# Computationally Efficient Sparse Signal Recovery by Deep Unfolded-Periodic Sketched ISTA

Tatsuki Tokumura, Ayano Nakai-Kasai and Tadashi Wadayama
Nagoya Institute of Technology, Japan
E-mail: t.tokumura762@stn.nitech.ac.jp {nakai.ayano,wadayama}@nitech.ac.jp

***Abstract*—In this paper, we propose a novel sparse signal recovery algorithm, DU-PSISTA (Deep Unfolded-Periodic Sketched Iterative Shrinkage-Thresholding Algorithm), which aims to balance computational efficiency and accuracy for recovering high-dimensional sparse signals. The proposed method periodically alternates between standard ISTA and a sketched variant that leverages random projections to reduce the dimensionality of gradient computations. This hybrid structure enables flexible control over the trade-off between accuracy and computational complexity through a pre-configurable period parameter. Furthermore, we incorporate deep unfolding into the algorithm that embeds learnable parameters such as step sizes and thresholding factors into each iteration. These parameters are optimized through data-driven training, enabling the model to adaptively improve convergence and performance. Numerical experiments confirm that our method achieves comparable recovery performance to conventional deep unfolded ISTA while reducing computation, especially when the period parameter and sketch size are properly selected.**

## I. Introduction

Recent advances in sensing technologies, wireless communication systems, medical imaging, and IoT (Internet of Things) devices have led to a dramatic increase in both the dimensionality and volume of data acquired from the real world. For instance, fMRI (functional Magnetic Resonance Imaging) for brain activity mapping typically requires the acquisition of tens of thousands of voxel values at high temporal resolution, resulting in massive, high-dimensional datasets. Similarly, in applications such as wireless communications, radar systems, and image processing, acquired signals are frequently high-dimensional and often include noise or missing entries. These challenges necessitate the development of signal processing methods capable of efficient and accurate recovery of such high-dimensional data [1].

In recent years, a sparse signal processing technique known as compressed sensing [2] has attracted considerable attention. These approaches simultaneously enable dimensionality reduction and salient feature extraction, facilitating their deployment in a wide range of fields, including medical imaging, audio processing, communications, and robotics. One example of this demand lies in next-generation wireless networks. With the emergence of 6G (6th Generation Mobile Communication System) and IoT technologies, the number of connected devices is growing exponentially, leading to a corresponding increase in the volume of signals exchanged between base stations and devices. Such systems demand highly efficient and accurate execution of fundamental tasks such as channel estimation and active user detection. Especially, in grant-free uplink scenarios [3], where base station scheduling is eliminated, the activity patterns of devices exhibit temporal and spatial sparsity. As a result, the problem of active user detection can be formulated as a sparse signal recovery problem to be solved in real-time speed.

ISTA (Iterative Shrinkage-Thresholding Algorithm) [4] is a widely adopted iterative optimization method for solving such a problem due to its simplicity and theoretical convergence guarantees. However, ISTA includes matrix-vector product in each iteration, leading to high computational cost when dealing with large-scale data matrices. This cost can become a critical bottleneck for real-time applications with strict latency requirements.

To address this limitation, we focus on recent progress in sketching techniques [5] to reduce the computational burden of iterative algorithms. Sketching projects high-dimensional data onto a lower-dimensional subspace using random projections while approximately preserving the original problem structure. Algorithms such as IHS (Iterative Hessian Sketch) and GPIS (Gradient Projection Iterative Sketch) have demonstrated the ability to approximate solutions to the original problem using significantly fewer measurements, with theoretical guarantees on performance [1]. However, there is a trade-off between reducible dimension and convergence performance due to the approximated structure of sketching. It is desirable to develop a method that achieves both high computational efficiency and reasonable recovery performance when applied to ISTA.

In this paper, we propose a hybrid framework that leverages both original updates and sketched updates of ISTA. The proposed algorithm alternates the original and sketched updates in a periodic manner to reduce the computational load associated with the matrix-vector product calculation. This algorithm design is expected to suppress deterioration of convergence performance caused by sketching. This hybrid architecture has a potential to flexibly control a trade-off between computational efficiency and recovery accuracy, which can be adjusted according to system requirements and environmental constraints.

The proposed hybrid algorithm has a number of tunable parameters that can affect performance but it is not trivial to set appropriate values for all of them. We therefore incorporate the concept of deep unfolding [6] into the proposed algorithm, which interprets iterative optimization algorithms as feedforward neural networks, and enables optimization of

979-8-3315-7206-8/25/$31.00 

embedded learnable parameters by data-driven training [7]. Unlike deep neural networks, deep unfolding models maintain interpretability and generally require fewer training samples. This approach builds on Gregor and LeCun's work improving ISTA and demonstrating its effectiveness, with reported success in applications such as MIMO (Multiple-Input Multiple-Output) signal detection and sparse signal estimation [8]–[10].

The goal of this work is to develop a novel sparse signal recovery algorithm that integrates sketching for computationally efficient operations and deep unfolding for performance improvement, thereby enabling efficient and scalable recovery of high-dimensional sparse signals in practical applications.

## II. Preliminaries

### A. ISTA

Compressed sensing [11] is the problem of estimating an unknown sparse original signal vector based on the observation vector and a known observation matrix. A sparse vector is a vector in which the number of nonzero elements is small relative to the length of the vector. This problem is defined by the following observation model. Let the original sparse signal vector be $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}} \in \mathbb{R}^n$ and the observation matrix be $\boldsymbol{A} \in \mathbb{R}^{m \times n} (m < n)$. Then, the observation model is expressed as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}$, where the noise vector $\boldsymbol{w} \in \mathbb{R}^m$ follows Gaussian distribution with mean $\boldsymbol{0}$ and covariance $\sigma^2 \boldsymbol{I}$.

To solve the sparse signal estimate problem, one can consider the convex optimization problem called LASSO (Least Absolute Shrinkage and Selection Operator) [12], which corresponds to a regularized least squares method and the estimated vector is given by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^n} \left( \frac{1}{2} ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 + \tau ||\boldsymbol{x}||_1 \right), \tag{1}$$

where $||\boldsymbol{x}||_1 = \sum_{i=1}^n |x_i|$ is the $\ell^1$ norm. The regularization coefficient $\tau \in \mathbb{R}$ $(\tau > 0)$ can be used to control the strength of the $\ell^1$ regularization term. The first term works to reduce the squared error between the original signal vector and the estimated signal vector, and the second term works to make the estimated signal vector more sparse. As a method for solving LASSO, ISTA [4] is a well-known algorithm. ISTA is a method that iteratively minimizes the cost function of (1) by the proximal gradient method. The iterative formula of ISTA is defined as

$$\boldsymbol{z}^{(t)} = \boldsymbol{x}^{(t)} - \eta \boldsymbol{A}^{\mathrm{T}} (\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y}), \tag{2}$$

$$\boldsymbol{x}^{(t+1)} = S_\lambda(\boldsymbol{z}^{(t)}), \quad (t = 1, 2, \ldots, T), \tag{3}$$

where the parameter $\eta$ is the step size, $\lambda$ is the thresholding parameter and set to $\lambda = \tau\eta$, $T$ denotes the maximum number of iterations, and $\boldsymbol{x}^{(1)} = \boldsymbol{0}$. Moreover, $S_\lambda(\boldsymbol{z}^{(t)})$ is called the soft-thresholding function and is expressed as

$$S_\lambda(x) = \begin{cases} x - \lambda & (x \geq \lambda) \\ 0 & (-\lambda < x < \lambda) \\ x + \lambda & (x \leq -\lambda). \end{cases} \tag{4}$$

The soft-thresholding function $S_\lambda(\boldsymbol{z}^{(t)})$ is applied element-wise when operating on vectors. It is known that setting the step size as $\eta = 1/L$ is optimal in terms of convergence speed, where $L$ is the maximum eigenvalue of the matrix $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$, that is, $L = \lambda_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$.

### B. Deep Unfolding

In this section, we describe deep unfolding [6] used in this paper. Deep unfolding is a method in which learnable parameters are embedded into iterative algorithms and optimized by learning with data. This approach allows the algorithm to adapt to data while preserving the structure of the original iterations. Moreover, it has been reported that deep unfolding can improve convergence speed in many applications [13].

Let $f_{\boldsymbol{\theta}}(\cdot)$ denote the output of an algorithm. Here, $\boldsymbol{\theta}$ represents the set of learnable parameters embedded in each iteration of the algorithm. To train these parameters, a dataset $\mathcal{D} \equiv \{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^D$ is prepared. The learnable parameters $\boldsymbol{\theta}$ are adjusted so as to minimize a loss function between the original signal $\boldsymbol{X}_i$ and the estimate vector $f_{\boldsymbol{\theta}}(\boldsymbol{Y}_i)$.

For many regression problems, the squared error function is used as the loss function. That is given by $L(\boldsymbol{\theta}) \equiv \sum_{(\boldsymbol{X},\boldsymbol{Y}) \in \mathcal{D}} ||f_{\boldsymbol{\theta}}(\boldsymbol{Y}) - \boldsymbol{X}||^2$. Learned parameters $\boldsymbol{\theta}^*$ are obtained through typical training processes of deep learning such as mini-batch learning.

### C. Sketching

In recent signal processing, balancing the high dimensionality of data and the constraints of computational resources has become a major issue. Conventional iterative optimization algorithms perform gradient calculations using all data at every step, so when the number of rows in the observation matrix or the amount of data is large, computational cost and memory usage become bottlenecks.

Against this background, Tang et al. [5] proposed an algorithm that integrates dimensionality reduction known as sketching [1] into optimization methods based on gradient descent methods, enabling fast and structure-preserving estimation. As a result, it is possible to efficiently obtain a solution close to that of the original high-dimensional problem with low computational complexity.

We briefly review the projected gradient method with sketching proposed in [5]. The target optimization problem is the following least squares problem:

$$\min_{\boldsymbol{x} \in \mathcal{K}} f(\boldsymbol{x}), \ f(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{5}$$

where $\mathcal{K} \subset \mathbb{R}^n$ is a convex set imposing structural constraints. A projection operator is defined as

$$\mathcal{P}_{\mathcal{K}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{z} \in \mathbb{R}^n} \left\{ \mathbb{I}_{\mathcal{K}}(\boldsymbol{z}) + \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 \right\}, \tag{6}$$

where $\mathbb{I}_{\mathcal{K}}(\boldsymbol{z})$ is the indicator function for the set $\mathcal{K}$. By using the projection operator, the solution to (5) can be obtained iteratively by the update formula $\boldsymbol{x}^{(t+1)} = \mathcal{P}_{\mathcal{K}} \left( \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)}) \right), (t = 1, 2, \ldots, T)$, where $\eta$ is the step

size. Since $\nabla f(\boldsymbol{x}) = \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$, this can be rewritten as $\boldsymbol{x}^{(t+1)} = \mathcal{P}_{\mathcal{K}}\left(\boldsymbol{x}^{(t)} - \eta \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y})\right)$.

Linear sketching is a dimensionality reduction technique that applies a matrix with low dimension from the left. By introducing sketching, the matrix-vector product operations in $\boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$ used in gradient calculation can be approximated in a lower-dimensional space. A sketching matrix $\boldsymbol{S} \in \mathbb{R}^{l \times m}$ ($l \ll m$) is introduced. The number $l$ of rows is called sketch size. We can consider an alternative least squares problem

$$\min_{\boldsymbol{x} \in \mathcal{K}} \tilde{f}(\boldsymbol{x}), \;\; \tilde{f}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{S}\boldsymbol{y}\|_2^2. \tag{7}$$

Accordingly, the gradient in this case is given by $\nabla \tilde{f}(\boldsymbol{x}) = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})$, and the update formula is replaced by

$$\boldsymbol{x}^{(t+1)} = \mathcal{P}_{\mathcal{K}}\left(\boldsymbol{x}^{(t)} - \eta \boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{y})\right). \tag{8}$$

Tang et al. theoretically demonstrated that if the sketch size $l$ is appropriately selected and algorithms such as GPIS and IHS are used, a solution close to the optimal solution of the original problem can be obtained [1], [14]. For the sketching matrix $\boldsymbol{S}$, Gaussian random matrices, subsampled randomized trigonometric transform sketching matrices, Count Sketch, etc., are often used. By selecting an appropriate $\boldsymbol{S}$, it becomes possible to perform dimensionality reduction while preserving the original problem structure as much as possible and reduce computational cost.

## III. Proposed Method

In this paper, we introduce sketching techniques to ISTA, which is inspired by [5], and propose a hybrid algorithm that combines original gradient update with sketched gradient update. Furthermore, by applying deep unfolding to this algorithm, we make the parameters in each iteration learnable, aiming to improve recovery accuracy while maintaining computational efficiency.

### A. Sketched ISTA

To improve the computational efficiency of ISTA, we apply a sketching technique to matrix $\boldsymbol{A}$ to reduce the computational load of iterative calculations. Specifically, by sketching $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with a random matrix $\boldsymbol{S} \in \mathbb{R}^{l \times m} (l \ll m)$, we transform the observation model as:

$$\boldsymbol{S}\boldsymbol{y} = \boldsymbol{S}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{S}\boldsymbol{w}, \tag{9}$$

and consider the problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{2}\|\boldsymbol{S}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{S}\boldsymbol{y}\|_2^2 + \tau\|\boldsymbol{x}\|_1. \tag{10}$$

This allows *sketched ISTA* to execute the proximal gradient iterations as conventional ISTA, but in a smaller dimension. The iterative formula for sketched ISTA is given by:

$$\boldsymbol{z}^{(t)} = \boldsymbol{x}^{(t)} - \eta \boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}(\boldsymbol{S}\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{S}\boldsymbol{y}), \tag{11}$$

$$\boldsymbol{x}^{(t+1)} = S_\lambda(\boldsymbol{z}^{(t)}), \quad (t = 1, 2, \ldots, T), \tag{12}$$

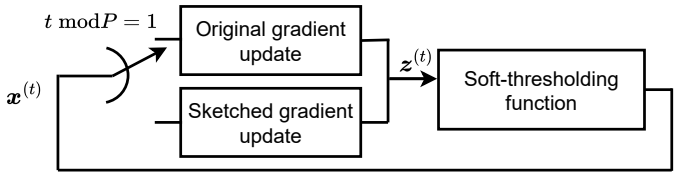


Fig. 1. Algorithm diagram of Periodic Sketched ISTA

where the maximum number of iterations is $T$. The original ISTA includes the matrix-vector product $\boldsymbol{A}\boldsymbol{x}^{(t)}$ which requires $mn$ multiplications and $m(n-1)$ additions. By introducing the sketching matrix $\boldsymbol{S}$, these factors for the product $(\boldsymbol{S}\boldsymbol{A})\boldsymbol{x}^{(t)}$ are reduced to $ln$ multiplications and $l(n-1)$ additions.

We adopt the Gaussian sketch [5] in this paper. Specifically, each element of the sketching matrix $\boldsymbol{S} \in \mathbb{R}^{l \times m}$ is independently drawn from a Gaussian distribution with zero mean and variance $1/l$, that is, $S_{ij} \sim \mathcal{N}(0, 1/l)$.

### B. Periodic Sketched ISTA

In this paper, we propose *PSISTA* (Periodic Sketched ISTA) as a novel algorithm that periodically switches between standard ISTA and sketched ISTA. Figure 1 illustrates the iterative processes of the proposed algorithm.

Specifically, the iterative process of PSISTA is defined as follows for the $t$-th iteration:

$$\mathrm{OGU}(\boldsymbol{x}) = \boldsymbol{x} - \eta \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}), \tag{13}$$

$$\mathrm{SGU}(\boldsymbol{x}) = \boldsymbol{x} - \eta \boldsymbol{A}^{\mathrm{T}}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{S}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}), \tag{14}$$

$$\boldsymbol{z}^{(t)} = \begin{cases} \mathrm{OGU}(\boldsymbol{x}^{(t)}) & \text{if } t \bmod P = 1, \\ \mathrm{SGU}(\boldsymbol{x}^{(t)}) & \text{otherwise}, \end{cases} \tag{15}$$

$$\boldsymbol{x}^{(t+1)} = S_\lambda(\boldsymbol{z}^{(t)}), \tag{16}$$

where OGU and SGU stand for Original Gradient Update (2) and Sketched Gradient Update (11), respectively.

A positive integer parameter $P$ is named period and used to control the period of introducing the original gradient update. Thus, it becomes possible to leverage the advantages of the lower computational cost of sketched gradient update while suppressing the degradation of estimation accuracy by periodically inserting original gradient update. Note that original gradient update is applied at the beginning of each period.

The purpose of this method is to balance computational efficiency and estimation accuracy by integrating the low computational cost of the sketched ISTA with the high representational capability of standard ISTA.

### C. PSISTA with Deep Unfolding

The PSISTA has the potential to flexibly control a trade-off between computational efficiency and recovery accuracy but there is concern that the performance will monotonically decrease with the number of times sketching is introduced. In this paper, we apply deep unfolding to the proposed PSISTA method to improve performance through data-driven hyperparameter learning. By unfolding each iterative step of PSISTA

in the time direction, we extend it to a trainable structure. In each layer, parameters such as step size are introduced as learnable parameters.

We set the step size $\eta$ and thresholding parameter $\lambda$ in PSISTA as the learnable parameters. When the maximum number of iterations of the algorithm is $T$, each parameter at the $t$-th iteration can be expressed as $\eta^{(t)}, \lambda^{(t)}$. We learn $\boldsymbol{\eta} \equiv (\eta^{(1)}, \eta^{(2)}, \ldots, \eta^{(T)})$ and $\boldsymbol{\lambda} \equiv (\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(T)})$ using deep unfolding. The learned parameters are used during performance measurement. We name the proposed PSISTA to which deep unfolding is applied as *DU-PSISTA* (Deep Unfolded-Periodic Sketched ISTA). If these parameters are trained properly, it is expected to design an algorithm that achieves performance close to that of the original ISTA before dimensionality reduction by sketching, while maintaining high computational efficiency.

### D. Computational Complexity of Sketched ISTA

This section evaluates the computational complexity for the standard ISTA, sketched ISTA, and PSISTA. The computational complexity here denotes the total number of additions and multiplications required for $T$ iterations of each algorithm. Note that we do not include some pre-computations in the computational complexity; the derivation of the sketched matrix $\boldsymbol{SA}$ and vector $\boldsymbol{Sy}$, and deep unfolding because these are computed only once before the algorithm is executed and fixed during the execution.

We first consider the computational complexity per iteration for both ISTA and sketched ISTA. In both cases, matrix-vector products included in gradient computation account for the largest proportion of addition and multiplication counts. Specifically, for the standard ISTA, $\boldsymbol{A}\boldsymbol{x}^{(t)}$ is the product of $m \times n$ matrix and $n \times 1$ vector so that it requires $mn$ multiplications and $m(n-1)$ additions. For sketched ISTA, these can be reduced to $ln$ multiplications and $l(n-1)$ additions. By designing the sketching such that $l \ll m$, it becomes possible to significantly reduce the computational complexity per iteration overall computational. The complexity of the soft-thresholding operation can be $n$ additions. In summary, per-iteration computational complexities of ISTA, $O_{\text{ISTA}}$, and sketched ISTA, $O_{\text{Sketch}}$, are given by:

$$O_{\text{ISTA}} = 4mn + 3n - m, \tag{17}$$

$$O_{\text{Sketch}} = 4ln + 3n - l. \tag{18}$$

The total complexity of $T$ iterations of the standard ISTA is $C_{\text{ISTA}} = T \cdot O_{\text{ISTA}}$.

We next consider the complexity of the proposed PSISTA, which includes the original ISTA-based update and sketched ISTA-based update. The number of executions of the original ISTA-based update, $N_{\text{ISTA}}$, and sketched ISTA-based update, $N_{\text{Sketch}}$, are given by:

$$N_{\text{ISTA}} = \left\lfloor \frac{T}{P} \right\rfloor + \begin{cases} 1 & \text{if } T \bmod P \geq 1 \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

$$N_{\text{Sketch}} = T - N_{\text{ISTA}}, \tag{20}$$

where $P$ is the period. From the above, the total computational complexity $C_{\text{PSISTA}}$ of PSISTA is:

$$\begin{aligned} C_{\text{PSISTA}} &= N_{\text{ISTA}} \cdot O_{\text{ISTA}} + N_{\text{Sketch}} \cdot O_{\text{Sketch}} \\ &= N_{\text{ISTA}}(4mn + 3n - m) + (T - N_{\text{ISTA}})(4ln + 3n - l). \end{aligned} \tag{21}$$

Introducing sketching even once during the iterative process reduces the overall computational complexity compared to standard ISTA.

## IV. Computer Experiments

### A. Overview of Experiments

In this section, we describe the experimental settings of the computer experiments. We compare the proposed DU-PSISTA with the conventional DU-ISTA. We evaluate the performance of the standard ISTA, DU-ISTA, sketched ISTA, sketched ISTA with deep unfolding (DU-Sketched ISTA), and DU-PSISTA. The observation matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and the sketching matrix $\boldsymbol{S} \in \mathbb{R}^{l \times m}$ were generated with elements drawn from normal distributions with means 0 and variances 1 and $1/l$, respectively. Each element of the original signal $\boldsymbol{x} \in \mathbb{R}^n$ follows i.i.d. Bernoulli-Gaussian distribution where it is 0 with probability 0.95 and follows $\mathcal{N}(0,1)$ with probability 0.05. We set the noise variance to $\sigma^2 = 0.01$. The performance was evaluated with large size system $(m, n) = (1024, 512)$ and small size system $(256, 128)$.

The step size $\eta$ and thresholding parameter $\lambda$ of ISTA were both set to $1/\lambda_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$. The initial values for the learnable parameters $\eta^{(t)}$ and $\lambda^{(t)}$ were also set to this value. Performance was evaluated by MSE (mean squared error) over 50 different $\boldsymbol{A}$,$\boldsymbol{S}$ pairs, each with 50 samples of signals $(\boldsymbol{x}, \boldsymbol{y})$. We used incremental learning [4] and mini-batch learning for deep unfolding. The mini-batch size and the number of inner loop iterations were set to 50. In each inner loop, the matrices $\boldsymbol{S}$ and $\boldsymbol{A}$ were regenerated, and mini-batches were constructed accordingly for training. The mean squared loss function was used. Parameter updates were performed using the Adam optimizer.

### B. Verification of the Effect of Period

In this section, we investigate the impact of varying the period $P$ in DU-PSISTA on recovery performance. We set $l = 256, 64$ for the large and small systems, respectively. The period $P$ was varied among 2, 3, 5, and 8 to evaluate its impact on recovery performance.

First, the computational complexity for each system is calculated from (21) and summarized in Tables I and II. From the tables, it can be observed that the computational complexity decreases as the period $P$ increases and the sketch size $l$ decreases.

Figures 2 and 3 present the MSE performance of each system with varying the period $P$. In each figure, the blue line is the baseline DU-ISTA and the light blue area above the line indicates the range within 2.0 times the MSE of DU-ISTA. As shown in the figures, the recovery accuracy deteriorates with increasing $P$, and for larger $P$, the MSE exceeds 2.0 times that

TABLE I
TOTAL COMPUTATIONAL COMPLEXITY $C_{\text{PSISTA}}$ FOR EACH SKETCH SIZE $l$ AND PERIOD $P$ FOR LARGE SYSTEM ($n = 1024$, $m = 512$, $T = 40$)

| $P\backslash l$ | 256 | 128 | 64 | 32 |
|---|---|---|---|---|
| 1 ($C_{\text{ISTA}}$) | 83,988,480 (100.0%) | 83,988,480 (100.0%) | 83,988,480 (100.0%) | 83,988,480 (100.0%) |
| 2 | 63,022,080 (75.0%) | 52,538,880 (62.6%) | 47,297,280 (56.3%) | 44,676,480 (53.2%) |
| 3 | 56,732,160 (67.5%) | 43,104,000 (51.3%) | 36,289,920 (43.2%) | 32,882,880 (39.2%) |
| 5 | 50,442,240 (60.1%) | 33,669,120 (40.1%) | 25,282,560 (30.1%) | 21,089,280 (25.1%) |
| 8 | 47,297,280 (56.3%) | 28,951,680 (34.5%) | 19,778,880 (23.5%) | 15,192,480 (18.1%) |

TABLE II
TOTAL COMPUTATIONAL COMPLEXITY $C_{\text{PSISTA}}$ FOR EACH SKETCH SIZE $l$ AND PERIOD $P$ FOR SMALL SYSTEM ($n = 256$, $m = 128$, $T = 40$)

| $P\backslash l$ | 64 | 32 | 16 | 8 |
|---|---|---|---|---|
| 1 ($C_{\text{ISTA}}$) | 5,268,480 (100.0%) | 5,268,480 (100.0%) | 5,268,480 (100.0%) | 5,268,480 (100.0%) |
| 2 | 3,959,040 (75.1%) | 3,304,320 (62.7%) | 2,976,960 (56.5%) | 2,813,280 (53.4%) |
| 3 | 3,566,208 (67.7%) | 2,715,072 (51.5%) | 2,289,504 (43.5%) | 2,076,720 (39.4%) |
| 5 | 3,173,376 (60.2%) | 2,125,824 (40.3%) | 1,602,048 (30.4%) | 1,340,160 (25.4%) |
| 8 | 2,976,960 (56.5%) | 1,831,200 (34.8%) | 1,258,320 (23.9%) | 971,880 (18.4%) |

of DU-ISTA, which is considered to be due to the accumulation of approximation errors caused by more frequent use of the sketching matrix. On the other hand, when $P = 2$, DU-PSISTA achieves accuracy comparable to DU-ISTA. These results indicate a trade-off between computational efficiency and estimation accuracy, and suggest that a smaller period $P$ is preferable for maintaining high recovery performance.

In both system sizes, DU-PSISTA demonstrated excellent performance in reducing MSE and achieving fast convergence when the period $P$ was set to a small value, such as 2 or 3. Specifically, the proposed method can reduce the computational complexity to 68–75% while keeping MSE within or near the range of 2.0 times of that of DU-ISTA. It indicates that the proposed method achieves a good balance between efficiency and accuracy.

### C. Verification of the Effect of Sketch Size

In this section, we investigate the impact of varying sketch size $l$ in DU-PSISTA on recovery performance. The sketch size $l$ was varied among $\{256, 128, 64, 32, 16\}$ for the large system and $\{64, 32, 16, 8\}$ for the small system. We set $P = 2$ for both systems.

Figures 4 and 5 show the recovery accuracy for the large and small systems, respectively. The blue line is the baseline DU-ISTA, and the blue area above the line indicates the range within 2.0 times the MSE of DU-ISTA. For the large system in Fig. 4, it can be observed that when $l = 256$, the MSE of DU-PSISTA remains within 2.0 times that of DU-ISTA across all iterations $T$. However, for $l = 128$ and smaller, the MSE increases as $l$ decreases, indicating a degradation in recovery

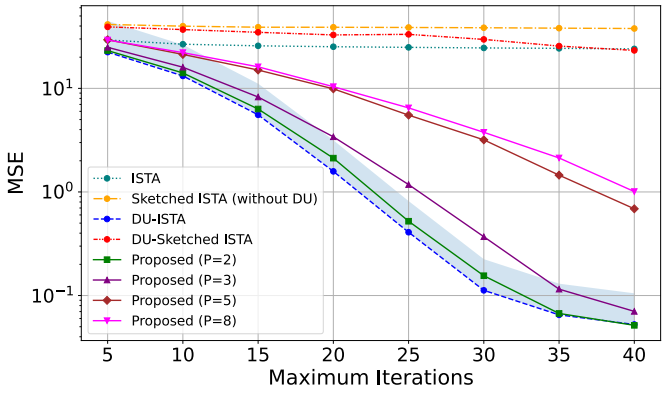


Fig. 2. Comparison of MSE performance for different period $P$ in DU-PSISTA for large system ($n = 1024, m = 512, l = 256$).

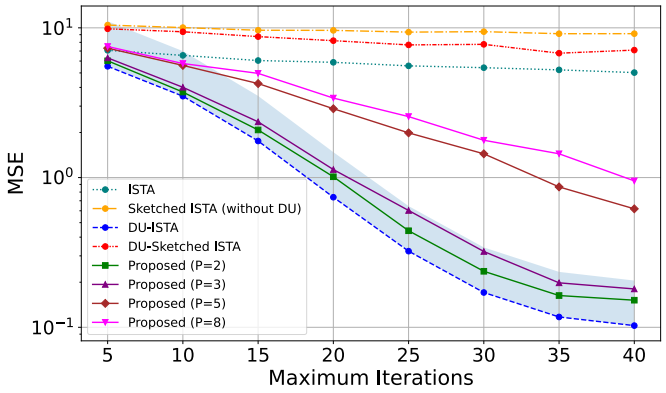


Fig. 3. Comparison of MSE performance for different period $P$ in DU-PSISTA for small system ($n = 256, m = 128, l = 64$).

accuracy. Similarly for the small system in Fig. 5, the case of $l = 64$ is within the range of 2.0 times the MSE of DU-ISTA but the other cases are not.

From these results, it is evident that reducing the sketch size $l$ leads to a decline in estimation accuracy for both system sizes. This is likely because the reduced dimensionality of the feature space after sketching limits the ability to sufficiently capture essential signal features. However, by varying $l$ from 256 to 128, 64, 32, and further to 16, the computational complexity is significantly reduced, resulting in improved computational efficiency. Therefore, the proposed DU-PSISTA demonstrates a good balance between accuracy and computational cost when the number of rows $l$ is relatively large, such as 256 for the larger system and 64 for the smaller system.

## V. CONCLUSION

In this paper, we proposed DU-PSISTA as a new method aiming to achieve both computational efficiency and estimation accuracy for the recovery problem of high-dimensional and sparse signals. The proposed algorithm is characterized by the ability to reduce computational cost while maintaining accuracy by periodically switching between original and sketched

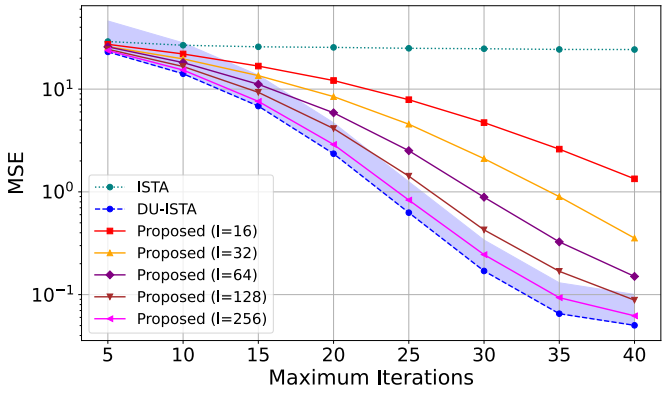


Fig. 4. Comparison of MSE performance for different sketch size $l$ in DU-PSISTA for large system ($n = 1024, m = 512, P = 2$).

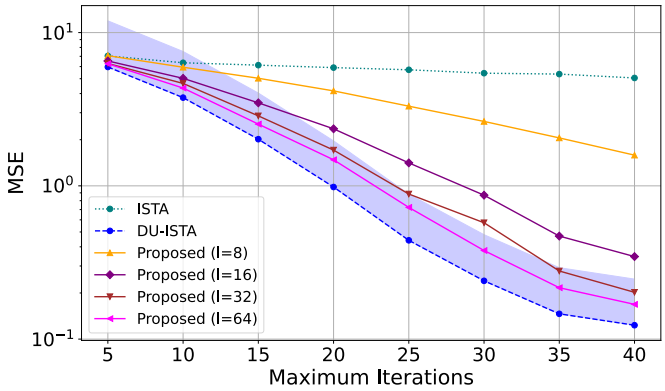


Fig. 5. Comparison of MSE performance for different sketch size $l$ in DU-PSISTA for small system ($n = 256, m = 128, P = 2$).

gradient update. Furthermore, by applying deep unfolding to this method, we extended it so that step sizes and thresholding parameters in each iteration can be learned, enabling data-driven optimization of hyperparameters that previously had to be manually adjusted.

The effectiveness of the proposed DU-PSISTA was verified through computer experiments conducted under various system sizes, sketch sizes, and period settings. As a result, it was confirmed that, especially with settings that appropriately balance original and sketched gradient updates, it is possible to reduce computational complexity by 32 % while maintaining recovery performance comparable to the conventional DU-ISTA.

As future work, we would like to incorporate other structured sketching matrices including fast transform or sparsification to further reduce computational complexity. It is worth examining the trade-off between the complexity and recovery performance in detail. Developing into a more adaptive method that can reduce training cost of deep unfolding is one of the other challenging directions for improving performance. It would also be important to verify the scalability of the proposed method with respect to system size.

## References


[1] D. P. Woodruff, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[2] Y. C. Eldar and G. Kutyniok, *Compressed sensing: theory and applications*. Cambridge university press, 2012.

[3] L. Liu, E. G. Larsson, W. Yu, P. Popovski, C. Stefanovic, and E. De Carvalho, "Sparse signal processing for grant-free massive connectivity: A future paradigm for random access protocols in the internet of things," *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 88–99, 2018.

[4] D. Ito, S. Takabe, and T. Wadayama, "Trainable ista for sparse signal recovery," *IEEE Trans. Signal Process.*, vol. 67, no. 12, pp. 3113–3125, 2019.

[5] J. Tang, M. Golbabaee, and M. Davies, "Exploiting the structure via sketched gradient algorithms," in *Proc. 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2017, pp. 1305–1309.

[6] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint, arXiv:1409.2574*, 2014.

[7] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, IEEE, 2019, pp. 266–271.

[8] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th international conference on international conference on machine learning*, 2010, pp. 399–406.

[9] H. Song, X. You, C. Zhang, and C. Studer, "Soft-output joint channel estimation and data detection using deep unfolding," in *Proc. 2021 IEEE Information Theory Workshop (ITW)*, IEEE, 2021, pp. 1–5.

[10] D. You, J. Xie, and J. Zhang, "Ista-net++: Flexible deep unfolding network for compressive sensing," in *Proc. 2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, pp. 1–6.

[11] K. Hayashi, M. Nagahara, and T. Tanaka, "A user's guide to compressed sensing for communications systems," *IEICE Trans. Commun.*, vol. 96, no. 3, pp. 685–712, 2013.

[12] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Ser. B Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[13] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Trans. Signal Inf. Process.*, vol. 3, no. e2, 2014.

[14] M. Pilanci and M. J. Wainwright, "Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares," *Journal of Machine Learning Research*, vol. 17, no. 53, pp. 1–38, 2016.