental results. Section 5 concludes the paper with a brief discussion of potential future directions.

2. Proposed Methods

Mathematical Formulations

The task of HSI inpainting can be formulated as reconstructing the clean image, X, from a noisy and incomplete measurement, Y, where an additive noise, N, and a masking operator, M, are present:

$$Y = \mathcal{M}\{X\} + N \tag{1}$$

The clean image $X \in \mathbb{R}^q$ (where $q = n_r \times n_c \times n_b$), with n_r and n_c representing the spatial dimensions of the image, and n_b represents the total number of spectral bands. The operator $\mathcal{M} \in \mathbb{R}^{q \times q}$ is a binary mask, where zero represents the missing pixels, and one represents the observed and valid pixels. Thus, the masking operator $\mathcal{M}$ can be represented as a diagonal square matrix. N is the additive Gaussian noise of appropriate size. Usually, M is provided, and the formulation (1) describes a linear system, which can be expressed as follows:

$$\boldsymbol{y} = \mathrm{M}\boldsymbol{x} + \boldsymbol{n} \tag{2}$$

In this equation, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{n}$ represent the vectorized forms of X, Y, and N, respectively, and M is a diagonal matrix. We obtain the recovered, i.e., inpainted, image, $\boldsymbol{x}^*$, by applying sparse representation to each image patch, $P_i(\boldsymbol{x})$. Here, the operator $P_i(\cdot)$ extracts the i-th patch from image $\boldsymbol{x}$, where each patch may consist of only valid pixels or a combination of valid and missing pixels, depending on the size of $P_i(\cdot)$. The inpainted image $\boldsymbol{x}^*$ can be obtained by solving the following optimization problem:

$$\begin{aligned}(\boldsymbol{x}^*, \boldsymbol{\alpha}^*) = \underset{\boldsymbol{x},\boldsymbol{\alpha}}{\operatorname{argmin}}\, \gamma\|\boldsymbol{y} - \mathrm{M}\boldsymbol{x}\|_2^2 + w_{lr}\|\boldsymbol{x}\|_* \\ + \|\sum_i (P_i(\boldsymbol{x}) - \Phi\boldsymbol{\alpha}_i)\|_2^2 + w_s \sum_i \|\boldsymbol{\alpha}_i\|_1\end{aligned} \tag{3}$$

The objective in the expression above consists of three terms. The first term is the data fidelity term, weighted by the parameter γ. Since estimating $\boldsymbol{x}$ from $\boldsymbol{y}$ is inherently ill-posed, i.e., with more unknown variables than equations, the solution is non-unique. To address this, we introduce two additional "priors" to regularize the inpainting problem:

low-rank and sparsity constraints. The second term penalizes solution $\boldsymbol{x}$ to be low-rank, which is typically employed as a surrogate for rank minimization. The third term restricts the missing pixels to be generated from the subspace approximated by the valid pixels. These terms are weighted by parameters w_{lr} and w_s, respectively. The sparse representation problem is solved using a given dictionary, Φ. Specifically, Φ can be constructed either by using the end-members of some pixels with pure spectra or by learning from the noisy pixels in the observations. It is here exclusively learned from noisy pixels in the observations using online dictionary learning [28]. By adopting the augmented Lagrangian and introducing the auxiliary variable $\boldsymbol{u}$ [34], problem (3) can be rewritten as follows:

$$\begin{aligned}(\boldsymbol{x}^*, \boldsymbol{\alpha}^*) = &\underset{\boldsymbol{x},\boldsymbol{\alpha}}{\operatorname{argmin}}\, \gamma\|\boldsymbol{y} - \mathrm{M}\boldsymbol{x}\|_2^2 + w_{lr}\|\boldsymbol{u}\|_* + w_s \sum_i \|\boldsymbol{\alpha}_i\|_1 \\ &+ \frac{\boldsymbol{\mu}_1}{2}\|\sum_i (P_i(\boldsymbol{x}) - \Phi\boldsymbol{\alpha}_i) + \frac{\boldsymbol{\lambda}_1}{\boldsymbol{\mu}_1}\|_2^2 \\ &\text{s.t.} \quad \boldsymbol{x} = \boldsymbol{u}\end{aligned} \tag{4}$$

The Lagrangian multiplier and penalty terms are denoted as $\boldsymbol{\lambda}_1$ and $\boldsymbol{\mu}_1$, respectively. Using the alternating direction method of multipliers (ADMM), we can sequentially update the three variables $\boldsymbol{\alpha}$, $\boldsymbol{u}$, and $\boldsymbol{x}$ to solve problem (4):

(1) *Fixing* $\boldsymbol{u}$ *and* $\boldsymbol{x}$*, and updating* $\boldsymbol{\alpha}$:

$$\begin{aligned}\boldsymbol{\alpha}^{k+1} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \frac{\boldsymbol{\mu}_1^k}{2} \sum_i \|(P_i(\boldsymbol{x}^k) + \frac{\boldsymbol{\lambda}_1^k}{\boldsymbol{\mu}_1^k}) - \Phi\boldsymbol{\alpha}_i\|_2^2 \\ + w_s \sum_i \|\boldsymbol{\alpha}_i\|_1\end{aligned} \tag{5}$$

The problem in Equation (5) is a patch-based, sparse coding problem that can be solved using iterative solvers. Let us denote the first term on the right-hand side of Equation (5) as $f(\boldsymbol{\alpha})$ and the second term as $g(\boldsymbol{\alpha})$ while dropping the subscript for simplicity. Problem (5) can be represented as a constrained optimization problem:

$$\begin{aligned}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{v}}) = &\underset{\boldsymbol{\alpha},\boldsymbol{v}}{\operatorname{argmin}}\, f(\boldsymbol{\alpha}) + w_s g(\boldsymbol{v}) \\ &\text{s.t.} \quad \boldsymbol{\alpha} = \boldsymbol{v}\end{aligned} \tag{6}$$

By introducing an auxiliary variable, $\boldsymbol{u}$, and a multiplier, ρ, the augmented Lagrangian of (6) becomes the following:

$$\mathcal{L}(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{u}}) = f(\boldsymbol{\alpha}) + w_s g(\boldsymbol{v}) + \boldsymbol{u}^T(\boldsymbol{\alpha} - \boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{\alpha} - \boldsymbol{v}\|_2^2 \tag{7}$$

Then, ADMM finds solutions by breaking the constrained optimization problem (6) into several sub-problems and updating them separately in an iterative fashion. The pseudocode is provided in Algorithm 1.

The $\boldsymbol{v}^{k+1}$ sub-problem can be solved by the proximal operator [35]. The authors in [36] realized that $\boldsymbol{\alpha}^{k+1} + \boldsymbol{u}^k$ can be treated as a "noisy" version of $\boldsymbol{v}$, and $g(\boldsymbol{v})$ can be seen as a regularization term. Then, the whole process resembles a denoising operation applying to the intermediate results, which can be simply replaced with the PnP denoisers such as BM3D and the non-local mean (NLM) denoiser. In our approach, we employed PnP-ISTA [36] as an alternative solution for solving problem (5). We denote the gradient of f as ∇f and I as an Identity matrix with an appropriate shape. The entire process can then be replaced with an off-the-shelf denoiser $\mathcal{D}$, operating on ∇f [36]. Each iteration takes the following form:

$$\boldsymbol{\alpha}^{k+1} = \mathcal{D}(\mathrm{I} - \nabla f)(\boldsymbol{\alpha}^k) \tag{8}$$

PnP-ISTA can significantly improve the reconstruction quality over traditional ISTA [37] due to the following reasons: (1) it enables the integration of state-of-the-art denoisers to handle different noise patterns and different types of images, which is particularly flexible, as one can apply any off-the-shelf denoiser without massively changing the existing framework; and (2) it allows the use of denoisers tailored to different problems (e.g., a denoiser that promotes sparsity or smoothness, or even a denoising neural network that encodes more complex priors).

Algorithm 1 Pseudocode of ADMM for solving problem (6).

Initialization $w_s, \rho, k \leftarrow 0, \boldsymbol{\alpha}^k \leftarrow 0, \boldsymbol{u}^k \leftarrow 0, \boldsymbol{v}^k \leftarrow 0$.
while Not Converged **do**
  $\boldsymbol{\alpha}^{k+1} \leftarrow \mathrm{argmin}_{\boldsymbol{\alpha}}\, f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{\alpha} - (\boldsymbol{v}^k - \boldsymbol{u}^k)\|_2^2$.
  $\boldsymbol{v}^{k+1} \leftarrow \mathrm{argmin}_{\boldsymbol{v}}\, w_s g(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - (\boldsymbol{\alpha}^{k+1} + \boldsymbol{u}^k)\|_2^2$.
  $\boldsymbol{u}^{k+1} \leftarrow \frac{1}{\rho}\boldsymbol{u}^k + (\boldsymbol{\alpha}^{k+1} - \boldsymbol{v}^{k+1})$.
end while

(2) *We can update* $\boldsymbol{u}$ *in this setting by*:

$$\boldsymbol{u}^{k+1} = \underset{\boldsymbol{u}}{\mathrm{argmin}}\, w_{lr}\|\boldsymbol{u}\|_* + \frac{\boldsymbol{\mu_2}^k}{2}\|(\boldsymbol{x}^k + \frac{\boldsymbol{\lambda_2}^k}{\boldsymbol{\mu}_2^k}) - \boldsymbol{u}\|_2^2 \tag{9}$$

where $\boldsymbol{\lambda}_2$ and $\boldsymbol{\mu}_2$ are the corresponding Lagrangian multiplier and penalty term, respectively, which we use subscripts to distinguish between $\boldsymbol{\lambda}_1$ and $\boldsymbol{\mu}_1$ in step (4). The problem can be solved using the singular value thresholding (SVT) algorithm [38] as follows:

$$\boldsymbol{u}^{k+1} = \mathcal{SVT}(\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}) \tag{10}$$

Specifically, denote $\frac{w_{lr}}{\boldsymbol{\mu}_2^k}$ as τ, matrix $\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}$ as A, and the soft thresholding operator $\mathcal{S}_\tau$ as follows:

$$\mathcal{S}_\tau(x) = sign(x) \cdot max(|x - \tau|, 0) \tag{11}$$

where $sign(\cdot)$ is an operator taking the positive part of its input. Applying (11) to the singular values of A promotes low-rankness:

$$\boldsymbol{u}^{k+1} = \mathcal{SVT}(\mathrm{A}) = \mathrm{U_A}\mathcal{S}_\tau(\Sigma_\mathrm{A})\mathrm{V_A^T} \tag{12}$$

In the proposed LRS-PnP-DIP algorithm [20], this step is substituted with a deep image prior (DIP) $f_\theta(z)$, where θ denotes the network parameters that need to be updated, and the input z is set to $\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}$; i.e., the latent image from the previous iterations:

$$\boldsymbol{u}^{k+1} = f_\theta(\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}) \tag{13}$$

(3) *Fixing* α *and* $\boldsymbol{u}$*, and updating* $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, \gamma \|\boldsymbol{y} - \mathrm{M}\boldsymbol{x}\|_2^2 + \sum_i \|(P_i(\boldsymbol{x}) + \frac{\boldsymbol{\lambda}_1^k}{\boldsymbol{\mu}_1^k}) \\ -\Phi\boldsymbol{\alpha}_i^{k+1}\|_2^2 + \frac{\boldsymbol{\mu}_2^k}{2}\|(\boldsymbol{x} + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}) - \boldsymbol{u}^{k+1}\|_2^2 \end{aligned} \tag{14}$$

A closed-form solution for $\boldsymbol{x}$ exists as follows:

$$\begin{aligned} \boldsymbol{x}^{k+1} = (\gamma \mathrm{M}^T\mathrm{M} + \boldsymbol{\mu}_1^k \sum_i \mathrm{P}_i{}^T\mathrm{P}_i + \boldsymbol{\mu}_2^k \mathrm{I})^{-1} \\ (\gamma \mathrm{M}^T\boldsymbol{y} + \boldsymbol{\mu}_1^k \sum_i \mathrm{P}_i\Phi\boldsymbol{\alpha}_i^{k+1} + \boldsymbol{\mu}_2^k\boldsymbol{u}^{k+1} - \sum_i \mathrm{P}_i\boldsymbol{\lambda}_1^k - \boldsymbol{\lambda}_2^k \mathrm{I}) \end{aligned} \tag{15}$$

In the expression above, the patch selection operator $P_i(\cdot)$ is rewritten in its matrix form P_i. To calculate $\boldsymbol{x}^{k+1}$ in practice, we can leverage the fact that, for the HSI inpainting task, both $\mathrm{M}^T\mathrm{M}$ and $\mathrm{P}_i{}^T\mathrm{P}_i$ are diagonal matrices. As a results, the matrix inversion of $(\gamma \mathrm{M}^T\mathrm{M} + \boldsymbol{\mu}_1^k \sum_i \mathrm{P}_i{}^T\mathrm{P}_i + \boldsymbol{\mu}_2^k \mathrm{I})^{-1}$ can be simply implemented as an element-wise division. This trick significantly reduces the computational cost of the inversion, making the proposed algorithm more scalable for practical use.

(4) *The Lagrangian and penalty terms are updated as follows*:

$$\begin{aligned} \boldsymbol{\lambda}_1^{k+1} = \boldsymbol{\lambda}_1^k + \boldsymbol{\mu}_1^k(\boldsymbol{x}^{k+1} - \Phi\boldsymbol{\alpha}^{k+1}) \\ \boldsymbol{\lambda}_2^{k+1} = \boldsymbol{\lambda}_2^k + \boldsymbol{\mu}_2^k(\boldsymbol{x}^{k+1} - \boldsymbol{u}^{k+1}) \end{aligned} \tag{16}$$

$$\begin{aligned} \boldsymbol{\mu}_1^{k+1} = \rho_1\boldsymbol{\mu}_1^k \\ \boldsymbol{\mu}_2^{k+1} = \rho_2\boldsymbol{\mu}_2^k \end{aligned} \tag{17}$$

The LRS-PnP-DIP algorithm proposed in [20] is summarized in Algorithm 2.

Algorithm 2 (LRS-PnP-DIP) algorithm.

Require: masking matrix: M, noisy and incomplete HSI: $\boldsymbol{y}$, learned dictionary: Φ. denoiser: $\mathcal{D}$. max iteration: It_{max}. DIP: f_θ
Output: inpainted HSI image $\boldsymbol{x}$.
  Initialization DIP parameters $\theta, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \rho_1, \rho_2$.
  while Not Converged **do**
    for $i = 1 : It_{max}$ do:
      $\boldsymbol{\alpha}^{k+1} = \mathcal{D}(\mathrm{I} - \nabla f)(\boldsymbol{\alpha}^k)$
    update DIP parameters θ, with target: $\boldsymbol{y}$ and intermediate results: $\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}$ as input.
    update $\boldsymbol{x}$ by (15).
    update Lagrangian parameters and penalty terms.
  end while

Remarks. In contrast to its original form, it is here proposed to replace the conventional PnP denoiser in (8) with an averaged NLM denoiser, as introduced in [39]. Additionally, we modify the $f_\theta(\boldsymbol{z})$ in (13) to obtain Lipschitz continuity with Lipschitz constant 1. The averaged NLM denoiser is a variant of NLM denoiser whose weight is designed to be a doubly symmetric matrix, ensuring that both the columns and rows sum to 1. With this doubly symmetric property, the spectral norm of the weight matrix is bounded by 1, meaning the averaged NLM denoiser is by design, non-expansive. (We refer the reader to [39] for detailed structures of the averaged NLM denoiser).

The enhanced algorithm is denoted as LRS-PnP-DIP(1-Lip), which is shown in Algorithm 3, and its flow chart is provided in Figure 1.

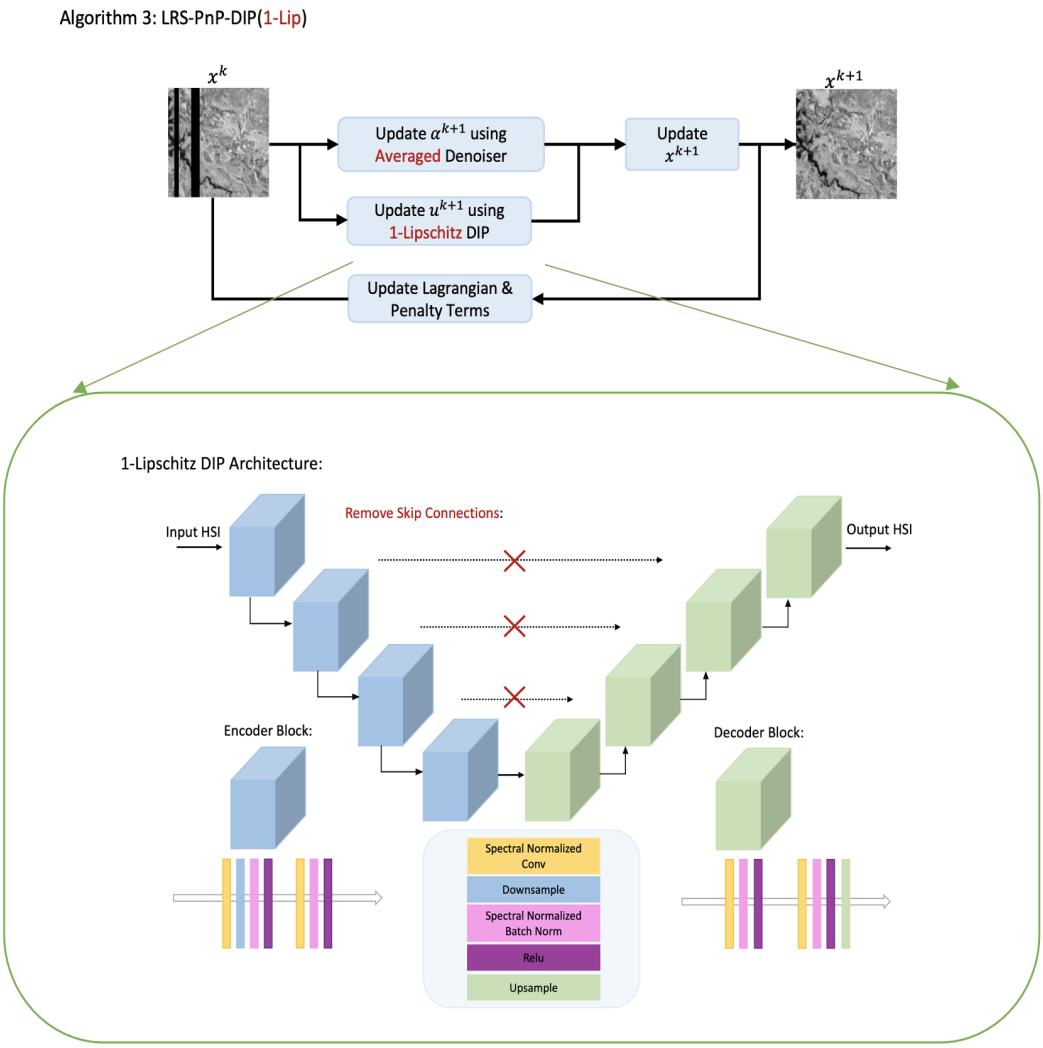

Figure 1. Flow chart of the proposed LRS-PnP-DIP(1-Lip) algorithm. At each iteration, $\boldsymbol{\alpha}^{k+1}$ $\boldsymbol{u}^{k+1}$ and $\boldsymbol{x}^{k+1}$ are sequentially updated. The 1-Lipschitz DIP is implemented by imposing Lipschitz constraints on all layers. We use red color to highlight the differences between this work and the LRS-PnP-DIP algorithm [20]. The detailed design of the 1-Lipschitz DIP is placed in Appendix A.1.

Algorithm 3 LRS-PnP-DIP(1-Lip) algorithm.

Require: masking matrix: M, noisy and incomplete HSI: $\boldsymbol{y}$, learned dictionary: Φ. averaged NLM Denoiser: $\mathcal{D}$. max iteration: It_{max}. 1-Lipschitz DIP: f_θ
Output: inpainted HSI image $\boldsymbol{x}$.
  Initialization DIP parameters $\theta, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \rho_1, \rho_2$.
  while Not Converged **do**
    for $i = 1 : It_{max}$ do:
      $\boldsymbol{\alpha}^{k+1} = \mathcal{D}(\mathrm{I} - \nabla f)(\boldsymbol{\alpha}^k)$
    update DIP parameters θ, with target: $\boldsymbol{y}$ and intermediate results: $\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}_2^k}$ as input.
    update $\boldsymbol{x}$ by (15).
    update Lagrangian parameters and penalty terms.
  end while

3. Convergence Analysis

In this section, we describe the fixed-point convergence of the proposed LRS-PnP-DIP(1-Lip) algorithm under mild assumptions. Fixed-point convergence refers to the type

of convergence wherein the algorithm asymptotically enters a steady state. To establish our main theorem, we need the following results:

Definition 1 (Non-expansive operator)**.** *An operator, $T : \mathbb{R}^n \to \mathbb{R}^n$, is said to be non-expansive if, for any $x, y \in \mathbb{R}^n$, the following applies:*

$$\|(T(x) - T(y))\|^2 \leq \|(x - y)\|^2 \tag{18}$$

Definition 2 (θ-averaged)**.** *An operator, $T : \mathbb{R}^n \to \mathbb{R}^n$, is said to be θ-averaged with some $\theta \in (0, 1)$ if there exists a non-expansive operator, R, such that we can write $T = (1 - \theta)I + \theta R$.*

Lemma 1. *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be θ-averaged for some $\theta \in (0, 1)$. Then, for any $x, y \in \mathbb{R}^n$, the following applies:*

$$\begin{aligned} \|(T(x) - T(y))\|^2 \leq \|(x - y)\|^2 \\ - \frac{1-\theta}{\theta}\|((I - T)(x) - (I - T)(y)\|^2 \end{aligned} \tag{19}$$

Proof of this lemma can be found in [[40], Lemma 6.1].

Definition 3 (Fixed point)**.** *We say that $x^* \in \mathbb{R}^n$ is a fixed point of the operator $T : \mathbb{R}^n \to \mathbb{R}^n$ if $T(x^*) = x^*$. We denote the set of fixed points as fix(T).*

Definition 4 (β-smoothed)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be differentiable; we say that f is β-smooth if there exists $\beta > 0$, such that $\|\nabla f(x) - \nabla f(y)\| \leq \beta\|x - y\|$ for any $x, y \in \mathbb{R}^n$.*

Definition 5 (Strong convexity)**.** *A differentiable function, f, is said to be strongly convex with modulus $\rho > 0$ if $f(x) - \frac{\rho}{2}\|x\|^2$ is convex.*

Lemma 2 (Property of strong convexity)**.** *Let f be strongly convex with modulus $\rho > 0$. Then, for any $x, y \in \mathbb{R}^n$, the following applies:*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \rho\|(x - y)\|^2 \tag{20}$$

Proof of this Lemma can be found in [41].

We made the following assumptions:

Assumption 1. *We assume that (1) the denoiser $\mathcal{D}$ used in the sparse coding step is linear and θ-averaged for some $\theta \in (0, 1)$, (2) the function $f(\boldsymbol{\alpha}) = \frac{\mu_1^k}{2}\|(\boldsymbol{x}^k + \frac{\lambda_1^k}{\mu_1^k}) - \Phi\boldsymbol{\alpha}\|_2^2$ is β-smoothed, and (3) $fix(\mathcal{D}(\mathrm{I} - \nabla f)) \neq \emptyset$.*

Remarks. The θ-averaged property is a subset of the non-expansive operator that most of the existing PnP frameworks have worked with [39,42]. In fact, the non-expansive assumption is found to be easily violated for denoisers such as BM3D and NLM denoisers when used in practice. Nevertheless, the convergence of PnP methods using such denoisers can still be empirically verified. In the experiments, we adopted the modified NLM denoiser whose spectral norm is bounded by 1. Hence, it satisfies the θ-averaged property by design [39].

Lemma 3 (Fixed-point convergence of PnP sparse coding)**.** *If Assumption 1 holds, then, for any $\boldsymbol{\alpha}$, and $\rho > \beta/2$, the sequence $(\boldsymbol{\alpha})^k{}_{k\geq 0}$ generated via step (8) converges to some $\boldsymbol{\alpha}^* \in fix(\mathcal{D}(I - \nabla f))$.*
Proof of this Lemma can be found in [[40], Theorem 3.5].

Assumption 2. *We assume that the DIP function f_θ is L-Lipschitz-bounded:*

$$\|(f_\theta(x) - f_\theta(y))\|^2 \leq L\|(x - y)\|^2 \tag{21}$$

for any x, y, and $L \leq 1$.

Remarks. In the above assumption, the smallest constant L that makes the inequality hold is called the Lipschitz constant. The Lipschitz constant is expressed as the maximum ratio between the absolute change in the output and the input. It quantifies how much the function output changes with respect to the input perturbations or, roughly speaking, how robust the function f_θ is. In deep learning, it. Assumption 2 guarantees that the trained DIP has a Lipschitz constant, $L \leq 1$, which is a desirable feature so that a network does not vary drastically in response to minuscule changes in its input. In the experiments, this can be achieved by constraining the spectral norm of each layer of the neural network during training.

We are now ready to state our main theorem.

Theorem 1 (Convergence of LRS-PnP-DIP(1-Lip) in the Lyapunov Sense)**.** *If both Assumptions 1 and 2 hold, with penalty $\boldsymbol{\mu}$, and with an L-Lipschitz constrained DIP ($L \leq 1$), then there exists a non-increasing function, $H^k = 2\|\boldsymbol{x}^k - \boldsymbol{x}^*\|^2 + \frac{1}{\mu^2}\|\boldsymbol{\lambda}_1^k - \boldsymbol{\lambda}_1^*\|^2 + \frac{1}{\mu^2}\|\boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^*\|^2$, such that all trajectories generated via LRS-PnP-DIP(1-Lip) are bounded, and that as $k \to \infty$, $\|\boldsymbol{x}^k - \boldsymbol{x}^*\|^2 \to 0$, $\|\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*\|^2 \to 0$, and $\|\boldsymbol{u}^k - \boldsymbol{u}^*\|^2 \to 0$; i.e., even if the equilibrium states of $\boldsymbol{x}$, $\boldsymbol{\alpha}$, and $\boldsymbol{u}$ are perturbed, they will finally converge to $\boldsymbol{x}^*$, $\boldsymbol{\alpha}^*$, and $\boldsymbol{u}^*$, respectively. The proposed LRS-PnP-DIP(1-Lip) algorithm is, thus, asymptotically stable.*

We placed the detailed proof of Theorem 1 in Appendix A.2.

4. Experimental Results

4.1. Implementation Details

We evaluate the proposed inpainting model on two publicly available hyperspectral datasets:

- The Chikusei airborne hyperspectral dataset [43] (the link to the Chikusei dataset can be found at https://naotoyokoya.com/Download.html, accessed on 4 January 2025); the test HS image consists of 192 channels, and it was cropped to 36 × 36 pixels in size.
- The Indian Pines dataset from AVIRIS sensor [44]; the test HS image consisted of 200 spectral bands, and it was cropped to 36 × 36 pixels size.

For each dataset, we trained the dictionary Φ with a size of 1296×2000, using only noisy and incomplete HSI images (the size of the test HS image and the dictionary were chosen to match the practical use case where the proposed inpainting algorithms were to be mounted on each nano-satellite to process different small image tiles in parallel [45]), this is a necessary step if we do not have access to pure spectra. If a standard spectral dataset exists, learning can be performed in the form of a pre-training step. However, we used the input HS image for this task to demonstrate that the proposed algorithms operate in a self-supervised setting. Additionally, we introduced Gaussian noise with noise level $\sigma = 0.12$ to the cropped HS images. Since the main focus of this paper is image inpainting, instead of denoising, σ was kept fixed across all experiments. We applied mask M to all the spectral bands in the given region, which represents the most challenging case. For the choice of the PnP denoiser, we used both the BM3D and modified NLM denoisers [39]; the latter is an averaged denoiser, which enjoys a non-expansive property that is crucial to convergence analysis.

Our implementation of the deep image prior model follows the same structure as in the deep hyperspectral prior [11]. During the training of 1-Lipschitz DIP, we applied Lipschitz regularization to all layers, as in [46]. In each iteration, we fed the DIP with the latent image from the previous iteration, rather than the random signal, as suggested by [11]. We did not perturb the input to support our convergence analysis, which is different from the canonical DIP. To eliminate the need for manually selecting the total number of iterations for the DIP model, we adopted the early stopping criterion proposed in [47] with a fixed window size of 20 and a patience number of 100, which detects the near-peak MPSNR point using windowed moving variance (WMV). Note that, for the training of conventional DIP, the early-stopping strategy is often employed to prevent it from fitting to random noise after certain iterations. However, the proposed LRS-PnP-DIP(1-Lip) algorithm mitigates the over-fitting issue due to the imposed constraint. The early-stopping criterion is applied primarily to optimize the algorithm's runtime so that one does not have to wait for the algorithm to reach the maximum number of iterations. In our formulation, the low-rank and sparsity constraints in the equation were weighted with parameters w_{lr} and w_s, respectively, while the data fidelity term was weighted with γ. Initially, we set w_s/w_{lr} to 1 and γ to 0.5. Notably, the choice of γ is highly dependent on the noise level of the observed image. If the noise level is low, the recovered image, X, should be similar to the noisy observation, Y, parameter γ should be large, and vice versa. The Adam optimizer was used, and the learning rate was set to 0.1. Finally, we used three widely used metrics, namely the mean signal-to-noise ratio (MPSNR), mean structural similarity (MSSIM), and mean spectral angle mapper (MSAM), to evaluate the performance of the model in all experiments.

4.2. Low-Rank vs. Sparsity

In this section, we examine the effectiveness of both sparsity and low-rank constraints in our proposed LRS-PnP algorithm. Figure 2 shows the inpainting performance of LRS-PnP over different τ for different masks (in percentages), where τ is defined as the ratio of the weight of the sparsity constraint and the low-rank constraint. For example, $\tau = 0$ means that the sparsity constraint is disabled, and mask = 50% means that 50% of the pixels are missing. We can conclude from the results that (1) the LRS-PnP algorithm works better with both low-rank and sparsity constraints, as using either an overwhelming sparsity constraint or a low-rank constraint alone tends to reduce inpainting performance, and (2) when the percentage of missing pixels increases, more weight should be placed on the sparsity constraint in order to achieve high-quality reconstruction. On the one hand, approximation errors may be introduced in the sparse coding step; i.e., step (5) in Algorithms 2 and 3. These errors can potentially be suppressed by the constraint of low rank. On the other hand, the sparsity constraint exploits the spectral correlations; i.e., the performance will be enhanced if spatial information is considered.

4.3. Low Rankness Due to DIP

In the original LRS-PnP-DIP Algorithm [20], we propose solving the low-rank minimization problem using DIP instead of the traditional SVT projection. A natural question arises: does the DIP (e.g., the architecture proposed in Figure 1) effectively capture the redundancies in the spectral domain? This is empirically justified in Figure 3, where we trained DIP on the corrupted HS image with varying numbers of spectral bands as input and output. The results show that the inpainting performance of DIP improved substantially when more spectral bands were utilized during training; the peak MPSNR was obtained when the neural network saw the entire 128 bands of the input HS images, rather than each single band individually. This finding is counterintuitive since DIP with 2D convolution is generally expected to handle only the spatial correlation of input

HSIs. Similar observations have been reported in [10,48], where the authors confirmed that DIP with 2D convolution implicitly explored the channel correlations, possibly due to the upsampling layers in the decoder part that manipulates the HSI channels in the spectral domain.

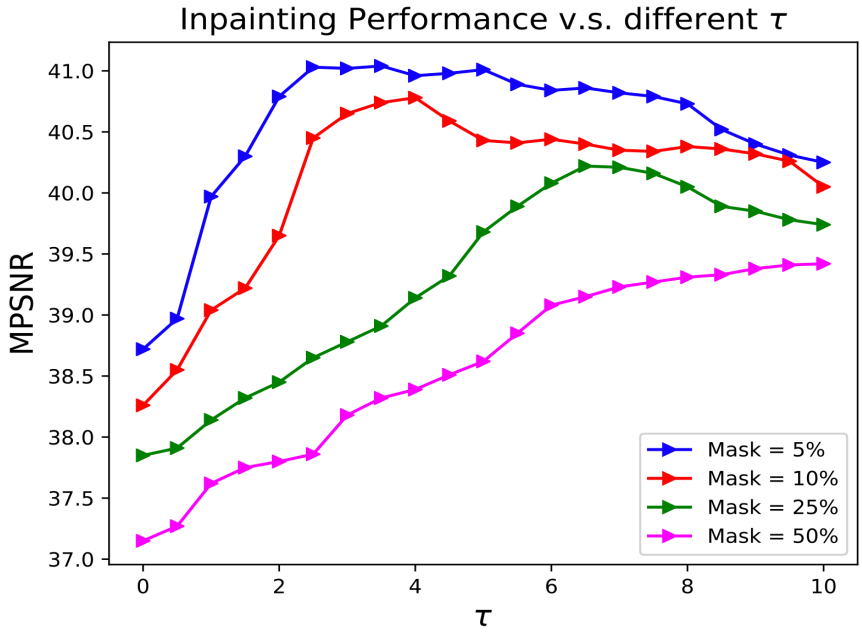

Figure 2. Comparison of MPSNR value of LRS-PnP among different τ ($\tau = w_s/w_{lr}$) under different masks.

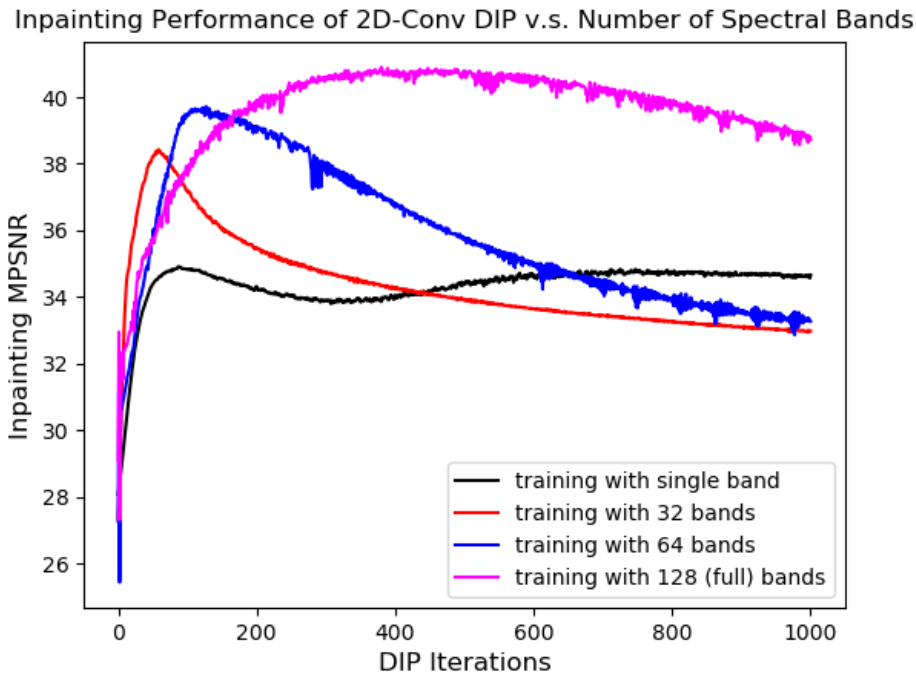

Figure 3. Learning capability of DIP vs. number of input and output channels. Training is conducted with a single-band HS image, meaning that the input HSIs are processed; there is no correlation in the spectral domain. There is a significant performance gain when there are more input bands, indicating that the DIP with 2D convolution has the ability to exploit the correlation between channels.

In Figure 4, we sort the singular values of the inpainted image of LRS-PnP (the pink line), LRS-PnP-DIP [20] (the blue line), and PnP sparse coding without a low-rank constraint (green line), in descending order. Among these algorithms, LRS-PnP performs

singular value projection to promote low rankness, while it is replaced with DIP in LRS-PnP-DIP. It is observed that using a well-trained DIP alone was sufficient to capture these low-rank details. Thus, we suggest replacing the conventional SVT process with DIP. Although there is no theoretical evidence on how DIP mimics the SVT, we show through experiments that the DIP can well preserve some small yet important singular values that would otherwise be discarded via SVT projection. Hence, it has the ability to better exploit the low-rank subspace in a data-driven fashion, possibly owing to its highly non-linear network structures and inductive bias.

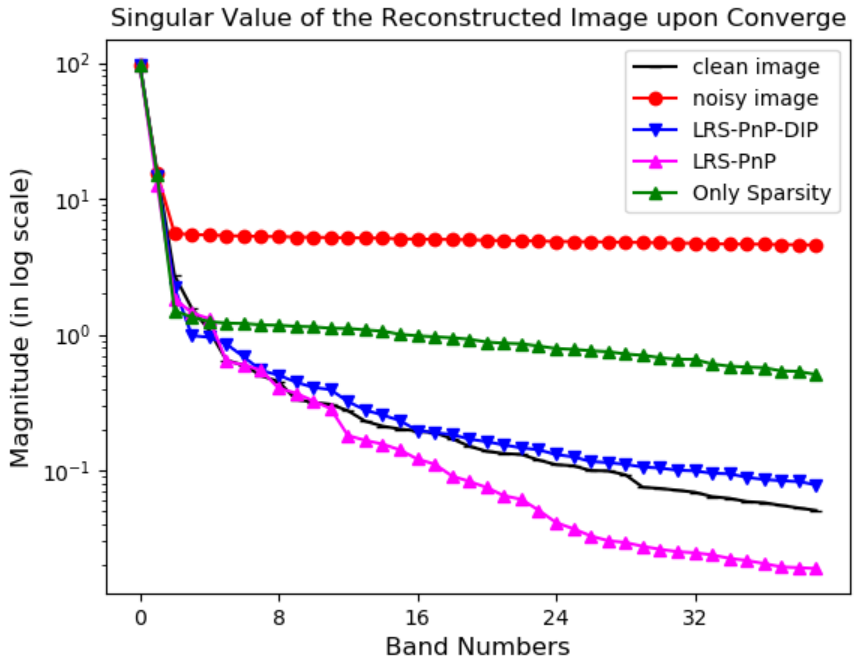

Figure 4. The amplitude of the singular value of the reconstructed image upon converge. The important singular values are captured and preserved via the 2D-convolution DIP, which is even more accurate than the traditional SVT projection.

4.4. Convergence

The use of the PnP denoiser and DIP in the context of HSIs has been recently investigated in [12,13]. However, theoretical evidence of the stability of the algorithm is still lacking. To bridge this gap, we verify the convergence of LRS-PnP-DIP(1-Lip). In Figure 5, the top left, top right, and bottom left figures represent the successive differences of each state on a log scale, and the bottom right figure represents the inpainting performance. For LRS-PnP-DIP, we used the BM3D denoiser and DIP without constraints. For LRS-PnP-DIP(1-Lip), we used the modified NLM denoiser and the 1-Lipschitz constraint DIP. The latter automatically satisfies Assumptions 1 and 2; thus, the fixed-point convergence can be guaranteed. Interestingly, we noticed that LRS-PnP-DIP with BM3D and conventional DIP still works pretty well in practice even though there are fluctuations in state $\boldsymbol{x}$, as can be seen in the top right and bottom left plots in Figure 5. We deduced that such instability mainly comes from the process of solving DIP sub-problems, rather than the denoising sub-problems, as the primal variable $\boldsymbol{\lambda}_1$ smoothly converges with BM3D denoisers as well.

From the bottom right plot, we observed that the best results are obtained with LRS-PnP-DIP in terms of reconstruction quality, and the LRS-PnP-DIP(1-Lip) is only slightly lower than its unconstrained counterpart. Our observations on the reduced performance of Lipschitz DIP are in agreement with existing works [46,49], where imposing the Lipschitz constraint during training does hurt the capacity and expressivity of the neural network.

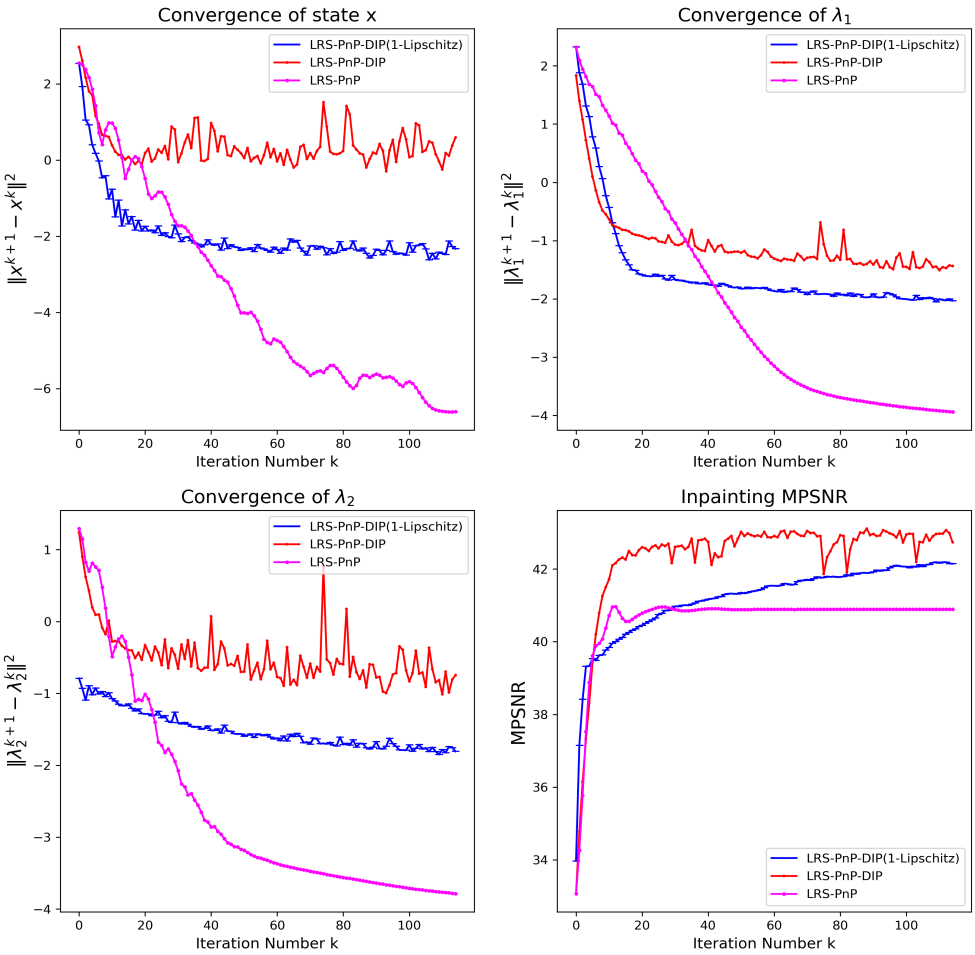

Figure 5. Empirical converge of LRS-PnP-DIP(1-Lip) with modified NLM denoiser and non-expansive/1-Lipschitz DIP. Top left, top right, and bottom left: successive difference of $\boldsymbol{x}$, $\boldsymbol{\lambda}_1$, and $\boldsymbol{\lambda}_2$ in the log scale, respectively. Bottom right: the inpainting MPSNR vs. the number of iterations.

4.5. Comparison with State of the Art

The proposed algorithms were compared with learning-based algorithms: DHP [11], GLON [4], R-DLRHyIn [16], DeepRED [22], PnP-DIP [24], DeepHyIn [50], and DDS2M [19]. To ensure the fairness of comparison, we employed the same U-net as the backbone for DHP, DeepRED, R-DLRHyIn, PnP-DIP, and LRS-PnP-DIP(1-Lip) and the attention-based U-net with the same complexity for DeepHyIn and DDS2M. For GLON, we replaced its original FFDNet with the one trained on the hyperspectral dataset [51] and kept the others intact. (The reason for making this adaption is that the neural network proposed in GlON [4] was trained on RGB/grayscale images. It was found to be beneficial to use an FFDNet, e.g., in [51], trained on the HSI dataset.) For DeepHyIn, we used the end members extracted from the clean image patch. For DDS2M, we fine-tuned the diffusion-related parameters, as suggested in [19]. The statistical results of each test sample are averaged over 20 experiments to account for the effect of random seeds during training.

In Figure 6, we plot the recovered spectrum of the center of the missing area of LRS-PnP-DIP(1-Lip) for each method. It can be seen that LRS-PnP-DIP(1-Lip) produces a more consistent and realistic spectrum in the missing region compared to methods such as GLON, DHP, DeepRED, R-DLRHyIn, DDS2M, and DeepHyIn. We present the visual inpainting results for each method on the Chikusei dataset and the Indian Pines dataset, as depicted in Figures 7 and 8, respectively. The quantitative statistical results for each dataset, evaluated under various mask shapes, are summarized in Table 1 for the Chikusei dataset and Table 2 for the Indian Pines dataset.

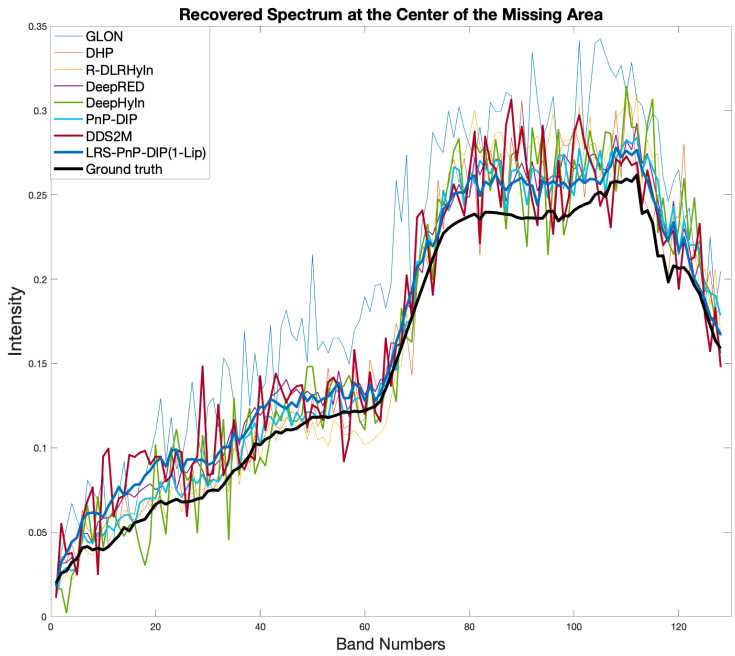

Figure 6. Different algorithms and their recovered spectrum of the center pixel with the assumption that the whole spectrum bands are missing. MPSNR increases from the thin line to the thick line.

It is shown that LRS-PnP-DIP(1-Lip) can effectively suppress noise compared to methods such as R-DLRHyIn, DeepHyIn, and DDS2M. Moreover, it alleviates the over-smoothing phenomenon of the DeepRED and PnP-DIP algorithms by preserving more details in the non-missing region; this is clear for the last two test samples in Figure 7 and for the first four test samples in Figure 8. From Tables 1 and 2, it can be seen that LRS-PnP-DIP(1-Lip) almost always demonstrates the highest inpainting performance on both the Chikusei and Indian Pines datasets compared to other methods. The second best is established via DDS2M, which is the closest method to ours in terms of inpainting quality. This is possibly due to its ability to leverage inductive bias from hierarchical diffusion structures. However, as is evident in Figures 7 and 8, DDS2M and other DIP-based extensions occasionally exhibit limitations, such as producing noisy backgrounds or failing to effectively inpaint missing strips. It is important to note that these methods utilize conventional DIP as backbones; hence, they are susceptible to the instability issue of DIP where the algorithm may eventually fit into pure noise without proper constraints. In contrast, LRS-PnP-DIP(1-Lip) is superior, as it not only harvests the strong learning capability of DHP but also enjoys a stability guarantee. For the HSI inpainting task, we anticipate that some information, from the background/non-missing pixels to be carried to the deeper layers, there is a performance loss when removing the skip connections between the encoder and the decoder. Nevertheless, LRS-PnP-DIP(1-Lip) mutes these layers to trade performance for convergence, and it is empirically found that, even if LRS-PnP-DIP(1-Lip) is highly restricted due to layer-wise spectral normalization, it can preserve more textures and produce less noisy results than DDS2M. This observation leads to the conclusion that the proposed LRS-PnP-DIP(1-Lip) algorithm would be preferable for solving practical inpainting tasks compared to existing solutions. It has the ability to yield faithful reconstructions while ensuring stability and robustness.

Table 1. Comparison between the proposed algorithm and other learning-based inpainting algorithms on the Chikusei dataset: the mean and variance over 20 samples are shown here. During the experiments, four different mask shapes were used, which are depicted in the first four columns of Figure 7. The best results are underlined for each type of mask. For the metrics MPSNR and MSSIM, higher values indicate better performance. Conversely, for the MSAM metric, the lower the better.

Mask Type	**Methods**	**Input**	**GLON** [4]	**DHP** [11]	**R-DLRHyIn** [16]	**DeepRED** [22]	**DeepHyIn** [50]	**PnP-DIP** [24]	**DDS2M** [19]	**LRS-PnP-DIP(1-Lip)**
Mask Type 1	MPSNR ↑	33.0740	41.0324	41.3956(±0.52)	41.5714(±0.31)	41.5765(±0.28)	41.6902(±0.57)	41.7595(±0.25)	41.7728(±0.35)	41.8023(±0.16)
	MSSIM ↑	0.2441	0.9078	0.9102(±0.03)	0.9135(±0.01)	0.9121(±0.02)	0.9180(±0.02)	0.9250(±0.02)	0.9268(±0.02)	0.9275(±0.01)
	MSAM ↓	0.7341	0.1342	0.1296(±0.01)	0.1250(±0.01)	0.1237(±0.01)	0.1221(±0.01)	0.1203(±0.01)	0.1183(±0.01)	0.1185(±0.01)
Mask Type 2	MPSNR ↑	31.7151	38.4103	39.7510(±0.65)	40.1586(±0.84)	40.4119(±0.33)	40.6260(±0.40)	40.6661(±0.52)	40.7904(±0.74)	41.4910(±0.23)
	MSSIM ↑	0.2338	0.8768	0.8961(±0.01)	0.9041(±0.01)	0.9049(±0.01)	0.9100(±0.01)	0.9121(±0.01)	0.9130(±0.01)	0.9189(±0.01)
	MSAM ↓	0.8296	0.1518	0.1436(±0.02)	0.1373(±0.01)	0.1320(±0.01)	0.1315(±0.01)	0.1298(±0.01)	0.1264(±0.01)	0.1203(±0.01)
Mask Type 3	MPSNR ↑	30.3258	35.6927	38.1440(±0.90)	38.9580(±0.79)	39.6952(±0.99)	39.8608(±0.68)	39.8150(±0.40)	40.0779(±0.82)	40.8400(±0.49)
	MSSIM ↑	0.2018	0.8558	0.8805(±0.01)	0.8871(±0.01)	0.8902(±0.01)	08950(±0.01)	0.8958(±0.01)	0.9003(±0.01)	0.9088(±0.01)
	MSAM ↓	0.8441	0.1670	0.1563(±0.01)	0.1507(±0.01)	0.1495(±0.01)	0.1480(±0.01)	0.1476(±0.01)	0.1455(±0.01)	0.1392(±0.01)
Mask Type 4	MPSNR ↑	27.9802	34.5011	36.6018(±1.15)	37.6504(±0.89)	37.9080(±0.59)	38.1961(±0.77)	38.3442(±0.73)	38.7529(±0.88)	39.4068(±0.54)
	MSSIM ↑	0.1700	0.8267	0.8693(±0.01)	0.8740(±0.01)	0.8772(±0.01)	0.8769(±0.01)	0.8820(±0.01)	0.8849(±0.01)	0.8927(±0.01)
	MSAM ↓	0.8847	0.1803	0.1714(±0.03)	0.1636(±0.02)	0.1602(±0.01)	0.1580(±0.01)	0.1557(±0.01)	0.1532(±0.01)	0.1449(±0.01)

Table 2. Comparison between the proposed algorithm and other learning-based inpainting algorithms on the Indian Pines dataset: The mean and variance over 20 samples are shown here. During the experiments, four different mask shapes were used, which are depicted in the first four columns of Figure 7. The best results are underlined for each type of mask.

Mask Type	**Methods**	**Input**	**GLON** [4]	**DHP** [11]	**R-DLRHyIn** [16]	**DeepRED** [22]	**DeepHyIn** [50]	**PnP-DIP** [24]	**DDS2M** [19]	**LRS-PnP-DIP(1-Lip)**
Mask Type 1	MPSNR ↑	31.6903	39.7676	40.1917(±0.33)	40.5490(±0.41)	40.7982(±0.32)	40.9309(±0.60)	40.8897(±0.34)	41.1084(±0.29)	41.3423(±0.24)
	MSSIM ↑	0.2268	0.8884	0.8917(±0.02)	0.8930(±0.02)	0.8932(±0.01)	0.9028(±0.01)	0.9050(±0.01)	0.9092(±0.01)	0.9139(±0.01)
	MSAM ↓	0.7723	0.1594	0.1580(±0.01)	0.1369(±0.01)	0.1368(±0.01)	0.1320(±0.01)	0.1255(±0.01)	0.1240(±0.01)	0.1256(±0.01)
Mask Type 2	MPSNR ↑	29.6815	36.9092	38.1331(±0.22)	38.7072(±0.25)	38.5630(±0.19)	38.7976(±0.40)	39.0021(±0.32)	39.9861(±0.40)	40.3010(±0.23)
	MSSIM ↑	0.2098	0.8457	0.8849(±0.01)	0.8902(±0.01)	0.8864(±0.01)	0.8919(±0.01)	0.8972(±0.01)	0.9011(±0.01)	0.9048(±0.01)
	MSAM ↓	0.8723	0.1514	0.1509(±0.01)	0.1469(±0.01)	0.1450(±0.01)	0.1444(±0.01)	0.1420(±0.01)	0.1380(±0.01)	0.1310(±0.01)
Mask Type 3	MPSNR ↑	28.6195	36.7447	37.0714(±0.28)	37.4790(±0.20)	37.3976(±0.39)	37.4555(±0.39)	37.7821(±0.22)	37.8070(±0.52)	38.0599(±0.33)
	MSSIM ↑	0.1864	0.8395	0.8787(±0.01)	0.8870(±0.01)	0.8869(±0.01)	0.8893(±0.01)	0.8921(±0.01)	0.8945(±0.01)	0.8982(±0.01)
	MSAM ↓	0.8823	0.1584	0.1579(±0.01)	0.1539(±0.01)	0.1521(±0.01)	0.1490(±0.01)	0.1488(±0.01)	0.1470(±0.01)	0.1458(±0.01)
Mask Type 4	MPSNR ↑	24.6998	34.5630	35.8009(±0.30)	35.8292(±0.22)	35.8145(±0.32)	35.8905(±0.23)	36.1022(±0.18)	36.4938(±0.34)	36.5104(±0.23)
	MSSIM ↑	0.1672	0.8342	0.8592(±0.01)	0.8651(±0.01)	0.8678(±0.01)	0.8698(±0.01)	0.8733(±0.01)	0.8764(±0.01)	0.8779(±0.01)
	MSAM ↓	0.8977	0.1644	0.1610(±0.02)	0.1611(±0.01)	0.1602(±0.01)	0.1588(±0.01)	0.1580(±0.01)	0.1552(±0.01)	0.1554(±0.01)

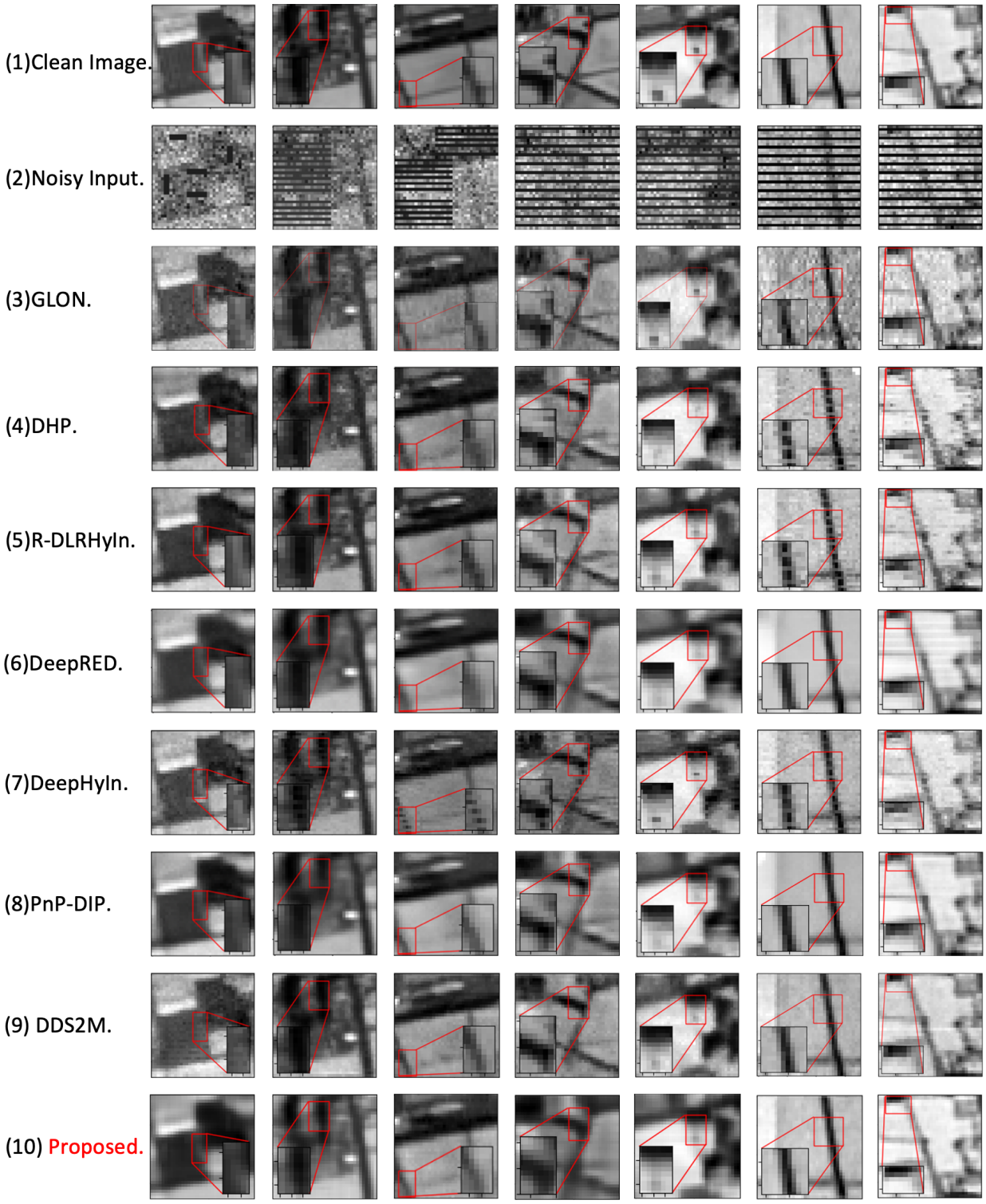

Figure 7. Comparison between the proposed algorithm and other learning-based inpainting algorithms on the Chikusei dataset. From top to bottom: (**1**) clean image, (**2**) input image, (**3**) GLON, (**4**) DHP, (**5**) R-DLRHyIn, (**6**) DeepRED, (**7**) DeepHyIn, (**8**) PnP-DIP, (**9**) DDS2M, and (**10**) LRS-PnP-DIP(1-Lip). All images are visualized at band 80.

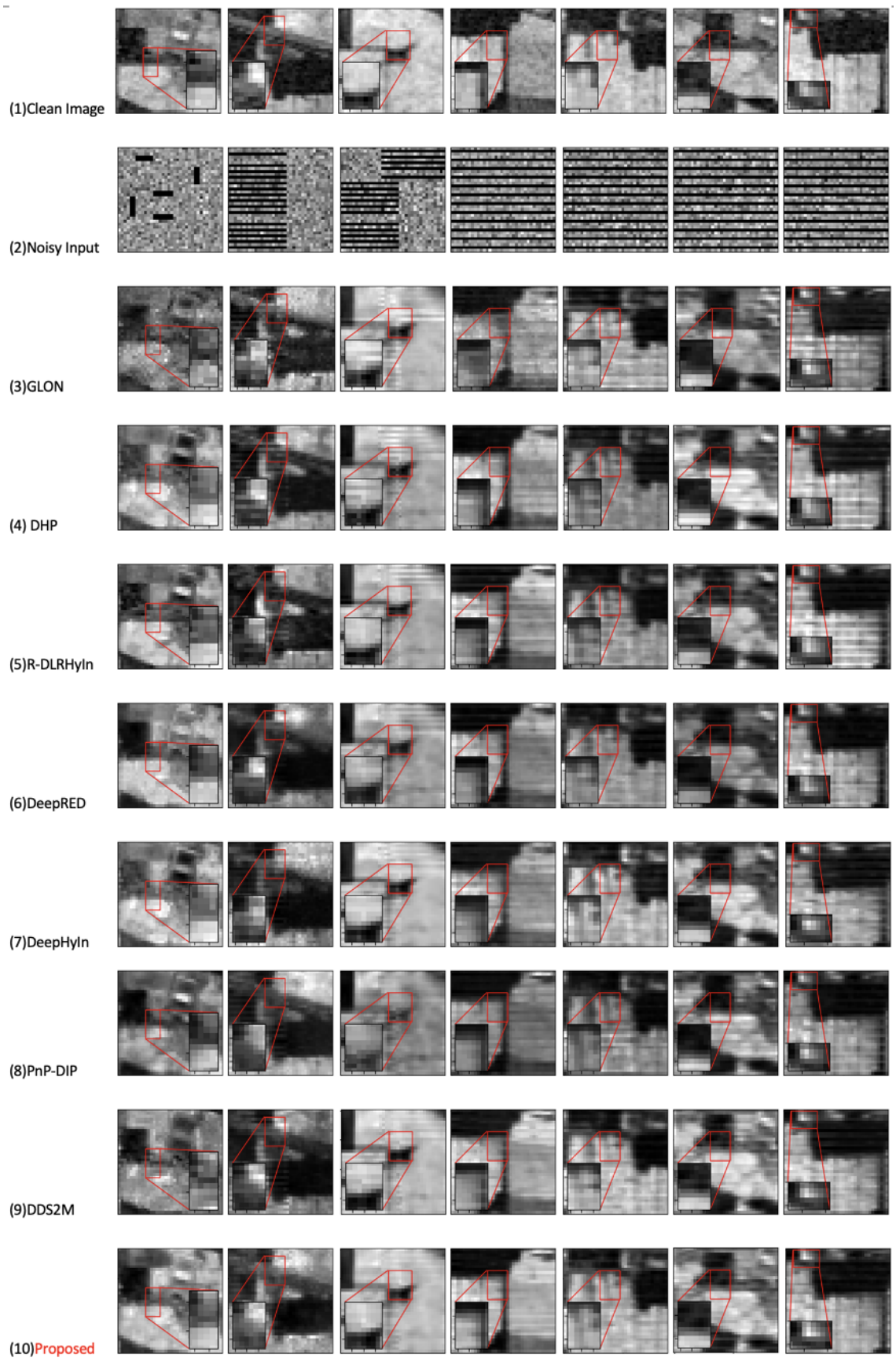

Figure 8. Comparison between the proposed algorithm and other learning-based inpainting algorithms on the Indian Pines dataset. From top to bottom: (**1**) clean image, (**2**) input image, (**3**) GLON, (**4**) DHP, (**5**) R-DLRHyIn, (**6**) DeepRED, (**7**) DeepHyIn, (**8**) PnP-DIP, (**9**) DDS2M, and (**10**) LRS-PnP-DIP(1-Lip). All images are visualized at band 150.

5. Conclusions

The LRS-PnP algorithm, along with its variant LRS-PnP-DIP proposed in the previous work [20], are innovative hyperspectral inpainting methods designed to address the challenge of missing pixels in noisy and incomplete HS images. These techniques can better handle the challenging situations in practical HSI acquisition systems where the entire spectral band may be absent. Through experiments, we show that both sparsity and the low-rank priors can be effectively integrated as constraints during the training of DIP; the resulting LRS-PnP-DIP algorithm leverages the strong learning capability of the powerful generative DIP model and is able to exploit spectral and spatial redundancies inherent to HSIs. Because the proposed methods do not require any training data, they are particularly attractive for practical hyperspectral inpainting tasks, where only the observed corrupted image is available for reference. However, the LRS-PnP-DIP algorithm suffers from instability issues and may occasionally diverge due to the DIP. In this paper, this long-lasting issue was effectively addressed through the proposed LRS-PnP-DIP(1-Lip) algorithm. Experimental results on real datasets demonstrate that the LRS-PnP-DIP(1-Lip) algorithm not only resolves the instability issue but also delivers results competitive with the state-of-the-art HSI inpainting algorithms. Unlike previous works where the convergence of the algorithm is either missing or only empirically verified, such as [12–14,24,52], this work has provided theoretical evidence and established convergence guarantees under non-expansiveness and Lipschitz continuity assumptions on the PnP denoiser and the DIP neural network. However, the performance of DIP has been observed to be sensitive to the network structure [11] and to the training parameters, such as the number of iterations and the learning rate. When the proposed LRS-PnP-DIP(1-Lip) algorithm is given an insufficient number of iterations to learn, DIPs may not work as expected. In future work, we would like to explore the inductive bias of DHPs, as well as a broader variety of deep learning models, such as transformers (the transformer has recently been explored in the remote sensing community [53]) to exploit their potential in solving HSI inpainting problems. Additionally, the extension of the proposed methods to other HSI processing tasks, such as self-supervised HSI denoising, unmixing, and deblurring, is left to a future investigation.

Author Contributions: Conceptualization, S.L. and M.Y.; methodology, S.L. and M.Y.; software, S.L.; validation, S.L. and M.Y.; formal analysis, S.L. and M.Y.; investigation, S.L. and M.Y.; resources, S.L.; data curation, S.L.; writing—original draft preparation, S.L; writing—review and editing, S.L. and M.Y.; visualization, S.L. and M.Y.; supervision, M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The code for reproducing results is available at https://github.com/shuoli0708/LRS-PnP-DIP, accessed on 14 January 2025. The data presented in this study are available upon request from the corresponding author.

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

Appendix A

Appendix A.1. On the Design of 1-Lipschitz DIP

In the proposed LRS-PnP-DIP(1-Lip) algorithm, we redesigned the DIP to have a Lipschitz constant of 1. To do so, we started by taking a deeper look at how the Lipschitz constant is calculated for each layer; then, we showed that the Lipschitz constraint can be enforced to construct a 1-Lipschitz DIP.

Appendix A.1.1. Convolutional Layer

The convolutional layer $conv(f)$ works by performing the convolutional operation on the input to extract the features. The output of the convolution operation can be expressed as follows:

$$conv(f) = \mathrm{W} \cdot f \tag{A1}$$

where W is a double-block circular matrix. Substituting (A1) into the definition of the Lipschitz constant, we have the following:

$$\begin{aligned} &\|(\mathrm{W}\boldsymbol{x}_1 - \mathrm{W}\boldsymbol{x}_2)\|^2 \leq L\|(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|^2 \\ &\Leftrightarrow \|\mathrm{W}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|^2 \leq L\|(\boldsymbol{x}_1 - \boldsymbol{x}_2)\|^2 \\ &\Leftrightarrow \|\mathrm{W}x\|^2 \leq L\|(x)\|^2 \quad (\text{Let } \boldsymbol{x} = \boldsymbol{x}_1 - \boldsymbol{x}_2, \boldsymbol{x}_1 \neq \boldsymbol{x}_2) \end{aligned} \tag{A2}$$

It follows that

$$\begin{aligned} &\frac{\|\mathrm{W}(x)\|^2}{\|x\|^2} \leq L \\ &\text{or equivalently,} \quad L_{smallest} = \sup_{x \neq 0} \frac{\|\mathrm{W}x\|^2}{\|x\|^2} = \|\mathrm{W}\|^2 \end{aligned} \tag{A3}$$

The above expression reveals the fact that the spectral norm of W is bounded by the Lipschitz constant L. Hence, regularizing the convolutional layer $conv(f)$ to have Lipschitz continuity boils down to constraining the matrix W; e.g., in our case, W is expected to have all of its singular values be at most 1, meaning that $conv(f)$ is 1-Lipschitz. To design such a matrix, W, we adopted the power iteration method, as suggested in [46]. The power iteration method works iteratively and approximates the largest singular value of W. Given a sufficient number of iterations, it converges to the true spectral norm of W with almost negligible errors. In practice, this can be achieved with only a few iterations, which is quite efficient compared to the SVD method.

Appendix A.1.2. Skip Connection Layer

The DIP has skip connections between the encoder and the decoder. These skip/residual connections ensure anuninterrupted gradient flow, which can effectively tackle the vanishing gradient problem. Moreover, it allows some previous features to be reused later in the network. In particular, for our HSI inpainting problem, we expect some information about the clean pixels from the background to be carried to the next layers.
The skip connections take the following form:

$$skip(f) = f + \mathrm{W}(f) \tag{A4}$$

where W stands for the composition of possible different functions or layers (e.g., fully connected layers and activation functions). Since the skip connections are additive operators, the Lipschitz constant is determined by both of its components. Suppose that functions f_1 and f_2, are Lipschitz continuous with the Lipschitz constants L_1 and L_2, respectively. Then, the composition $f^{'} = f_1 + f_2$ satisfies the following:

$$\begin{aligned} \|f^{'}(\boldsymbol{x}_1 - \boldsymbol{x}_2)\| &= \|(f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_1)) - (f_1(\boldsymbol{x}_2) + f_2(\boldsymbol{x}_2))\|^2 \\ &= \|(f_1(\boldsymbol{x}_1) - f_1(\boldsymbol{x}_2)) + (f_2(\boldsymbol{x}_1 - f_2(\boldsymbol{x}_2))\|^2 \end{aligned} \tag{A5}$$

Applying the triangular inequality to the L_2 norm, we have the following:

$$\begin{aligned} &\|(f_1(\boldsymbol{x}_1) - f_1(\boldsymbol{x}_2)) + (f_2(\boldsymbol{x}_1 - f_2(\boldsymbol{x}_2))\|^2 \\ &\quad \leq \|f_1(\boldsymbol{x}_1) - f_1(\boldsymbol{x}_2)\|^2 + \|f_2(\boldsymbol{x}_1) - f_2(\boldsymbol{x}_2)\|^2 \end{aligned} \tag{A6}$$

Furthermore, the right-hand side of inequality (A6) can be bounded by the Lipschitz constant of functions f_1 and f_2 as follows:

$$\begin{aligned} &\|f_1(\boldsymbol{x}_1) - f_1(\boldsymbol{x}_2)\|^2 + \|f_2(\boldsymbol{x}_1) - f_2(\boldsymbol{x}_2)\|^2 \\ &\quad \leq L_1\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 + L_2\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \\ &\quad = (L_1 + L_2)\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \end{aligned} \tag{A7}$$

We can conclude that the Lipschitz constant of $f^{'} = f_1 + f_2$ is bounded by the sum of the Lipschitz constants of f_1 and f_2. Therefore, for the skip connections defined in (A4), the Lipschitz constant is bounded by $1 + L_{\mathrm{W}}$, with a non-negative L_{W}. Because the skip connections would have a Lipschitz constant exceeding 1, we removed them from the implementation of the 1-Lipschitz DIP.

Appendix A.1.3. Pooling Layer

The pooling layer $pool(f)$ works by dividing the input image into a set of sub-image patches and converting them to a single value. We used max-pooling in the encoder to extract feature maps from the input HSIs. Max pooling takes the maximum values of the neurons and discards the rest:

$$pool(f) = max(f_1, f_2, f_3 \ldots . f_n) \tag{A8}$$

where n denotes the number of image patches. The expression above can be seen as an affine transformation with a fixed weight; i.e.,

$$\begin{aligned} &pool(f) = \mathrm{W} \cdot f \\ &\mathrm{W}_i = \begin{cases} 1 & \text{if} \quad f_i = max(f_1, f_2, f_3 \ldots . f_n) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{A9}$$

The gradient of W is 1 for the neurons with the maximum value and 0 otherwise; the latter does not participate in the propagation. Hence, by definition, the max pooling layer is Lipschitz-continuous with a Lipschitz constant of 1.

Appendix A.1.4. Activation Layer

The Lipschitz constant of the activation layer $act(f)$ is trivial, as most activation functions such as ReLu, Leaky-ReLu, Tanh, and Sigmoid have Lipschitz constants of 1.

Appendix A.1.5. Batch Normalization Layer

The batch normalization layer $BN(f)$ works by scaling and normalizing the inputs so as to mitigate issues related to internal covariance shifts. It is defined as follows:

$$BN(f) = \gamma \cdot \frac{f - \mathbb{E}(f)}{\sqrt{Var(f)}} + b \tag{A10}$$

In the above formulation, $\mathbb{E}(f)$ is the mini-batch mean, and $Var(f)$ is the mini-batch variance. f is normalized, scaled by γ, and shifted with respect to the bias b. γ and b are two learned parameters during training, which forces the network to also learn the identity

transformation and the ability to make a decision on how to balance them. From (A10), it is evident that $BN(f)$ is an affine transform with the transformation matrix W being as follows:

$$\mathrm{W} = \mathrm{diag}\left(\frac{\gamma}{\sqrt{Var(f)}}\right) \tag{A11}$$

Therefore, similar to the analysis of the convolutional layers, the Lipschitz constant of the batch normalization layer is the spectral norm of W. However, $BN(f)$ typically normalizes its input by the variance and scales it by the factor γ; this would inevitably destroy the Lipschitz continuity. Hence, we propose a variant to the batch normalization layer as follows:

$$BN^{'}(f) = L \cdot (f - \mathbb{E}(f)) + b \tag{A12}$$

The modified BN operator $BN^{'}$ centers its input by subtracting the mean, which is quite different from its original form, (A10). In the experiments, we observed that such an adaptation did not affect the performance of the DIP.

Appendix A.1.6. Lipschitz Constant of the Full Network

Consider a neural network, f_{net}, which is a composition of n independent sub-networks, $f_1, f_2, \ldots f_n$, each of which has its own Lipschitz constant, $L_1, L_2, \ldots L_n$:

$$f_{net}(x) = (f_1 \circ f_2 \circ \ldots f_n)(x) \tag{A13}$$

Then, the Lipschitz constant of f_{net} can be upper-bounded by the following:

$$\begin{aligned} L_{net} &\leq L_1 \cdot L_2 \cdot \ldots \cdot L_n \\ &= \prod_{i=1}^{n} L_i \end{aligned} \tag{A14}$$

Hence, we can tackle the Lipschitz constant of each layer independently and combine them to construct the upper bound of the entire network. We note that the bound certified in (A14) is not tight and that there are spaces for establishing a tighter bound by considering the network as a whole, which we will leave for future work.

Appendix A.1.7. Enforcing Lipschitz Constraint

Inspired by the work in [46], we introduce an extra projection step after each weight update as follows:

$$P(\mathrm{W}, L) = \frac{1}{\max(1, \|\mathrm{W}\|^2/L)}\mathrm{W} \tag{A15}$$

where L stands for the desired Lipschitz constant, particularly $L = 1$ for 1-Lipschitz DIP. This step aims to project the weight matrix, W, back to the feasible set for those layers that violate the Lipschitz constraint.

Appendix A.2. Proof of Theorem 1

The proof relies on the Lyapunov stability theory, which is the heart of the dynamic system analysis [54]. Lyapunov stability theory can be categorized into (a) the indirect method, which analyzes the convergence through the system state equation, and (b) the direct method, which explicitly describes the behavior of the system's trajectories and its convergence by making use of the Lyapunov function. We refer the reader to [55] for a more detailed definition of the Lyapunov function (specifically, Theorem 1.2 and Theorem 3.3). In some contexts, it is also known as an energy or dissipative function [56]. Compared to the former, the direct method is more appealing, as the convergence can be established by

only showing the existence of such a function. In this proof, we begin by defining a function for our proposed LRS-PnP-DIP(1-Lip) algorithm, and we prove its validity. That is, we construct an energy function, H^k, that describes the behaviors of each states in Algorithm 3, and to prove that H^k is a non-increasing function as iteration proceeds; hence, the total energy of the dynamic system established via Algorithm 3 is conservative and stable.
Let $H^k = 2\|\boldsymbol{x}^k - \boldsymbol{x}^*\|^2 + \frac{1}{\mu^2}\|\boldsymbol{\lambda}_1^k - \boldsymbol{\lambda}_1^*\|^2 + \frac{1}{\mu^2}\|\boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^*\|^2$, for a non-zero $\boldsymbol{\mu}$.
Remarks. This design follows similar structures as in the original convergence proof of the ADMM algorithm [34] and in a recent work [57]. Here, H^k is a function of the system's state change, which is, by design, non-negative. The first two assumptions in Theorem 1.2 [55] automatically hold. Thus, we only need to show that the proposed candidate H^k is a non-increasing function in order to be a valid Lyapunov function. More specifically, we will show that H^k is a decreasing function, which satisfies the following:

$$H^k - H^{k+1} \geq C \tag{A16}$$

where C is a positive constant.

Firstly, recall that the LRS-PnP-DIP(1-Lip) algorithm takes the following update steps:

$$\boldsymbol{\alpha}^{k+1} = \mathcal{T}(\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_1^k}{\boldsymbol{\mu}}) \tag{A17}$$

$$\boldsymbol{u}^{k+1} = f_\theta(\boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}}) \tag{A18}$$

$$\begin{aligned} \boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{y} - \mathrm{M}\boldsymbol{x}\|_2^2 + \frac{\boldsymbol{\mu}}{2}\|(\boldsymbol{x} + \frac{\boldsymbol{\lambda}_1^k}{\boldsymbol{\mu}}) - \Phi\boldsymbol{\alpha}^{k+1}\|_2^2 \\ + \frac{\boldsymbol{\mu}}{2}\|(\boldsymbol{x} + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}}) - \boldsymbol{u}^{k+1}\|_2^2 \end{aligned} \tag{A19}$$

$$\begin{aligned} \boldsymbol{\lambda}_1^{k+1} &= \boldsymbol{\lambda}_1^k + \boldsymbol{\mu}(\boldsymbol{x}^{k+1} - \Phi\boldsymbol{\alpha}^{k+1}) \\ \boldsymbol{\lambda}_2^{k+1} &= \boldsymbol{\lambda}_2^k + \boldsymbol{\mu}(\boldsymbol{x}^{k+1} - \boldsymbol{u}^{k+1}) \end{aligned} \tag{A20}$$

We define $\boldsymbol{x}_e^k = \boldsymbol{x}^k - \boldsymbol{x}^*$, $\boldsymbol{u}_e^k = \boldsymbol{u}^k - \boldsymbol{u}^*$, $\boldsymbol{\alpha_e}^k = \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^*$, $\boldsymbol{\lambda}_{1e}^k = \boldsymbol{\lambda}_1^k - \boldsymbol{\lambda}_1^*$, and $\boldsymbol{\lambda}_{2e}^k = \boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^*$ in the subsequent proof for simplicity. This results in the following:

$$\begin{cases} \boldsymbol{x}_e^{k+1} = \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \\ \boldsymbol{u}_e^{k+1} = \boldsymbol{u}^{k+1} - \boldsymbol{u}^* \\ \boldsymbol{\alpha}_e^{k+1} = \boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^* \\ \boldsymbol{\lambda}_{1e}^{k+1} = \boldsymbol{\lambda}_1^{k+1} - \boldsymbol{\lambda}_1^* \\ \boldsymbol{\lambda}_{2e}^{k+1} = \boldsymbol{\lambda}_2^{k+1} - \boldsymbol{\lambda}_2^* \end{cases} \tag{A21}$$

We take Equation (A19) as our starting point and denote the first term, $\|\boldsymbol{y} - \mathrm{M}\boldsymbol{x}\|_2^2$, as $f(\boldsymbol{x})$. The first-order optimality of equation (A19) implies the following:

$$\begin{aligned} \nabla f(\boldsymbol{x}) + \boldsymbol{\mu}(\boldsymbol{x} + \frac{\boldsymbol{\lambda}_1^k}{\boldsymbol{\mu}} - \Phi\boldsymbol{\alpha}^{k+1}) \\ + \boldsymbol{\mu}(\boldsymbol{x} + \frac{\boldsymbol{\lambda}_2^k}{\boldsymbol{\mu}} - \boldsymbol{u}^{k+1}) = 0 \end{aligned} \tag{A22}$$

Since the minimizer $\boldsymbol{x}^{k+1}$ satisfies (A22), we plug it into (A22) to obtain the following:

$$\begin{aligned} &\nabla f(\boldsymbol{x})^{k+1} + \mu(\boldsymbol{x}^{k+1} - \Phi\boldsymbol{\alpha}^{k+1} + \frac{\boldsymbol{\lambda}_1^k}{\mu}) \\ &+\mu(\boldsymbol{x}^{k+1} - \boldsymbol{u}^{k+1} + \frac{\boldsymbol{\lambda}_2^k}{\mu}) \\ \overset{(A20)}{=} &\nabla f(\boldsymbol{x})^{k+1} + \mu(\frac{\boldsymbol{\lambda}_1^{k+1} - \boldsymbol{\lambda}_1^k}{\mu} + \frac{\boldsymbol{\lambda}_1^k}{\mu}) \\ &+\mu(\frac{\boldsymbol{\lambda}_2^{k+1} - \boldsymbol{\lambda}_1^k}{\mu} + \frac{\boldsymbol{\lambda}_2^k}{\mu}) \\ &= \nabla f(\boldsymbol{x})^{k+1} + \boldsymbol{\lambda}_1^{k+1} + \boldsymbol{\lambda}_2^{k+1} = 0 \end{aligned} \tag{A23}$$

The critical point satisfies, i.e., $k \to \infty$:

$$\nabla f(\boldsymbol{x})^* + \boldsymbol{\lambda}_1^* + \boldsymbol{\lambda}_2^* = 0 \tag{A24}$$

Due to the strong convexity of $f(\boldsymbol{x})$, using Lemma 2 with $\boldsymbol{x} = \boldsymbol{x}^{k+1}$ and $\boldsymbol{y} = \boldsymbol{x}^*$ yields the following:

$$\left\langle \nabla f(\boldsymbol{x})^{k+1} - \nabla f(\boldsymbol{x})^*, \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \right\rangle \geq \rho \|(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)\|^2 \tag{A25}$$

Combining the above inequality with Equations (A23) and (A24), we have the following:

$$\left\langle -\boldsymbol{\lambda}_{1e}^{k+1} - \boldsymbol{\lambda}_{2e}^{k+1}, \boldsymbol{x}_e^{k+1} \right\rangle \geq \rho \|\boldsymbol{x}_e^{k+1}\|^2 \tag{A26}$$

Secondly, using Assumption 2, the DHP $f_\theta(z)$ is L-Lipschitz with $L \leq 1$ and $\boldsymbol{x} = \boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_2^k}{\mu}$, $\boldsymbol{y} = \boldsymbol{x}^* + \frac{\boldsymbol{\lambda}_2^*}{\mu}$; we get the following:

$$\begin{aligned} \|\boldsymbol{u}^{k+1} - \boldsymbol{u}^*\|^2 &\leq \|(\boldsymbol{x}^k - \boldsymbol{x}^*) + \frac{\boldsymbol{\lambda}_2^k - \boldsymbol{\lambda}_2^*}{\mu}\|^2 \\ or \|\boldsymbol{u}_e^{k+1}\|^2 &\leq \|\boldsymbol{x}_e^k + \frac{\boldsymbol{\lambda}_{2e}^k}{\mu}\|^2 \end{aligned} \tag{A27}$$

Thirdly, using Assumption 1 and the resulting Lemma 1, suggesting that the operator $\mathcal{T}$ used in the $\boldsymbol{\alpha}$ updating step (A17) is θ-averaged, and $\boldsymbol{x} = \boldsymbol{x}^k + \frac{\boldsymbol{\lambda}_1^k}{\mu}$, $\boldsymbol{y} = \boldsymbol{x}^* + \frac{\boldsymbol{\lambda}_1^*}{\mu}$, we have the following:

$$\begin{aligned} \|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\|^2 &\leq \|(\boldsymbol{x}^k - \boldsymbol{x}^*) + \frac{\boldsymbol{\lambda}_1^k - \boldsymbol{\lambda}_1^*}{\mu}\|^2 \\ or \|\boldsymbol{\alpha}_e^{k+1}\|^2 &\leq \|\boldsymbol{x}_e^k + \frac{\boldsymbol{\lambda}_{1e}^k}{\mu}\|^2 \end{aligned} \tag{A28}$$

Now, we multiply (A26) on both sides by $\frac{2}{\mu}$, and we gather the resulting inequality with (A27) and (A28):

$$\begin{cases} -\frac{2}{\mu}\left\langle \boldsymbol{\lambda}_{1e}^{k+1}, \boldsymbol{x}_e^{k+1} \right\rangle - \frac{2}{\mu}\left\langle \boldsymbol{\lambda}_{2e}^{k+1}, \boldsymbol{x}_e^{k+1} \right\rangle \geq \frac{2}{\mu}\rho \|\boldsymbol{x}_e^{k+1}\|^2 \\ \|\boldsymbol{x}_e^k + \frac{\boldsymbol{\lambda}_{2e}^k}{\mu}\|^2 \geq \|\boldsymbol{u}_e^{k+1}\|^2 \\ \|\boldsymbol{x}_e^k + \frac{\boldsymbol{\lambda}_{1e}^k}{\mu}\|^2 \geq \|\boldsymbol{\alpha}_e^{k+1}\|^2 \end{cases} \tag{A29}$$

We put μ inside the left-hand side of the first inequality, and we add them to give the following:

$$\begin{aligned} &-2\left\langle \frac{\lambda_{1e}^{k+1}}{\mu}, x_e^{k+1} \right\rangle - 2\left\langle \frac{\lambda_{2e}^{k+1}}{\mu}, x_e^{k+1} \right\rangle + \|x_e^k + \frac{\lambda_{2e}^k}{\mu}\|^2 \\ &+\|x_e^k + \frac{\lambda_{1e}^k}{\mu}\|^2 \geq \frac{2}{\mu}\rho\|x_e^{k+1}\|^2 + \|u_e^{k+1}\|^2 + \|\alpha_e^{k+1}\|^2 \end{aligned} \tag{A30}$$

which can be written as:

$$\begin{aligned} &-(\frac{1}{\mu^2}\|\lambda_{1e}^{k+1}\|^2 + \|x_e^{k+1}\|^2 - \|\frac{\lambda_{1e}^{k+1}}{\mu} - x_e^{k+1}\|^2) \\ &-(\frac{1}{\mu^2}\|\lambda_{2e}^{k+1}\|^2 + \|x_e^{k+1}\|^2 - \|\frac{\lambda_{2e}^{k+1}}{\mu} - x_e^{k+1}\|^2) \\ &\quad +\frac{1}{\mu^2}\|\lambda_{2e}^k\|^2 + \|x_e^k\|^2 + 2\left\langle \frac{\lambda_{2e}^k}{\mu}, x_e^k \right\rangle \\ &\quad +\frac{1}{\mu^2}\|\lambda_{1e}^k\|^2 + \|x_e^k\|^2 + 2\left\langle \frac{\lambda_{1e}^k}{\mu}, x_e^k \right\rangle \\ &\quad \geq \frac{2}{\mu}\rho\|x_e^{k+1}\|^2 + \|u_e^{k+1}\|^2 + \|\alpha_e^{k+1}\|^2 \end{aligned} \tag{A31}$$

After rearrangement, we get the following:

$$\begin{aligned} &2(\|x_e^k\|^2 - \|x_e^{k+1}\|^2) + \frac{1}{\mu^2}(\|\lambda_{1e}^k\|^2 - \|\lambda_{1e}^{k+1}\|^2) + \frac{1}{\mu^2}(\|\lambda_{2e}^k\|^2 \\ &\quad - \|\lambda_{2e}^{k+1}\|^2) \\ &\quad \geq \frac{2}{\mu}\rho\|x_e^{k+1}\|^2 + \|u_e^{k+1}\|^2 + \|\alpha_e^{k+1}\|^2 \\ &\quad + \|\frac{\lambda_{1e}^{k+1}}{\mu} - x_e^{k+1}\|^2 + \|\frac{\lambda_{2e}^{k+1}}{\mu} - x_e^{k+1}\|^2 \\ &\quad - 2\left\langle \frac{\lambda_{2e}^k}{\mu}, x_e^k \right\rangle - 2\left\langle \frac{\lambda_{1e}^k}{\mu}, x_e^k \right\rangle \end{aligned} \tag{A32}$$

Recall that

$$\begin{aligned} &H^k - H^{k+1} \\ &\quad = 2\|x^k - x^*\|^2 + \frac{1}{\mu^2}\|\lambda_1^k - \lambda_1^*\|^2 + \frac{1}{\mu^2}\|\lambda_2^k - \lambda_2^*\|^2 \\ &\quad - 2\|x^{k+1} - x^*\|^2 - \frac{1}{\mu^2}\|\lambda_1^{k+1} - \lambda_1^*\|^2 - \frac{1}{\mu^2}\|\lambda_2^{k+1} - \lambda_2^*\|^2 \\ &\quad \overset{(A21)}{=} 2(\|x_e^k\|^2 - \|x_e^{k+1}\|^2) + \frac{1}{\mu^2}(\|\lambda_{1e}^k\|^2 - \|\lambda_{1e}^{k+1}\|^2) \\ &\quad + \frac{1}{\mu^2}(\|\lambda_{2e}^k\|^2 - \|\lambda_{2e}^{k+1}\|^2) \end{aligned} \tag{A33}$$

It can be seen that the left-hand side of inequality (A32) recovers exactly $H^k - H^{k+1}$. To make H^k a non-increasing function, we require the entire right-hand side to be non-negative; it is thus sufficient to show that the last two terms, $-2\left\langle \frac{\lambda_{2e}^k}{\mu}, x_e^k \right\rangle - 2\left\langle \frac{\lambda_{1e}^k}{\mu}, x_e^k \right\rangle$, are non-negative. This is straightforward to show if we plug $k = k - 1$ into the first line of (A29) to get the following:

$$
\begin{aligned}
H^k - H^{k+1} \geq & \frac{2}{\mu}\rho\|\boldsymbol{x}_e^{k+1}\|^2 + \|\boldsymbol{u}_e^{k+1}\|^2 + \|\boldsymbol{\alpha}_e^{k+1}\|^2 \\
& + \|\frac{\boldsymbol{\lambda}_{1e}^{k+1}}{\mu} - \boldsymbol{x}_e^{k+1}\|^2 + \|\frac{\boldsymbol{\lambda}_{2e}^{k+1}}{\mu} - \boldsymbol{x}_e^{k+1}\|^2 \\
& + \frac{2}{\mu}\rho\|\boldsymbol{x}_e^{k}\|^2 \\
& \geq 0
\end{aligned} \tag{A34}
$$

Now, if we add both sides of (A33), from $k = 0$ to $k = \infty$, it follows that

$$
\begin{aligned}
\sum_{k=0}^{\infty} \frac{2}{\mu}\rho\|\boldsymbol{x}_e^{k+1}\|^2 + \sum_{k=0}^{\infty} \|\boldsymbol{u}_e^{k+1}\|^2 + \sum_{k=0}^{\infty} \|\boldsymbol{\alpha}_e^{k+1}\|^2 \\
\leq H^0 - H^\infty < \infty
\end{aligned} \tag{A35}
$$

We can conclude that sequences $\boldsymbol{x}_e^{k+1}$, $\boldsymbol{u}_e^{k+1}$ and $\boldsymbol{\alpha}_e^{k+1}$ are all bounded sequences due to the Lyapunov theorem. That is, as $k \to \infty$:

$$
\begin{aligned}
\lim_{k\to\infty} \|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\|^2 \to 0 \\
\lim_{k\to\infty} \|\boldsymbol{u}^{k+1} - \boldsymbol{u}^*\|^2 \to 0 \\
\lim_{k\to\infty} \|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^*\|^2 \to 0
\end{aligned} \tag{A36}
$$

Thus, the iterations generated via the LRS-PnP-DIP(1-Lip) algorithm converge to the critical points $(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\alpha}^*)$ with a sufficiently large k, and all the trajectories are bounded. LRS-PnP-DIP(1-Lip) is also asymptotically stable according to Theorem 3.3 [55].

Appendix A.3. Parameters Tuning and Ablation Tests

Appendix A.3.1. Sensitivity Analysis and Hyperparameters Selections

In this section, we provide the detailed selection of hyperparameters for the proposed LRS-PnP-DIP(1-Lip) algorithm in the experiments. In Table A2, we provide the inpainting performance of LRS-PnP-DIP(1-Lip) with the weights of low-rank constraint w_{lr} and sparsity constraints w_s varying within the range of 0.1 to 1. It can be seen that the optimal performance is obtained when both low-rank and sparsity constraints are present, as long as there is no overwhelming of either w_{lr} or w_s. This indicates that LRS-PnP-DIP(1-Lip) is insensitive to the precise value of the low-rank constraint w_{lr} or sparsity constraints w_s, which is different from the LRS-PnP algorithm [20] discussed in Section 4.2. Therefore, we set both w_{lr} and w_s to 1 across all experiments.

Table A1. The selection of hyperparameters for the proposed LRS-PnP-DIP(1-Lip) algorithm.

Parameters	**Number of Iterations**	γ	w_{lr}	w_s	μ_1	μ_2	λ_1	λ_2	σ_y
	200	1/2	1	1	1/2	1/2	0	0	0.12
Parameters	**DIP Learning Rate**	**DIP perturbed Noise Level**	**Lipschitz Constant: L**	**PnP-ISTA Iterations**	λ_{ISTA}				
	0.1	0	1	50	0.1				

Table A2. Inpainting MPSNR of LRS-PnP-DIP(1-Lip) algorithm with varying w_{lr} and w_s. Using either overwhelming w_{lr} or w_s could deteriorate the inpainting results.

w_s \ w_{lr}	**0**	**0.1**	**0.25**	**0.5**	**0.75**	**1**
0	30.692	37.479(±0.63)	38.121(±0.46)	38.475(±0.48)	38.686(±0.52)	38.884(±0.31)
0.1	37.004	40.849(±0.31)	39.107(±0.35)	39.321(±0.37)	39.443(±0.30)	39.812(±0.34)
0.25	37.211	40.355(±0.29)	40.798(±0.36)	40.266(±0.29)	39.770(±0.23)	39.906(±0.31)
0.5	37.592	39.766(±0.27)	40.807(±0.24)	40.831(±0.26)	40.532(±0.25)	40.245(±0.32)
0.75	37.663	39.897(±0.37)	40.373(±0.33)	40.565(±0.27)	40.807(±0.30)	40.780(±0.26)
1	37.957	38.591(±0.46)	39.248(±0.31)	39.848(±0.25)	39.902(±0.25)	40.835(±0.28)

Appendix A.3.2. Computational Efficiency

Table A3 provides per-iteration running time and the total running time upon converge for each method discussed in Section 4.5. To ensure a fair comparison, all methods, except for GLON [4], which uses a pre-trained FFDNet, were implemented with the same DIP backbone. All experiments were conducted in Python with PyTorch 1.13 using the NVIDIA GeForce RTX 3090 GPU. It can be seen that GLON suggests the fastest inference time, but at the cost of sub-optimal inpainting performance. DHP [11], R-DLRHyIn [16], and DeepHyIn [50] share similar computational complexity; these are followed by DDS2M [19], which typically requires a large number of diffusion steps in order to yield faithful reconstruction. On the other hand, methods such as DeepRED [22], PnP-DIP [24] and LRS-PnP-DIP(1-Lip) are computationally more expensive, which is mainly due to the use of PnP denoisers as regularizers at each iteration. Although LRS-PnP-DIP(1-Lip) introduces extra computation complexity to solve the sparsity-constrained problem (5), its overall running time is competitive with that of the DeepRED and PnP-DIP methods, This is mainly because LRS-PnP-DIP(1-Lip) enjoys a stability guarantee, which makes it converge faster with fewer iterations.

Table A3. Different algorithms and their running times on both the Chikusei and Indian Pines datasets.

Dataset	**Cost**	**GLON** [4]	**DHP** [11]	**R-DLRHyIn** [16]	**DeepHyIn** [50]	**DDS2M** [19]	**DeepRED** [22]	**PnP-DIP** [24]	**LRS-PnP-DIP(1-Lip)**
Chikusei	**per-iteration**	0.138	0.083	0.089	0.102	0.114	2.412	2.542	3.622
	all iterations	69.302	89.596	106.855	95.351	382.027	482.985	473.064	394.601
Indian Pines	**per-iteration**	0.104	0.109	0.115	0.136	0.149	2.732	2.967	3.905
	all iterations	74.741	97.215	115.541	103.344	401.654	494.088	474.156	425.601

Appendix A.3.3. Effect of the Lipschitz Constraint DIP

As an extension to the LRS-PnP-DIP algorithm, LRS-PnP-DIP(1-Lip) enforces the Lipschitz continuity (A14) of DIP by applying the Lipschitz constraint to each layer of the neural network independently with Lipschitz constant $L = 1$. In the proposed LRS-PnP-DIP(1-Lip) algorithm, we remove the residual connections from the DIP (as was discussed in Appendix A.1.2). However, changing from Skip-Net to U-Net may potentially lead to reduced performance, which is part of the trade-off with the convergence. In Figure A1, we provide an ablation test comparing the performance of LRS-PnP-DIP with/without 1-Lipschitz constraint. It can be seen that 1-Lipschitz constraint promotes the stability of the reconstruction process, ensuring the convergence of the algorithm. This improvement comes at the cost of only a slight reduction in inpainting performance.

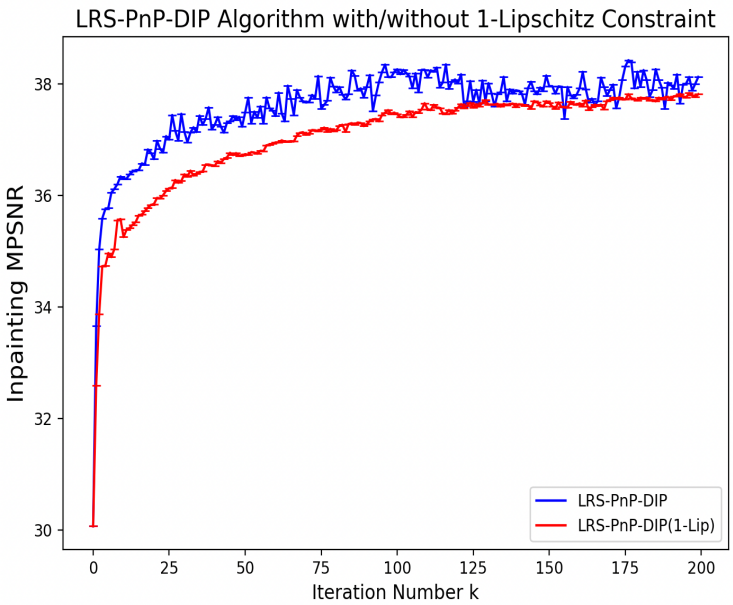

Figure A1. Inpainting performance of LRS-PnP-DIP algorithm with/without 1-Lipschitz constraint.

Appendix A.3.4. Effect of the Averaged Denoiser

Compared to the conventional NLM denoiser, the averaged property of the modified NLM denoiser imposes a significantly more restrictive condition. Such a modification also raises the question as to whether it will damage the performance of the original one. In Table A4, we record the inpainting MPSNR and MSSIM of the LRS-PnP-DIP algorithm [20] with different PnP denoisers; the LRS-PnP-DIP with averaged NLM denoiser achieves almost the same performance as with the BM3D or the conventional NLM denoiser.

Table A4. Inpainting performance of LRS-PnP-DIP algorithm with different PnP denoisers, under noise level $\sigma_y = 0.1, 0.15, 0.2$, respectively. The mean and variance over 20 samples are shown here. The best results are highlighted in bold.

Method	Metric	Input	BM3D Denoiser	NLM Denoiser	Averaged NLM Denoiser
$\sigma_y = 0.1$	MPSNR ↑	31.76	41.23(±0.25)	41.25(±0.30)	**41.22**(±0.13)
	MSSIM ↑	0.304	0.920(±0.01)	0.918(±0.01)	**0.920**(±0.01)
$\sigma_y = 0.15$	MPSNR ↑	30.38	39.55(±0.29)	39.60(±0.22)	**39.55**(±0.220)
	MSSIM ↑	0.247	0.913(±0.01)	0.915(±0.01)	**0.914**(±0.01)
$\sigma_y = 0.2$	MPSNR ↑	28.75	37.68(±0.32)	37.65(±0.35)	**37.67**(±0.24)
	MSSIM ↑	0.229	0.904(±0.01)	0.903(±0.01)	**0.903**(±0.01)

Appendix A.3.5. Effect of the DIP Network Architectures

As was reported in a series of DIP-related works [10,11,24,50], the performance of DIP is sensitive to the structure of the neural networks. Table A5 provides a comparative analysis of several network architectures, including ResNet 2D/3D, U-Net 2D/3D, and Skip-Net 2D/3D. Empirical results indicate that Skip-Net 2D (i.e., a U-Net architecture with skip connections and 2D convolution layers) achieves the best performance for the HSI inpainting task. Hence, Skip-Net 2D was selected and implemented as the backbone architecture for the proposed LRS-PnP-DIP, LRS-PnP-DIP(1-Lip), and all other competing methods in the comparative analysis. Furthermore, it was found that ResNet architectures yield inferior results within the proposed framework, highlighting the importance of network design choices for achieving optimal inpainting performance. We left the exploration

and identification of the most suitable DIP network structure for HSI inpainting as a future research direction.

Table A5. On the choice of different DIP network architectures in the proposed LRS-PnP-DIP algorithm. For the metrics MPSNR and MSSIM, higher values indicate better performance. The best results are highlighted in bold.

Methods	MPSNR↑	MSSIM↑
Input	22.582	0.178
ResNet 2D	30.975(±0.62)	0.610(±0.02)
ResNet 3D	29.661(±1.20)	0.589(±0.03)
UNet 2D	35.963(±0.42)	0.882(±0.02)
UNet 3D	35.438(±0.56)	0.868(±0.01)
Skip-Net 2D	**37.971**(±0.30)	**0.899**(±0.01)
Skip-Net 3D	37.050(±0.32)	0.890(±0.01)

References

1. Ortega, S.; Guerra, R.; Diaz, M.; Fabelo, H.; López, S.; Callico, G.M.; Sarmiento, R. Hyperspectral push-broom microscope development and characterization. *IEEE Access* **2019**, *7*, 122473–122491. [CrossRef]
2. EMIT L1B At-Sensor Calibrated Radiance and Geolocation Data 60 m V001. Available online: https://search.earthdata.nasa.gov/search/granules?p=C2408009906-LPCLOUD&pg[0][v]=f&pg[0][gsk]=-start_date&q=%22EMIT%22&tl=1711560236!3!! (accessed on 27 August 2024).
3. Zhuang, L.; Bioucas-Dias, J.M. Fast hyperspectral image denoising and inpainting based on low-rank and sparse representations. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2018**, *11*, 730–742. [CrossRef]
4. Zhao, X.L.; Yang, J.H.; Ma, T.H.; Jiang, T.X.; Ng, M.K.; Huang, T.Z. Tensor completion via complementary global, local, and nonlocal priors. *IEEE Trans. Image Process.* **2021**, *31*, 984–999. [CrossRef]
5. Li, B.Z.; Zhao, X.L.; Wang, J.L.; Chen, Y.; Jiang, T.X.; Liu, J. Tensor completion via collaborative sparse and low-rank transforms. *IEEE Trans. Comput. Imaging* **2021**, *7*, 1289–1303. [CrossRef]
6. Luo, Y.S.; Zhao, X.L.; Jiang, T.X.; Chang, Y.; Ng, M.K.; Li, C. Self-supervised nonlinear transform-based tensor nuclear norm for multi-dimensional image recovery. *IEEE Trans. Image Process.* **2022**, *31*, 3793–3808. [CrossRef]
7. Wu, Y.; Liu, J.; Gong, M.; Liu, Z.; Miao, Q.; Ma, W. MPCT: Multiscale point cloud transformer with a residual network. *IEEE Trans. Multimed.* **2023**, *26*, 3505–3516. [CrossRef]
8. Yuan, Y.; Wu, Y.; Fan, X.; Gong, M.; Ma, W.; Miao, Q. EGST: Enhanced geometric structure transformer for point cloud registration. *IEEE Trans. Vis. Comput. Graph.* **2023**, *30*, 6222–6234. [CrossRef] [PubMed]
9. Wong, R.; Zhang, Z.; Wang, Y.; Chen, F.; Zeng, D. HSI-IPNet: Hyperspectral imagery inpainting by deep learning with adaptive spectral extraction. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 4369–4380. [CrossRef]
10. Ulyanov, D.; Vedaldi, A.; Lempitsky, V. Deep image prior. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 9446–9454.
11. Sidorov, O.; Yngve Hardeberg, J. Deep hyperspectral prior: Single-image denoising, inpainting, super-resolution. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27 October–2 November 2019.
12. Meng, Z.; Yu, Z.; Xu, K.; Yuan, X. Self-supervised neural networks for spectral snapshot compressive imaging. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2622–2631.
13. Lai, Z.; Wei, K.; Fu, Y. Deep Plug-and-Play Prior for Hyperspectral Image Restoration. *Neurocomputing* **2022**, *481*, 281–293. [CrossRef]
14. Wu, Z.; Yang, C.; Su, X.; Yuan, X. Adaptive Deep PnP Algorithm for Video Snapshot Compressive Imaging. *arXiv* **2022**, arXiv:2201.05483. [CrossRef]
15. Gan, W.; Eldeniz, C.; Liu, J.; Chen, S.; An, H.; Kamilov, U.S. Image reconstruction for mri using deep cnn priors trained without groundtruth. In Proceedings of the 2020 54th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 1–4 November 2020; pp. 475–479.
16. Niresi, K.F.; Chi, C.Y. Robust Hyperspectral Inpainting via Low-Rank Regularized Untrained Convolutional Neural Network. *IEEE Geosci. Remote Sens. Lett.* **2023**, *20*, 1–5. [CrossRef]

17. Hong, D.; Yao, J.; Li, C.; Meng, D.; Yokoya, N.; Chanussot, J. Decoupled-and-coupled networks: Self-supervised hyperspectral image super-resolution with subpixel fusion. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5527812. [CrossRef]
18. Yao, J.; Hong, D.; Chanussot, J.; Meng, D.; Zhu, X.; Xu, Z. Cross-attention in coupled unmixing nets for unsupervised hyperspectral super-resolution. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Part XXIX 16, pp. 208–224.
19. Miao, Y.; Zhang, L.; Zhang, L.; Tao, D. Dds2m: Self-supervised denoising diffusion spatio-spectral model for hyperspectral image restoration. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Paris, France, 2–6 October 2023; pp. 12086–12096.
20. Li, S.; Yaghoobi, M. Self-Supervised Hyperspectral Inpainting with the Optimisation inspired Deep Neural Network Prior. In Proceedings of the 2023 31st European Signal Processing Conference (EUSIPCO), Helsinki, Finland, 4–8 September 2023; pp. 471–475.
21. Heckel, R.; Hand, P. Deep decoder: Concise image representations from untrained non-convolutional networks. *arXiv* **2018**, arXiv:1810.03982.
22. Mataev, G.; Milanfar, P.; Elad, M. DeepRED: Deep image prior powered by RED. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Republic of Korea, 27 October–2 November 2019.
23. Liu, J.; Sun, Y.; Xu, X.; Kamilov, U.S. Image restoration using total variation regularized deep image prior. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 7715–7719.
24. Sun, Z.; Latorre, F.; Sanchez, T.; Cevher, V. A plug-and-play deep image prior. In Proceedings of the ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada, 6–11 June 2021; pp. 8103–8107.
25. Chen, Y.; Nasrabadi, N.M.; Tran, T.D. Hyperspectral image classification using dictionary-based sparse representation. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 3973–3985. [CrossRef]
26. Zhang, H.; He, W.; Zhang, L.; Shen, H.; Yuan, Q. Hyperspectral image restoration using low-rank matrix recovery. *IEEE Trans. Geosci. Remote Sens.* **2013**, *52*, 4729–4743. [CrossRef]
27. Li, C.; Zhang, B.; Hong, D.; Yao, J.; Chanussot, J. LRR-Net: An Interpretable Deep Unfolding Network for Hyperspectral Anomaly Detection. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 1–12. [CrossRef]
28. Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G. Online dictionary learning for sparse coding. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 689–696.
29. Peng, J.; Sun, W.; Du, Q. Self-paced joint sparse representation for the classification of hyperspectral images. *IEEE Trans. Geosci. Remote Sens.* **2018**, *57*, 1183–1194. [CrossRef]
30. Zhao, Y.Q.; Yang, J. Hyperspectral image denoising via sparse representation and low-rank constraint. *IEEE Trans. Geosci. Remote Sens.* **2014**, *53*, 296–308. [CrossRef]
31. Iordache, M.D.; Bioucas-Dias, J.M.; Plaza, A. Sparse unmixing of hyperspectral data. *IEEE Trans. Geosci. Remote Sens.* **2011**, *49*, 2014–2039. [CrossRef]
32. Xue, J.; Zhao, Y.; Liao, W.; Kong, S.G. Joint spatial and spectral low-rank regularization for hyperspectral image denoising. *IEEE Trans. Geosci. Remote Sens.* **2017**, *56*, 1940–1958. [CrossRef]
33. Ma, G.; Huang, T.Z.; Huang, J.; Zheng, C.C. Local low-rank and sparse representation for hyperspectral image denoising. *IEEE Access* **2019**, *7*, 79850–79865. [CrossRef]
34. Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends® Mach. Learn.* **2011**, *3*, 1–122.
35. Parikh, N.; Boyd, S. Proximal algorithms. *Found. Trends® Optim.* **2014**, *1*, 127–239. [CrossRef]
36. Venkatakrishnan, S.V.; Bouman, C.A.; Wohlberg, B. Plug-and-play priors for model based reconstruction. In Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing, Austin, TX, USA, 3–5 December 2013; pp. 945–948.
37. Daubechies, I.; Defrise, M.; De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math. J. Issued Courant Inst. Math. Sci.* **2004**, *57*, 1413–1457. [CrossRef]
38. Cai, J.F.; Candès, E.J.; Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* **2010**, *20*, 1956–1982. [CrossRef]
39. Sreehari, S.; Venkatakrishnan, S.V.; Wohlberg, B.; Buzzard, G.T.; Drummy, L.F.; Simmons, J.P.; Bouman, C.A. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Trans. Comput. Imaging* **2016**, *2*, 408–423. [CrossRef]
40. Nair, P.; Gavaskar, R.G.; Chaudhury, K.N. Fixed-point and objective convergence of plug-and-play algorithms. *IEEE Trans. Comput. Imaging* **2021**, *7*, 337–348. [CrossRef]
41. Ekeland, I.; Temam, R. *Convex Analysis and Variational Problems*; SIAM: Philadelphia, PA, USA, 1999.
42. Ryu, E.; Liu, J.; Wang, S.; Chen, X.; Wang, Z.; Yin, W. Plug-and-play methods provably converge with properly trained denoisers. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5546–5557.
43. Yokoya, N.; Iwasaki, A. Airborne Hyperspectral Data over Chikusei; SAL-2016-05-27; University of Tokyo: Tokyo, Japan, 2016.

44. Baumgardner, M.F.; Biehl, L.L.; Landgrebe, D.A. 220 band AVIRIS hyperspectral image data set: June 12, 1992 Indian Pine test site 3. *Purdue Univ. Res. Repos.* **2015**, *10*, 991.
45. Denby, B.; Lucia, B. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 16–20 March 2020; pp. 939–954.
46. Gouk, H.; Frank, E.; Pfahringer, B.; Cree, M.J. Regularisation of neural networks by enforcing lipschitz continuity. *Mach. Learn.* **2021**, *110*, 393–416. [CrossRef]
47. Wang, H.; Li, T.; Zhuang, Z.; Chen, T.; Liang, H.; Sun, J. Early Stopping for Deep Image Prior. *arXiv* **2021**, arXiv:2112.06074.
48. Nguyen, H.V.; Ulfarsson, M.O.; Sigurdsson, J.; Sveinsson, J.R. Deep sparse and low-rank prior for hyperspectral image denoising. In Proceedings of the IGARSS 2022–2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia, 17–22 July 2022; pp. 1217–1220.
49. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv* **2018**, arXiv:1802.05957.
50. Rasti, B.; Ghamisi, P.; Gloaguen, R. Unsupervised deep hyperspectral inpainting using a new mixing model. In Proceedings of the IGARSS 2022–2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia, 17–22 July 2022; pp. 1221–1224.
51. Maffei, A.; Haut, J.M.; Paoletti, M.E.; Plaza, J.; Bruzzone, L.; Plaza, A. A single model CNN for hyperspectral image denoising. *IEEE Trans. Geosci. Remote Sens.* **2019**, *58*, 2516–2529. [CrossRef]
52. Cascarano, P.; Sebastiani, A.; Comes, M.C. ADMM DIP-TV: Combining Total Variation and Deep Image Prior for image restoration. *arXiv* **2020**, arXiv:2009.11380.
53. Jiao, L.; Zhang, X.; Liu, X.; Liu, F.; Yang, S.; Ma, W.; Li, L.; Chen, P.; Feng, Z.; Guo, Y.; et al. Transformer meets remote sensing video detection and tracking: A comprehensive survey. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2023**, *16*, 1–45. [CrossRef]
54. Shevitz, D.; Paden, B. Lyapunov stability theory of nonsmooth systems. *IEEE Trans. Autom. Control* **1994**, *39*, 1910–1914. [CrossRef]
55. Bof, N.; Carli, R.; Schenato, L. Lyapunov theory for discrete time systems. *arXiv* **2018**, arXiv:1809.05289.
56. Hill, D.; Moylan, P. The stability of nonlinear dissipative systems. *IEEE Trans. Autom. Control* **1976**, *21*, 708–711. [CrossRef]
57. Zhang, T.; Shen, Z. A fundamental proof of convergence of alternating direction method of multipliers for weakly convex optimization. *J. Inequal. Appl.* **2019**, *2019*, 128. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.