Model-Based Deep Learning for One-Bit Compressive Sensing

Shahin Khobahi*, *Student Member, IEEE,* and Mojtaba Soltanalian, *Senior Member, IEEE*

***Abstract*—In this work, we consider the problem of one-bit deep compressive sensing from both a system design and a signal recovery perspective. In particular, we develop hybrid model-based deep learning architectures based on the deep unfolding methodology. We further interpret the overall data-acquisition and signal recovery modules as an auto-encoder structure allowing for learning task-specific sensing matrix, quantization thresholds, as well as the latent-parameters of iterative first-order optimization algorithms specifically designed for the problem of one-bit sparse signal recovery. The proposed model-based deep architectures have the ability to adaptively learn the proper quantization thresholds, paving the way for amplitude recovery in one-bit compressive sensing. We further show that the proposed methodology implicitly learns task-specific sensing matrices with very low coherence, which is highly desirable in a compressive sensing setting. Due to the model-based nature of the proposed deep architecture, it enjoys from the interpretability and versatility of model-based techniques as well as benefiting from the expressive power of data-driven methods. Specifically, owing to its model-based nature, it has far fewer parameters and requires far less samples for training as compared to black-box machine learning models. Our results demonstrate a significant improvement compared to state-of-the-art algorithms.**

***Index Terms*—Compressive sensing, low-resolution signal processing, one-bit quantization, deep unfolding, deep neural networks, model-based deep learning, autoencoders**

I. Introduction

In the past two decades, compressive sensing (CS) has shown significant potential in enhancing sensing and recovery performance in signal processing, occasionally with simpler hardware, and thus, has attracted noteworthy attention among researchers. CS is a method of signal acquisition which ensures the exact or almost exact reconstruction of certain classes of signals using far less number of samples than what is needed in the Nyquist sampling regime [1]—where the signals are typically reconstructed by finding the sparsest solution of an under-determined system of equations using various available means.

In a practical setting, each measurement is to be digitized into finite-precision values for further processing and storage purposes, which inevitably introduces a quantization error. This error is usually modeled as an additive Gaussian noise, independent of the input source signal; an approach that does not perform well in extreme cases of quantization. One-bit CS is one such extreme case where the quantizer is a simple sign comparator and each measurement is represented using only one bit information $r \in \{\pm 1\}$ [2]–[6]. One-bit quantizers are not only low-cost and low-power hardware components, but also much faster than traditional scalar quantizers, accompanied by great reduction in the complexity of hardware implementation. Several algorithms have been introduced in the literature for efficient reconstruction of sparse signals in one-bit CS scenarios (e.g., see [2]–[7] and the references therein). A detailed discussion of such algorithms is provided in Sec II.

Notation: We use bold lowercase letters for vectors and bold uppercase letters for matrices. $(\cdot)^T$, and $(\cdot)^H$ denote the vector/matrix transpose, and the Hermitian transpose, respectively. $\mathbf{1}$ and $\mathbf{0}$ are the all-one and all-zero vectors. $\|\boldsymbol{x}\|_n$ denotes the ℓ_n-norm of the vector $\boldsymbol{x}$ defined as $(\Sigma_k |\boldsymbol{x}(k)|^n)^{\frac{1}{n}}$. $\boldsymbol{x}(i)$ denotes the i-th element of the vector $\boldsymbol{x}$ and $\boldsymbol{A}(i,j)$ denotes the ij-th element of the matrix $\boldsymbol{A}$. Diag$(\boldsymbol{x})$ denotes the diagonal matrix formed by the entries of the vector argument $\boldsymbol{x}$. The operator $\succeq$ denotes the element-wise vector inequality operator.

A. Background and Relevant Prior Art

One-bit compressive sensing is mainly concerned with the following data-acquisition model:

$$\boldsymbol{r} = \text{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes a K-sparse source signal, $\boldsymbol{\Phi} \in \mathbb{R}^{m \times n}$ is the sensing matrix, and $\boldsymbol{b} \in \mathbb{R}^n$ denotes the quantization thresholds vector. In addition to the mentioned advantages of using one-bit ADCs for data-acquisition purposes, the use of one-bit information offers increased robustness to undesirable non-linearities in the data-acquition process. Furthermore, there exists strong empirical evidence that recovering a sparse source signal from only one-bit measurement can outperform its multi-bit CS counterpart [4], [8].

In this works, we seek to take a deeper look at the one-bit CS arena from both a *system design* perspective related to the design of task-specific high-quality sensing matrices $\boldsymbol{\Phi}$ and one-bit quantization systems (i.e., designing task-specific qunatization thresholds $\boldsymbol{b}$), and the development of accurate and efficient *task-specific signal reconstruction techniques* for one-bit signal recovery in a CS setting. In the following, we go into the details of each aspect and give an overview of the existing methodologies.

Sensing Matrix Design. From a system design point of view, the most relevant factors to be taken into account are the sensing matrix and the one-bit quantization thresholds used for data-acquisition. In particular, two key requirements of a CS-based signal reconstruction algorithm are the *sparsity* of

This work was supported in part by U.S. National Science Foundation Grant CCF-1704401.

S. Khobahi and M. Soltanalian are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607 (**Corresponding author: skhoba2@uic.edu*).

the underlying signal of interest and the *incoherence* which is mainly related to the underlying sensing matrix $\mathbf{\Phi}$ employed at time of the acquisition. It can be shown that one can recover the underlying sparse signal from the linear compressive measurements with overwhelming high probability if the sensing matrix have a low coherence property [9]. Accordingly, such performance guarantees is based upon the Restricted Isometric Property (RIP) which is at the heart of CS theory. Specifically, for a sensing matrix $\mathbf{\Phi}$, and two sparse vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, the RIP can be stated as follows:

$$(1-\gamma)\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \leq \|\mathbf{\Phi}(\boldsymbol{x}-\boldsymbol{y})\|_2^2 \leq (1+\gamma)\|\boldsymbol{x}-\boldsymbol{y}\|_2^2,$$

where $\gamma \in (0,1)$. In short, for a matrix $\mathbf{\Phi}$ satisfying the RIP condition, we have that the distance of any two vectors (signals) is maintained up to the bounding factors $\{1-\delta, 1+\delta\}$ after applying the transformation $\mathbf{\Phi}$. Accordingly, one can perform an almost perfect reconstruction of an sparse signal with high probability when the RIP condition is met by the measurement matrix [1]. Nonetheless, certifying the RIP condition for a given matrix is a difficult task in general and it has been shown to be an NP-hard problem [10]. Consequently, there exist two main strategies in employing sensing matrices for CS in the literature: The first one considers the deployment of random matrices at the time of acquisition, while the other approach makes use of task-specific deterministic sensing matrices. Having said that, in a general CS setting, the works of [11], [12] have shown that random sensing matrices (e.g., Gaussian matrices) satisfy the RIP condition with a very high probability, hence providing mathematical ground-work for robust sparse signal recovery. Taking this into consideration, many signal reconstruction techniques, including but not limited to basis pursuit techniques, can be shown to provably recover a sparse signal when random matrices are employed [13]. Nonetheless, using random matrices is not applicable in many applications due to the imposed randomness in the measurement system [13]. More importantly, in many applications the sensing matrix must be designed in a fashion to account for the intricate physical model of the system and the measurement model. In such applications, one can resort to a deterministic design (in contrast to a purely random linear measurement) of the sensing matrix to accommodate for the measurement medium of interest. However, as previously mentioned, it is a very difficult task to verify the RIP condition for a matrix, and it cannot be easily used as an objective for a deterministic design of sensing matrices. Note that there exist several alternative measures for quantifying the quality of a sensing matrix. The most notable one and widely used metric is the *mutual coherence* [14], which is a mathematically tractable alternative metric for measuring the incoherence required by the compressed sensing theory and the success of many basis pursuit algorithms. Specifically, let $\bar{\mathbf{\Phi}}$ denote the column-normalized version of the sensing matrix $\mathbf{\Phi}$ and define the Gram matrix $\boldsymbol{M} \triangleq \bar{\mathbf{\Phi}}^H\bar{\mathbf{\Phi}}$. Then, the mutual coherence of a sensing matrix $\mathbf{\Phi}$ is given by

$$\mu(\mathbf{\Phi}) = \max_{i\neq j} \ |\boldsymbol{M}(i,j)|. \tag{2}$$

Furthermore, the off-diagonal entries $\{|\boldsymbol{M}(i,j)|\}_{i\neq j}$ represent the coherence coefficients of the sensing matrix $\mathbf{\Phi}$. Briefly speaking, the mutual coherence factor $\mu(\mathbf{\Phi})$ provides a measure of the worst-case similarity between the columns of $\mathbf{\Phi}$, and furthermore, a high mutual coherence results in a significant degradation in the performance of basis pursuit signal recovery techniques [15]. Hence, it is highly desirable to have a sensing matrix with low mutual coherence, corresponding to a Gram matrix $\boldsymbol{M}$ close to identity $\boldsymbol{I}$.

Designing task-specific and deterministic sensing matrices with low-coherence is an active research area in various fields such as coding and communication [16], quantum signal processing [17], machine learning [18], radar signal processing [19], among many others. Perhaps, one of the most interesting of these applications is the one-bit compressive sensing area. To the best of our knowledge, there exist no existing work in the literature that studies the design of task-specific deterministic sensing matrices in a one-bit CS setting and its advantages over using random matrices. Hence, *one of the main motivations of this work is to address this issue and to propose a unified framework that allows for designing task-specific deterministic sensing matrices that can handle the severe non-linearity imposed by the one-bit quantization at the time of acquisition, which further allows for a significant improvement of the signal reconstruction accuracy at the time of inference.* The proposed methodology does not require an explicit optimization over the mutual coherence which may be difficult to handle from an optimization point of view. Indeed, we empirically show that the proposed methodology *implicitly* learns task-specific sensing matrices with *very low mutual coherence* leading significantly enhancing the signal reconstruction accuracy.

We conclude this part by emphasizing that although it is common to consider the problem of one-bit CS (or CS in general) and the design of task-specific sensing matrices for such systems from a purely mathematical point-of-view, the development of CS and one-bit CS-based hardware data acquisition systems is still a great challenge in practice. The successful implementation of such systems might require further integration of theory and practice, considering various limitations of physical hardware.

Low-Resolution Quantization and Signal Recovery. The other factor that affects the performance of the signal recovery in a one-bit CS setting is the choice of the quantization thresholds. There exist two strategies to be undertaken regarding the qunatization thresholds: The first one is concerned with setting the quantization thresholds to zero while the other way is to consider non-zero thresholds. In both settings, the current one-bit CS recovery algorithms typically exploit the *consistency principle*, which represents the fact that the element-wise product of the sparse signal and the corresponding measurement is always positive [2], i.e. $\boldsymbol{r} \odot (\mathbf{\Phi}\boldsymbol{x} - \boldsymbol{b}) \succeq 0$. However, most of the existing literature on one-bit CS considers zero-level one-bit quantization thresholds (i.e., $\boldsymbol{b} = \mathbf{0}$) leading to a total loss of amplitude information during the data-acquisition process. Hence, by comparing the signal level with zero, one can only recover the direction of the source signal, i.e. $\boldsymbol{x}/\|\boldsymbol{x}\|_2$, and not the amplitude information $\boldsymbol{x}$. In its most general form, any solution $\boldsymbol{x}^*$ to the one-bit CS problem should: (i) satisfy the sparsity condition, i.e. $\|\boldsymbol{x}^*\|_0 \leq K$ with $K = \|\boldsymbol{x}\|_0$, and

(ii) achieve consistency, i.e. $\boldsymbol{r}\odot(\boldsymbol{\Phi x}^*-\boldsymbol{b})\succeq 0$. As mentioned above, most of the existing literature on the problem of one-bit CS recovery problem considers the case of $\boldsymbol{b}=\boldsymbol{0}$. In such a case, the solution to the one-bit CS problem can be expressed as:

$$\boldsymbol{x}^*=\underset{\boldsymbol{x}}{\operatorname{argmin}}\ \|\boldsymbol{x}\|_0 \ \text{ s.t. } \ \boldsymbol{r}=\operatorname{sign}(\boldsymbol{\Phi x}).$$

The above program is NP-hard and mathematically intractable [4]. However, there exist several powerful iterative algorithms to find $\boldsymbol{x}^*$ (for the case of $\boldsymbol{b}=\boldsymbol{0}$) that rely on a relaxation of the ℓ_0-norm to its convex hull (i.e., using ℓ_1-norm in lieu of ℓ_0-norm) to obtain an estimate of the support of the true source signal by restricting the feasible solutions to the unit-sphere, i.e. $\|\boldsymbol{x}\|_2=1$.

The most notable works which considers a zero quantization thresholding scheme are as follows. In [2], the authors assume a zero-level quantization threshold and propose an iterative algorithm called *renormalized fixed point iteration* (RFPI) where a convex barrier function is used to enforce the consistency principle (as a regularization term in the objective function). A detailed analysis of the RFPI algorithm is provided in Sec. II. It is worth mentioning that in a traditional CS setting, one consider the under-sampled measurements (i.e., $m<n$), however, the *over-sampling* regime is beneficial and of paramount interest in a one-bit CS setting in that the use of one-bit ADCs provide a cheap and fast way to acquire measurements and to potentially go beyond the limitations of the traditional CS methods. Another such reconstruction algorithm can be found in [3], referred to as *restricted step shrinkage* (RSS), for which a nonlinear barrier function is used as the regularizer to enforce the consistency principle. Compared to RFPI algorithm, RSS has three important advantages: provable convergence, improved consistency, and feasible performance [20]. Ref. [4] introduces a penalty-based robust recovery algorithm, called *binary iterative hard thresholding* (BIHT), in order to enforce the consistency principle. Contrary to RFPI algorithm, BIHT exploits the knowledge of the sparsity level of the signal as input, and was shown to be more robust to outliers and have a superior performance than that of the RFPI method in some cases (at the cost of knowing the sparsity level of the source signal a priori). Both RFPI and BIHT, however, only consider a zero-level quantization threshold, as a result, the amplitude information is lost due to comparing the acquired signal with zero. In [5] and [6], authors proposed modified versions of RFPI and BIHT, referred to as *noise-adaptive renormalized fixed point iteration* (NARFPI) and *adaptive outlier pursuit with sign flips* (AOP-f), that are more robust against bit flips in the measurement vector (that occur due to the presence of noise). In [21], the authors lay the ground work for a theoretical analysis of noisy one-bit CS problem based on a convex programming approach for the problem of one-bit sparse signal recovery in a noisy setting.

There exist limited work on employing non-zero quantization thresholds in a one-bit CS setting. Recently, the authors in [8] considered the problem of one-bit CS signal reconstruction in a non-zero quantization thresholds setting that enables the recovery of the norm of the source signal, i.e. recovering $\|\boldsymbol{x}\|_2$. However, the proposed method in [8] still fails to accurately recover the amplitude information of the source signal, and does not offer a straight-forward method to design the quantization thresholds. Although the one-bit CS has a deterministic system model, the authors in [22] consider a non-zero quantization scheme and provide a Bayesian formulation of the problem upon which a generalized approximate message passing (GAMP) algorithm is used for signal recovery purposes. Furthermore, non-zero quantization thresholds for one-bit compressive systems has been adopted in other fields such as one-bit compressive radar systems [19].

In light of the above, it is of paramount importance to develop *computationally efficient* one-bit CS models that can incorporate non-zero quantization thresholds to allow for recovering the amplitude information. Additionally, the vast literature on the one-bit CS recovery problem, does not yet tap into the potential of the available data at hand (to improve the performance recovery). One can significantly benefit from a methodology that can facilitate not only incorporation of the domain knowledge on the problem (i.e., being model-driven), but also the available data at hand to go beyond the performance of the traditional sparsity aware signal processing techniques in a one-bit CS scenario.

There has recently been a high demand for developing effective real-time signal processing algorithms that use the data to achieve improved performance [23]–[28]. In particular, the data-driven approaches relying on deep neural architectures such as convolutional neural networks [23], deep fully connected networks [24], stacked denoising autoencoders [25], and generative adversarial networks [29] have been studied for sparse signal recovery in generic quantized CS settings. we note that, parameterized mathematical models discussed above play a central role in understanding and design of large-scale information systems and signal processing methods. However, they often cannot take into account the intricate interactions innate to such systems. On the contrary, purely data-driven approaches, and specifically deep learning techniques, do not need explicit mathematical models for data generation and have a wider applicability at the cost of interpretability. The main advantage of the deep learning-based approach is that it employs several non-linear transformations to obtain an abstract representation of the underlying data. Data-driven approaches, on the other hand, lack the interpretability and trustability that comes with model-based signal processing. They are particularly prone to be questioned further, or at least not fully trusted by the users, especially in critical applications. Furthermore, the deterministic deep architectures are generic and it is unclear how to incorporate the existing knowledge on the problem in the processing stage. *The advantages associated with both model-based and data-driven methods show the need for developing frameworks that bridge the gap between the two approaches.*

The recent advent of the *deep unfolding framework* [30]–[35] and the corresponding deep unfolding networks (DUNs) has paved the way for a game-changing fusion of models and well-established signal processing approaches with data-driven architectures. In this way, we not only exploit the vast amounts of available data, but also integrate the prior knowledge of the system model in the processing stage. Deep unfolding relies on the establishment of an optimization or inference iterative algorithm, whose iterations are then *unfolded* into the layers of a deep network, where each layer is designed to resemble one iteration of the optimization/inference algorithm. The resulting hybrid method benefits from low computational cost (in execution stage) of deep neural networks, and at the same time, from the versatility and reliability of model-based methods; thus, appears to be an excellent tool in real-time signal processing applications due to the smaller degrees of freedom required for training and execution (afforded by integration of the problem-level reasoning, or the *model*). A detailed analysis of the deep unfolding methodology for the problem of one-bit CS is provided in Sec. III.

B. Contributions of the Paper

In this paper, we propose a novel hybrid model-based and data-driven methodology (based on DUNs) that addresses the drawbacks of both purely model-based (such as the discussed RFPI and BIHT algorithm) and purely data-driven approaches. The resulting methodology is far less data-hungry and assumes a slight over-parametrization of the system model as opposed to traditional deep learning techniques with extremely large number of variables to be learned. In particular, the proposed method seeks to bridge the gap between the data-driven and model-based approaches in the one-bit CS paradigm, resulting in a specialized architecture for the purpose of sparse signal recovery from one-bit measurements. In particular, the proposed methodology allows for learning task-specific sensing matrices with very low mutual coherence factor, as well as learning data-specific quantization thresholds. Furthermore, we propose a novel model-based interpretable deep learning model for sparse signal recovery in a one-bit CS setting and show that the proposed methodology outperforms the existing state-of-the-art methodologies in the area of one-bit CS. The proposed framework can be seen as a unification of system design and signal recovery techniques, allowing for a joint optimization of all system parameters. The contributions of this paper can be summarized as follows:

- We propose a novel hybrid model-based and data-driven one-bit compressive autoencoding (AE) methodology that can deal with the optimization of the sensing matrix $\boldsymbol{\Phi}$ (learning task-specific deterministic sensing matrices), the one-bit quantization thresholds $\boldsymbol{b}$, and the latent-variables of the decoder module according to the underlying distribution of the source signal. Hence, such a methodology allows for quick adaptation to new data distributions and environments.
- To the best of our knowledge, this is the first attempt in the one-bit CS paradigm that allows for joint optimization of the quantization thresholds and sensing matrix, also facilitating the *recovery of the amplitude information of the source signal*. We show that by using the proposed AEs, one can significantly improve upon existing iterative algorithms and gain much higher accuracy both in terms of recovering the magnitude and the support of the underlying source signal.
- The proposed methodology exhibits performance that goes beyond the traditional one-bit CS state-of-the-art and allows for designing sensing matrices that are distribution-specific. In conjunction to learning task-specific $\boldsymbol{\Phi}$, the quantization thresholds can also be learned in a joint manner such that the learned parameters improve the signal reconstruction accuracy and speed.
- We propose two generalized optimization algorithms that can be used as standalone algorithms for recovering the amplitude information of the source signal by utilizing non-zero quantization thresholds.

Organization of the Paper: The remainder of this paper is organized as follows. In Sec. II, we discuss the general problem formulation and system model of the one-bit compressive sensing problem and propose two general algorithms that pave the way for incorporating non-zero quantization thresholds. The proposed one-bit compressive autoencoding methodology is presented in Sec. III. The loss function characterization and training method for the proposed model-based deep architectures are discussed at the end of Sec. III. In Sec. IV, we investigate the performance of the proposed methods through various numerical simulations and for various scenarios. Finally, Sec. V concludes the paper.

II. System Model and Problem Formulation

In this paper, we are interested in a one-bit CS measurement model (i.e., the encoder module) with dynamics that can be described as follows:

$$\text{Encoder Module:} \qquad \boldsymbol{r} = \text{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}), \tag{3}$$

where $\boldsymbol{\Phi}^{m\times n}$ denotes the sensing matrix, $\boldsymbol{b} \in \mathbb{R}^m$ is the quantization thresholds, and $\boldsymbol{x} \in \mathbb{R}^n$ is assumed to be a K-sparse signal. Having the one-bit measurements of the form (3), one can pose the problem of sparse signal recovery from one-bit measurements $\boldsymbol{r}$ by solving the following non-convex program:

$$\mathcal{P}_0 : \min_{\boldsymbol{x}\in\mathbb{R}^n} \; \|\boldsymbol{x}\|_0, \;\; \text{s.t.} \;\; \boldsymbol{r} = \text{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}), \tag{4}$$

where the constraint in (4) is imposed to ensure a consistent reconstruction with the available one-bit information. Further note that the one-bit measurement consistency principle in (4) can be equivalently expressed as

$$\boldsymbol{R}\left(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}\right) \succeq \boldsymbol{0}, \tag{5}$$

where $\boldsymbol{R} = \text{Diag}(\boldsymbol{r})$.

Let us first consider the scenario in which the quantization thresholds $\boldsymbol{b}$ are all set to zero. In this case, the non-convex optimization problem $\mathcal{P}_0$ can be further relaxed and expressed as a well-known non-convex ℓ_1-minimization program on the unit sphere [2]:

$$\mathcal{P}_1 : \min_{\boldsymbol{x}} \; \|\boldsymbol{x}\|_1, \;\; \text{s.t.} \;\; \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x} \succeq \boldsymbol{0}, \;\; \|\boldsymbol{x}\|_2 = 1, \tag{6}$$

where the ℓ_1-norm acts as a sparsity inducing function. The intuition behind finding the sparsest signal on the ℓ_2 unit-sphere (i.e., fixing the energy of the recovered signal) is two-fold. First, it reduces the feasible set of the optimization problem as the amplitude information is lost, and second, it avoids the the trivial solution of $\hat{\boldsymbol{x}} = \boldsymbol{0}$. By comparing the acquired data $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$ with non-zero quantization thresholds, the constraint defined in (5) not only reduces the feasible set of the problem by defining a set of hyper-planes where the signal can reside on, but also, implicitly exclude the trivial solution. There exists an extensive body of research on approximately solving the non-convex optimization problem $\mathcal{P}_1$ (e.g., see [2], [3], [5], [36], [37], and the references therein). The most notable methods utilize a regularization term $\mathcal{R}(\boldsymbol{s})$ to enforce the consistency principle via a penalty term added to the ℓ_1-objective function, viz.

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_1 + \alpha\mathcal{R}(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}), \text{ s.t. } \|\boldsymbol{x}\|_2 = 1, \tag{7}$$

where $\alpha > 0$ is the penalty factor.

Among the numerous iterative algorithms available for tackling the optimization problem in (7), we plan to utilize and improve upon the state-of-the-art renormalized fixed-point iterations (RFPI) [2], and the Binary Iterative Hard Thresholding (BIHT) [4] algorithms as the starting point for our proposed hybrid model-aware deep architecture for the problem of one-bit compressive sensing. To this end, we interpret a one-bit CS setting as a single auto-encoder (AE) module allowing for an optimization over all system parametes (i.e., the sensing matrix, quantization thresholds, and the laten variables of first-order optimization techniques). Namely, in the subsequent sections, we use the mentioned algorithms as a base-line to design the decoder module of our one-bit CS AE. In particular, we unfold the iterations of the two specialized algorithms onto the layers of a deep neural network in a fashion that each layer of the proposed deep architecture mimics the behavior of one iteration of the base-line algorithm. Next, we perform an end-to-end learning approach by utilizing the back-propagation method to tune the parameters of both the decoder and the encoder functions of the proposed one-bit compressive AE.

A. Renormalized Fixed-Point Iteration (RFPI)

The RFPI algorithm considers a one-bit CS data acquisition model where the quantization thresholds are all set to zero. With $\boldsymbol{c} = \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}$ and $\boldsymbol{b} = \boldsymbol{0}$, the RFPI algorithm utilizes the following regularization term to enforce the consistency constraint in (6): $\mathcal{R}(\boldsymbol{c}) = \frac{1}{2}\|\rho(\boldsymbol{c})\|_2^2$, where $\rho(\boldsymbol{c}) \triangleq \max\{-\boldsymbol{c}, \boldsymbol{0}\}$, and the function $\max$ is applied element-wise on the vector arguments. Note that the function $\rho(\cdot)$ can be expressed in terms of the well-known Rectifier Linear Unit (ReLU) function extensively used by the deep learning research community, i.e. $\rho(\boldsymbol{c}) = \text{ReLU}(-\boldsymbol{c})$. Briefly speaking, the RFPI algorithm is a first-order optimization method (gradient-based) that operates as follows: given an initial point $\boldsymbol{x}_0$ on the unit-sphere (i.e., $\|\boldsymbol{x}_0\|_2 = 1$), the gradient step-size δ and a shrinkage thresholds α (or equivalently the penalty term), at each iteration i, the estimated signal $\boldsymbol{x}_i$ is obtained using the following update steps:

$$\boldsymbol{d}_i = \nabla_{\boldsymbol{x}}\,\mathcal{R}(\boldsymbol{z})\big|_{\boldsymbol{x}=\boldsymbol{x}_{i-1}} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}_{i-1}\right), \tag{8a}$$

$$\boldsymbol{t}_i = \left(1 + \delta\boldsymbol{d}_i^T\boldsymbol{x}_{i-1}\right)\boldsymbol{x}_{i-1} - \delta\boldsymbol{d}_i, \tag{8b}$$

$$\boldsymbol{v}_i = \text{sign}\left(\boldsymbol{t}_i\right) \odot \text{ReLU}\left(|\boldsymbol{t}_i| - (\delta/\alpha)\boldsymbol{1}\right), \tag{8c}$$

$$\boldsymbol{x}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|_2}. \tag{8d}$$

After the descent in (8a)-(8b), the update step in (8c) corresponds to a shrinkage step. More precisely, any element of the vector $\boldsymbol{t}_i$ that is below the threshold δ/α will be pulled down to zero (leading to enhanced sparsity). Finally, the algorithm projects the obtained vector $\boldsymbol{v}_i$ on the unit sphere to produce the latest estimation of the signal. Note that the latter step is necessary due to the fact that a zero-threshold vector (i.e., $\boldsymbol{b} = \boldsymbol{0}$) is employed at the time of the data acquisition, and hence, the amplitude information is lost.

While effective in signal reconstruction, there exist several drawbacks in using the RFPI method. For instance, it is required to use the algorithm on several problem instances, while increasing the value of the penalty factor α at each outer iteration of the algorithm, and to use the previously obtained solution as the initial point for tackling the recovery problem for any new problem instance. Moreover, it is not straight-forward how to choose the fixed step-size and the shrinkage threshold, that may depend on the latent-parameters of the system. In fact, it is evident that by carefully tuning the step-sizes and the shrinkage threshold $\tau = \delta/\alpha$, one can significantly boost the performance of the algorithm, and further alleviate the mentioned drawbacks of this method. In what follows, we extend the above iterations in a fashion that it allows for incorporating the non-zero quantization thresholds, and hence, enabling us to effectively recover the amplitude information of the source signal.

A.1. Extending the RFPI framework to non-zero quantization thresholds:
Recall that our focus is on the following encoding (measurement) model with an arbitrary threshold vector $\boldsymbol{b}$:

$$\boldsymbol{r} = \text{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}). \tag{9}$$

Therefore, the problem of one-bit CS signal recovery with a non-zero quantization threshold vector can be cast as:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \;\|\boldsymbol{x}\|_1, \;\text{ s.t. }\; \boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}) \succeq \boldsymbol{0}. \tag{10}$$

Inspired by the regularization-based relaxation employed in [2], we relax the above program and cast it as follows:

$$\mathcal{P}_2: \;\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{x}\|_1 + \frac{1}{2}\|\rho\left(\boldsymbol{R}\left(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}\right)\right)\|_2^2. \tag{11}$$

Note that the second term in the objective function above applies a quadratic penalty to the negative entries of the vector $\boldsymbol{R}\left(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}\right)$, i.e., the ones which are not consistent with the acquired one-bit measurements.

In this work, we consider an iterative first-order optimization solver to tackle the above optimization problem. Specifically, we make use of the proximal algorithm [38] to derive the updating steps required to solve $\mathcal{P}_2$. Let $f(\boldsymbol{x}) := g(\boldsymbol{x}) + h(\boldsymbol{x})$ be a composite convex objective function and consider the following optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \equiv g(\boldsymbol{x}) + h(\boldsymbol{x}). \tag{12}$$

Furthermore, define the proximal operator $\text{prox}_{\alpha h}(\boldsymbol{x})$ for a given convex differentiable function $h(\boldsymbol{x})$ as follows:

$$\text{prox}_{\alpha h}(\boldsymbol{x}) = \underset{\boldsymbol{z}\in\mathbb{R}^n}{\text{argmin}} \;\; h(\boldsymbol{z}) + \frac{1}{2\alpha}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2, \tag{13}$$

where $\alpha > 0$. Then, starting from an initial point $\boldsymbol{x}_0$ and a step-size $\delta \in (0,1)$, the overall updating equations of the proximal gradient method for solving (12) can be expressed as:

$$\boldsymbol{x}_{i+1} := \operatorname{prox}_{\alpha h}\left(\boldsymbol{x}_i - \delta \nabla_{\boldsymbol{x}} g(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}\right), \tag{14}$$

where it can be shown that for a convex function h, the proximal operator is monotone and the above mapping (14) has a fixed point, coinciding with the global solution of (12) [38]. Evidently, the proximal gradient algorithm is well suited for tackling our optimization problem $\mathcal{P}_2$. In order to find the updating steps specific to our problem, we consider the splitting of the objective function in (11) into $g(\boldsymbol{x}) = (1/2)\|\rho(\boldsymbol{R}(\boldsymbol{\Phi x} - \boldsymbol{b}))\|_2^2$, and $h(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$. In the following, we first derive the gradient calculations of the function $g(\boldsymbol{x})$, and then, proceed with presenting the final update equations akin to our optimization problem at hand based on (14).

Gradient calculation of $g(\boldsymbol{x})$: Recall that the function $\rho(\boldsymbol{x}) = \max\{-\boldsymbol{x}, \boldsymbol{0}\}$ is applied element-wise on the vector argument. Let $[\boldsymbol{x}]_i$ represent the i-th element of the vector $\boldsymbol{x}$, $\boldsymbol{\phi}_i$ be a column-vector denoting the i-th row of the sensing matrix $\boldsymbol{\Phi}$, and r_i and b_i represent the i-th one-bit measurement and quantization threshold, respectively. Define

$$g_i(\boldsymbol{x}) \triangleq \frac{1}{2}\left([\rho(\boldsymbol{R}(\boldsymbol{\Phi x} - \boldsymbol{b}))]_i\right)^2 \tag{15}$$

$$= \frac{1}{2}\left(\rho(r_i(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i))\right)^2$$

$$= \begin{cases} \frac{1}{2}(r_i)^2\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right)^2 & \text{if } r_i\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right) < 0, \\ 0 & \text{else.} \end{cases} \tag{16}$$

Note that $r_i \in \{\pm 1\}$ and the term $(r_i)^2 = 1$ is provided to obtain a concise representation later. Using the above definition, we have that $g(\boldsymbol{x}) = \sum_i g_i(\boldsymbol{x})$, and the convexity of each sub-function $g_i(\boldsymbol{x})$ renders the overall function $g(\boldsymbol{x})$ convex. Now, assuming $(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i) \neq 0$, the gradient of the sub-function $g_i(\boldsymbol{x})$ can be expressed as:

$$\nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) = \rho(r_i(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i))(-r_i\boldsymbol{\phi}) \tag{17}$$

$$= \begin{cases} r_i\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right)(-r_i\boldsymbol{\phi}_i) & \text{if } r_i\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right) < 0, \\ 0 & \text{else.} \end{cases}$$

Furthermore, note that $g_i(\boldsymbol{x})$ is convex, and hence, in the case of $(\boldsymbol{\phi}^T\boldsymbol{x} - b_i) = 0$, the set of subgradients of the function are given by the convex hull $\{\lambda\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right)(-\boldsymbol{\phi}_i) : \lambda \in [0,1]\} \ni \rho(r_i(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i))(-r_i\boldsymbol{\phi})$ (i.e., the term (17) is a subgradient as well). Therefore, the gradient of the overall objective function $g(\boldsymbol{x})$ can be compactly expressed as:

$$\nabla_{\boldsymbol{x}} g(\boldsymbol{x}) = \sum_i \nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}) = -(\boldsymbol{R\Phi})^T\rho\left(\boldsymbol{R}\left(\boldsymbol{\Phi x} - \boldsymbol{b}\right)\right). \tag{18}$$

The final step is to derive the proximal operator for the function $h(\boldsymbol{x})$. The proximal operator can be analytically derived for many convex functions including the one considered in this work. In particular, the proximal mapping for the function $h(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ is given by the well-known (element-wise) *soft-thresholding* function (defined below), which recasts the overall update equation given in (14) for solving our problem $\mathcal{P}_2$ as follows:

$$\boldsymbol{x}_{i+1} = \operatorname{prox}_{\alpha h}\left(\tilde{\boldsymbol{t}}_i\right) \tag{19}$$

$$= \operatorname{sign}\left(\tilde{\boldsymbol{t}}_i\right) \odot \max\left\{|\tilde{\boldsymbol{t}}_i| - (\delta/\alpha)\boldsymbol{1}, 0\right\}, \text{ and} \tag{20}$$

$$\tilde{\boldsymbol{t}}_i = \boldsymbol{x}_i - \delta\nabla_{\boldsymbol{x}} g(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}, \tag{21}$$

where all the functions above are applied element-wise on the vector argument. Generally speaking, the performance and the convergence of the above iterations depends heavily on the choice of step-size δ and the thresholding factors α. In the following section, we show how the above iterations can be unfolded onto the layers of a deep neural network allowing for obtaining an enhanced first-order optimizer.

The proposed algorithm for solving the optimization problem $\mathcal{P}_2$ associated with the incorporation of non-zero quantization thresholds is summarized below.

The Proposed Generalized RFP Iterations:

$$\tilde{\boldsymbol{d}}_i = -\left(\boldsymbol{R\Phi}\right)^T\rho\left(\boldsymbol{R}\left(\boldsymbol{\Phi x}_{i-1} - \boldsymbol{b}\right)\right), \tag{22a}$$

$$\tilde{\boldsymbol{t}}_i = \boldsymbol{x}_{i-1} - \delta\tilde{\boldsymbol{d}}_i, \tag{22b}$$

$$\boldsymbol{x}_i = \operatorname{sign}\left(\tilde{\boldsymbol{t}}_i\right) \odot \operatorname{ReLU}\left(|\tilde{\boldsymbol{t}}_i| - (\delta/\alpha)\boldsymbol{1}\right), \tag{22c}$$

where (22a) corresponds to computing the gradient at the current point, while the step (22b) can be viewed as taking a descent step on the one-sided ℓ_2-norm using the obtained gradient, and (22c) corresponds to applying the proximal mapping operator. In the rest of this paper, we refer to the iterations presented in (22) as *Generalized RFPI* (G-RFPI).

B. Binary Iterative Hard Thresholding Algorithm (BIHT)

The BIHT algorithm is a simple, yet powerful, first-order iterative reconstruction algorithm for the problem of one-bit CS where the sparsity level K is assumed to be known a priori. BIHT iterations can be seen as a simple modification of the iterative hard thresholding (IHT) algorithm proposed in [39]. Similar to the RFPI algorithm, the BIHT method considers a zero-level quantization threshold. However, in contrast to the RFPI algorithm, it exploits the knowledge of the sparsity level K of the signal of interest. In other words, the BIHT algorithm is designed to tackle the following counterpart of $\mathcal{P}_0$:

$$\mathcal{P}_3: \min_{\boldsymbol{x}\in\mathbb{R}^n} \|\rho\left(\boldsymbol{R\Phi x}\right)\|_1, \text{ s.t. } \|\boldsymbol{x}\|_0 = K, \ \|\boldsymbol{x}\|_2 = 1, \tag{23}$$

where $\rho(\boldsymbol{c}) = \max\{-\boldsymbol{c}, \boldsymbol{0}\}$ and $\boldsymbol{R} = \operatorname{Diag}(\boldsymbol{r})$ as before. Note that the one-sided ℓ_1 objective function above (also related to the hinge-loss) enforces the consistency principle previously introduced in (6), and that by solving the above optimization problem, we are working to achieve *maximal consistency* with the one-bit measurements $\boldsymbol{r}$. It is worth mentioning that one can also consider different objective functions, and not necessarily an ℓ_1 objective, as long as it promotes the data consistency principle (e.g., ℓ_2 norm). For a detailed analysis of different candidates for the objective function and their properties, see [39].

The BIHT iterations are described as follows. Let $\boldsymbol{c} = \boldsymbol{R\Phi x}$, and define $\mathcal{F}(\boldsymbol{c}) = \|\rho(\boldsymbol{c})\|_1$. Furthermore, let $\partial\mathcal{F}(\boldsymbol{x}) := \{\boldsymbol{f} : \mathcal{F}(\boldsymbol{x}) - \mathcal{F}(\boldsymbol{y}) \geq \langle\boldsymbol{f}, \boldsymbol{x} - \boldsymbol{y}\rangle\}$ denotes the sub-differential set of $\mathcal{F}$ at the point $\boldsymbol{x} \in \mathbb{R}^n$. Then, given an initial point $\boldsymbol{x}_0$, the sparsity level K, and one-bit measurements $\boldsymbol{r}$ (or equivalently $\boldsymbol{R}$), at the i-th iteration, the BIHT algorithm updates the current estimate of the signal $\boldsymbol{x}_i$ through the following steps:

$$\boldsymbol{u}_{i+1} = \boldsymbol{x}_i - \delta \boldsymbol{f}_i = \boldsymbol{x}_i + \frac{\delta}{2}\boldsymbol{\Phi}^T\left(\boldsymbol{r} - \operatorname{sign}(\boldsymbol{\Phi}\boldsymbol{x}_{i-1})\right), \quad (24a)$$

$$\boldsymbol{x}_{i+1} = \mathcal{H}_K\left(\boldsymbol{u}_i\right), \quad (24b)$$

where $\boldsymbol{f}_{i-1} \in \partial\mathcal{F}(\boldsymbol{x})$ denotes a sub-gradient of the one-side ℓ_1 objective function in $\mathcal{P}_3$ at $\boldsymbol{x}_i$, $\delta > 0$ governs the fixed gradient step-size, and the projection operator $\mathcal{H}_K(\boldsymbol{x})$ is defined such that it retains the largest K elements (in magnitude) of the vector argument, and set the rest of the elements to zero. Note that the term $(-1/2)\boldsymbol{\Phi}^T(\boldsymbol{r} - \operatorname{sign}(\boldsymbol{\Phi}\boldsymbol{x}))$ is a sub-gradient of the objective function at point $\boldsymbol{x}$ (more details is provided in subsection B.1 below).

The step (24a) can be interpreted as taking a descent step using the computed sub-gradient of the objective function (23), while the projection step in (24b) can be viewed as a projection of $\boldsymbol{u}_i$ onto the support set of K-sparse signals. Once the above iterations terminate either by fully satisfying the consistency principle (i.e., obtaining $\boldsymbol{x}^*$ such that $\mathcal{F}(\boldsymbol{x}^*) = 0$), or by achieving a maximum number of iterations, the ultimate step to be taken is projecting the final estimate $\boldsymbol{x}^*$ onto the unit-sphere, viz. $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^*/\|\boldsymbol{x}^*\|_2$. Note that this is in contrast to the RFPI algorithm as the BIHT iterations does not require a normalization step as in (8d) at each iteration.

B.1. Extending the BIHT framework to non-zero quantization thresholds:

The extension of the BIHT iterations to incorporate the non-zero thresholds vector $\boldsymbol{b}$ is straight-forward. In the case of non-zero quantization thresholds, we cast the signal recovery problem as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \mathcal{F}(\boldsymbol{x}) \equiv \|\rho\left(\boldsymbol{R}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b})\right)\|_1, \ \text{ s.t. } \ \|\boldsymbol{x}\|_0 = K, \quad (25)$$

where $\boldsymbol{R} = \operatorname{Diag}(\boldsymbol{r})$ and $\boldsymbol{r} = \operatorname{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b})$. Further note that the unit-ball constraint has been dropped due to the fact that the amplitude ambiguity is resolved by employing non-zero quantization thresholds.

In order to tackle the optimization problem (25), we make use of the well-known iterative hard thresholding algorithm (IHT) extensively used in the literature for general CS problems. In particular, the updating steps of the iterative hard thresholding algorithm is consist of taking a descent step on the objective function in (25) followed by a projection operator denoted by $\mathcal{H}_k$. Starting from a K-sparse initial point $\boldsymbol{x}_0$, the overall updating equation of the IHT algorithm for recovering the underlying K-sparse solution is given by:

$$\boldsymbol{x}_{i+1} = \mathcal{H}_k\left(\boldsymbol{x}_i - \delta\boldsymbol{f}_i\right), \quad (26)$$

where $\boldsymbol{f}_i \in \partial\mathcal{F}(\boldsymbol{x})$ at $\boldsymbol{x}_i$, and $\delta > 0$ denotes the step-size. Similar to the steps we took in (9)-(22), and extending the analysis provided in [4], we define $\mathcal{F}(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x})$, where the convex sub-function f_i is given by:

$$f_i(\boldsymbol{x}) = |[\rho\left(\boldsymbol{R}\left(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}\right)\right)]_i| \quad (27)$$

$$= \begin{cases} |\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i| & \text{if } r_i\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right) < 0, \\ 0 & \text{else.} \end{cases} \quad (28)$$

Assuming $(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i) \neq 0$, the gradient of the sub-function f_i can be expressed as follows:

$$\nabla_{\boldsymbol{x}} f_i(\boldsymbol{x}) = -\frac{1}{2}\left(r_i - \operatorname{sign}(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i)\right)\boldsymbol{\phi}_i \quad (29)$$

$$= \begin{cases} \operatorname{sign}(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i)\boldsymbol{\phi}_i & \text{if } r_i\left(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i\right) < 0, \\ 0 & \text{else.} \end{cases}$$

In the case of $(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i) = 0$, since the sub-function f_i is convex, we have the sub-gradients of $f_i(\boldsymbol{x})$ given by the set

$$\partial f_i(\boldsymbol{x}) = \left\{-\frac{\lambda}{2}(r_i - \operatorname{sign}(\phi_i^T\boldsymbol{x} - b_i))\boldsymbol{\phi}_i : \lambda \in [0,1]\right\} \quad (30)$$

$$\ni -\frac{1}{2}\left(r_i - \operatorname{sign}(\boldsymbol{\phi}_i^T\boldsymbol{x} - b_i)\right)\boldsymbol{\phi}_i. \quad (31)$$

Hence, considering that $\mathcal{F}(\boldsymbol{x}) = \sum_i f_i(\boldsymbol{x})$, we have

$$\boldsymbol{f} = -\frac{1}{2}\boldsymbol{\Phi}^T\left(\boldsymbol{r} - \operatorname{sign}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b})\right) \in \partial\mathcal{F}(\boldsymbol{x}). \quad (32)$$

Consequently, the updating steps of the IHT algorithm defined in (26) for solving the optimization problem (25) can be expressed as:

<u>The Proposed Generalized BIHT Iterations:</u>

$$\boldsymbol{u}_i = \boldsymbol{x}_{i-1} + \frac{\delta}{2}\boldsymbol{\Phi}^T\left(\boldsymbol{r} - \operatorname{sign}(\boldsymbol{\Phi}\boldsymbol{x}_{i-1} - \boldsymbol{b})\right), \quad (33a)$$

$$\boldsymbol{x}_i = \mathcal{H}_K\left(\boldsymbol{u}_i\right), \quad (33b)$$

Note the exception that in the proposed generalized BIHT iterations, there is no need for the normalization of the obtained estimate of the signal $\boldsymbol{x}^*$ after the update steps terminate. This is due to the fact that a non-zero quantization threshold vector is employed at time of the encoding, and hence, the amplitude information is not fully lost. In the rest of this paper, we refer to the above iterations as *Generalized BIHT* (G-BIHT) algorithm.

Although simple and powerful, the BIHT algorithm requires a careful choice of the gradient step-size δ for convergence and stability, and there is no straight-forward method to properly choose the gradient step-size. Moreover, it only utilizes a fixed step-size through all iterations. This motivates the development of a methodology by which one can design a decoder function that exploits adaptive gradient step-sizes, which can result in a significant improvement of the performance of the BIHT algorithm.

In the next section, we discuss a slight over-parametrization of the iterations of RFPI, G-RFPI, BIHT, and G-BIHT algorithms that paves the way for the design of our proposed one-bit compressive AE and for jointly designing the parameters of the encoder function defined in (3) parametrized on the sensing matrix $\boldsymbol{\Phi}$, the quantization thresholds $\boldsymbol{b}$, and the design of a set of decoder functions based on the discussed iterative optimization algorithms.

III. The Proposed Model-Based Deep Learning Models for One-Bit CS

We pursue the design of a novel *model-driven one-bit compressive sensing-based autoencoder* deep architecture that facilitates the joint design of the parameters of both the encoder and the decoder module when one-bit quantizers with non-zero thresholds are employed in the data acquisition process (i.e., the encoding module) for a K-sparse input signal $\boldsymbol{x} \in \mathbb{R}^n$. In particular, the decoder module can be seen as model-based deep architecture which is derived upon unfolding the iterations of first-order optimization techniques provided above onto the layers of a deep neural network.

In general terms, a AE is a generative model comprised of an encoder and a decoder module that are sequentially connected together. The purpose of an AE is to learn an abstract representation of the input data, while providing a powerful data reconstruction system through the decoder module. The input to such a system is a set of signals following a certain distribution, i.e. $\boldsymbol{x} \sim \mathcal{D}(\boldsymbol{x})$, and the output is the recovered signal from the decoder module $\hat{\boldsymbol{x}}$. Hence, the goal is to jointly learn an abstract representation of the underlying distribution of the signals through the encoder module, and simultaneously, learning a decoder module allowing for reconstruction of the compressed signals from the obtained abstract representations. Therefore, an AE can be defined by two main functions: *i*) an encoder function $f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1} : \mathbb{R}^n \mapsto \mathbb{R}^m$, parameterized on a set of variables $\boldsymbol{\Upsilon}_1$ that maps the input signal into a new vector space, and *ii*) a decoder function $f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2} : \mathbb{R}^m \mapsto \mathbb{R}^n$ parameterized on $\boldsymbol{\Upsilon}_2$, which maps the output of the encoder module back into the original signal space. Hence, the governing dynamics of a general auto-encoder can be expressed as $\hat{\boldsymbol{x}} = f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2} \circ f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x})$, where $\hat{\boldsymbol{x}}$ denotes the reconstructed signal.

In light of the above, we seek to interpret a one-bit CS system as an AE module facilitating not only the design of the sensing matrix $\boldsymbol{\Phi}$ and the quantization thresholds $\boldsymbol{b}$ that best captures the information of a K-sparse signal when one-bit quantizers are employed, but also to learn the parameters of an iterative optimization algorithm specifically designed for the task of signal recovery. To this end, we modify and unfold the iterations of the proposed G-RFPI algorithm defined in (22), and the GBIHT method defined in (33) onto the layers of a deep neural network and later use the deep learning tools to tune the parameters of the proposed one-bit compressive AE.

A. Structure of the Encoding Module

In its most general form, we define the encoder module of the proposed AE based on our data-acquisition model defined in (3), as follows:

$$f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x}) = \widetilde{\text{sign}}(\boldsymbol{\Phi}\boldsymbol{x} - \boldsymbol{b}), \tag{34}$$

where $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}, \boldsymbol{b}\}$ denotes the set of learnable parameters of the encoder function, and $\widetilde{\text{sign}}(\boldsymbol{x}) = \tanh(t \cdot \boldsymbol{x})$, for a large $t > 0$ (t was set to 50 in numerical investigations). Note that we replaced the original sign function with a smooth differentiable approximation of it based on the hyperbolic tangent function due to the fact that the sign function is not continuous and its gradient is zero everywhere except at the origin. Hence, the use of it would cripple stochastic gradient-based optimization methods (later used in back-propagation method for deep learning).

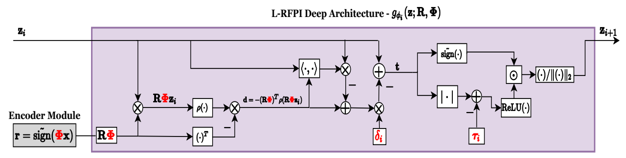
B. Structure of the Decoding Module

In this part, we describe the different scenarios under which we pursue the design of our decoder function by using the RFPI, BIHT, and the suggested G-RFPI and G-BIHT iterations. In particular, we fix the total complexity of our decoding module by fixing the total number of iterations allowed for the mentioned optimization iterations. Next, we slightly over-parameterize each iteration/step of the mentioned algorithms to increase the per-iteration degrees-of-freedom of each method and to further account for the learnable latent variables in the system. Finally, we unfold the iterations of each algorithm onto the layers of a deep architecture such that each layer of the deep network resembles one iteration of the base-line algorithm. We then seek to learn the parameters of both the decoder and encoder function using the training tools already developed for deep learning. We consider the following cases to design our decoder function:

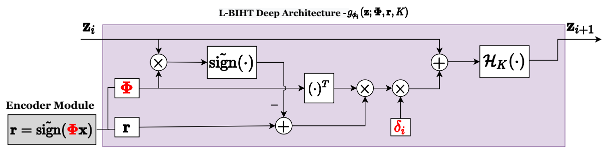
- **Learned RFPI (L-RFPI)**: We consider the RFPI iterations defined in (8) as our base-line but slightly over-parametrize its iterations by introducing a gradient step-size δ_i and a *shrinkage* thresholds vector $\boldsymbol{\tau}_i$ *for each iteration* i. This is in contrast to the original RFP iterations where a fixed gradient step-size δ, and shrinkage threshold $\tau = (\delta/\alpha)\mathbf{1}$ were employed for all iterations. Hence, the proposed unfolded over-parametrized iterations are much more expressive. The decoder function will be parameterized on $\boldsymbol{\Upsilon}_2 = \{\delta_i, \boldsymbol{\tau}_i\}_{i=0}^{L-1}$, and the encoder function will be parametrized on the set $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}\}$ (note that $\boldsymbol{b} = \mathbf{0}$).
- **Learned BIHT (L-BIHT)**: We consider the unfolding of the iterations of the BIHT defined in (24) similar to the previous case and by introducing per-iteration gradient step-sizes δ_i in lieu of a fixed gradient-step size along all iterations. In this case, the decoder function will be parametrized on the set $\boldsymbol{\Upsilon}_2 = \{\delta_i\}$, while the set of parameters of the encoding module is $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}\}$; both are to be learned.
- **Learned G-RFPI (LG-RFPI)**: We consider the unfolding of the proposed Generalized RFPI iterations in (22) in a non-zero quantization thresholds setting. We over-parameterize the iterations of the proposed G-RFPI by parametrizing the decoder function on the set $\boldsymbol{\Upsilon}_2 = \{\delta_i, \boldsymbol{\tau}_i\}_{i=0}^{L-1}$, and this time, by parameterizing the encoder function on both the sensing matrix and the quantization thresholds vector, i.e. $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}, \boldsymbol{b}\}$.
- **Learned G-BIHT (LG-BIHT)**: We consider the unfolding of the G-BIHT iterations defined in (33) in a similar manner, i.e. by parameterizing the decoder function on $\boldsymbol{\Upsilon}_2 = \{\delta_i\}_{i=0}^{L-1}$. However, similar to the previous case, we further parametrize the encoder function on the quantization thresholds vector in conjunction with the sensing matrix, i.e. $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}, \boldsymbol{b}\}$.

C. The Proposed One-Bit Compressive Autoencoding Approach

In the following, we describe the design of four novel deep architectures based on the above mentioned structures and discuss the governing dynamics of the proposed one-bit compressive sensing-based AE.


L-RFPI Deep Architecture - $g_{\phi_i}(\mathbf{z};\mathbf{R},\mathbf{\Phi})$
$\mathbf{z}_i$
$\mathbf{z}_{i+1}$
Encoder Module
$\mathbf{r} = \tilde{\text{sign}}(\mathbf{\Phi x})$
$\mathbf{R\Phi}$
$\mathbf{R\Phi z}_i$
$\rho(\cdot)$
$(\cdot)^T$
$\mathbf{d} = -(\mathbf{R\Phi})^T\rho(\mathbf{R\Phi z}_i)$
$\langle\cdot,\cdot\rangle$
$\mathbf{t}$
$\tilde{\text{sign}}(\cdot)$
$|\cdot|$
ReLU$(\cdot)$
$(\cdot)/\|(\cdot)\|_2$
δ_i
τ_i

(a)


L-BIHT Deep Architecture - $g_{\phi_i}(\mathbf{z};\mathbf{\Phi},\mathbf{r},K)$
$\mathbf{z}_i$
$\mathbf{z}_{i+1}$
Encoder Module
$\mathbf{r} = \tilde{\text{sign}}(\mathbf{\Phi x})$
$\mathbf{\Phi}$
$\mathbf{r}$
$\tilde{\text{sign}}(\cdot)$
$(\cdot)^T$
$\mathcal{H}_K(\cdot)$
δ_i

(b)

Fig. 1. An illustration of the computation dynamics in i-th layer of the proposed (a) L-RFPI and (b) L-BIHT deep architectures, in which the trainable parameters are highlighted in red color.

C.1. L-RFPI-Based Compressive Autoencoding: In this case, we consider the following parameterized encoder function:

$$f_{\mathbf{\Upsilon}_1}^{\text{Encoder}}(\boldsymbol{x}) = \tilde{\text{sign}}(\mathbf{\Phi}\boldsymbol{x}), \quad \text{where } \mathbf{\Upsilon}_1 = \{\mathbf{\Phi}\}. \tag{35}$$

As for the decoder function, and based on the RFPI iterations in (8), define $g_{\phi_i} : \mathbb{R}^m \mapsto \mathbb{R}^n$ as follows:

$$g_{\phi_i}(\boldsymbol{z};\mathbf{\Phi},\boldsymbol{R}) = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \quad \text{with} \tag{36a}$$

$$\boldsymbol{v} = \tilde{\text{sign}}\,(\boldsymbol{t}) \odot \text{ReLU}\,(|\boldsymbol{t}| - \boldsymbol{\tau}_i)\,, \tag{36b}$$

$$\boldsymbol{t} = \left(1 + \delta_i \boldsymbol{d}^T \boldsymbol{z}\right) \boldsymbol{z} - \delta_i \boldsymbol{d}, \tag{36c}$$

$$\boldsymbol{d} = -\left(\boldsymbol{R}\mathbf{\Phi}\right)^T \rho\left(\boldsymbol{R}\mathbf{\Phi}\boldsymbol{z}\right), \tag{36d}$$

where $\phi_i = \{\boldsymbol{\tau}_i, \delta_i\}$ represents the parameters of the function g_{ϕ_i}, and $\boldsymbol{\tau}_i \in \mathbb{R}^n$ denotes the sparsity inducing *shrinkage* thresholds vector, and δ_i represents the gradient step-size at iteration i. Next, we define the proposed L-RFPI composite decoder function as follows:

$$f_{\mathbf{\Upsilon}_2}^{\text{Decoder}}(\boldsymbol{z}_0) = g_{\phi_{L-1}} \circ g_{\phi_{L-2}} \circ \cdots \circ g_{\phi_1} \circ g_{\phi_0}(\boldsymbol{z}_0;\mathbf{\Phi},\boldsymbol{R}), \tag{37}$$

where $\mathbf{\Upsilon}_2 = \{\phi_i\}_{i=0}^{L-1}$ represents the learnable (tunable) parameters of the decoder function, and $\boldsymbol{z}_0$ is some initial point of choice. Note that we have over-parameterized the iterations of the RFPI algorithm by introducing the new variable $\boldsymbol{\tau}_i$ at each iteration for the sparsity inducing step in (36b). Moreover, in contrast with the original RFPI iterations, we have introduced a new step-size δ_i at each step of the iteration as well (see Eq. (36c)). Therefore, the above decoder function can be interpreted as performing L iterations of the original RFPI algorithm with an additional $L(n+1)-2$ degrees of freedom (as compared to the base algorithm) expressed in terms of the set of the shrinkage thresholds $\boldsymbol{\tau}_i$ and the gradient step-sizes δ_i, i.e. $\{\boldsymbol{\tau}_i, \delta_i\}_{i=0}^{L-1}$. As a reslt, the proposed decoder function is much more expressive than that of the iterations of RFPI algorithm. A depiction of the computation dynamics of the i-th layer of the proposed L-RFPI deep architecture is provided in Fig. 1(a).

Remark: Note that the above encoder and decoder function, once cascaded together, can be viewed as a deep neural network with $(L+1)$ layers, where the dynamics of the first layer is described by the encoder function defined in (35), and the governing dynamics of the succeeding layers is described by computations of the form (36a)-(36d). Equivalently, such a deep architecture can be viewed as a computational graph with shared variables among the computation nodes, and thus, its parameters can be efficiently optimized by utilizing known deep learning tools such as back-propagation. Hence, the goal is to *jointly* learn the parameters of such a cascaded network (i.e., $\mathbf{\Upsilon}_1 \cup \mathbf{\Upsilon}_2$) in an end-to-end manner by using the available data at hand coming from the underlying distribution of the source signal $\boldsymbol{x}$. ■

C.2. L-BIHT-Based Compressive Autoencoding: Similar to the previous case, we consider the same encoding function parametrized only on the learnable sensing matrix $\mathbf{\Phi}$ in a zero quantization thresholds setting, i.e. $\mathbf{\Upsilon}_1 = \{\mathbf{\Phi}\}$. The governing equations for the decoder function in the case of the proposed Learned BIHT are as follows. We re-define $g_{\phi_i} : \mathbb{R}^m \mapsto \mathbb{R}^n$ as:

$$g_{\phi_i}(\boldsymbol{z};\mathbf{\Phi},\boldsymbol{r},K) = \mathcal{H}_K\left(\boldsymbol{v}\right), \quad \text{for } i < L, \text{where} \tag{38a}$$

$$\boldsymbol{v} = \boldsymbol{z} + \delta_i \mathbf{\Phi}^T \left(\boldsymbol{r} - \tilde{\text{sign}}(\mathbf{\Phi}\boldsymbol{z})\right), \tag{38b}$$

with $\phi_i = \{\delta_i\}_{i=0}^{L-1}$, and where we have an added final layer $i = L$, to renormalize the reconstructed signal as,

$$g_{\phi_L}(\boldsymbol{z};\mathbf{\Phi},\boldsymbol{r}) = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|_2}. \tag{39}$$

Therefore, similar to the previous case, the proposed L-BIHT-based decoder function is defined as:

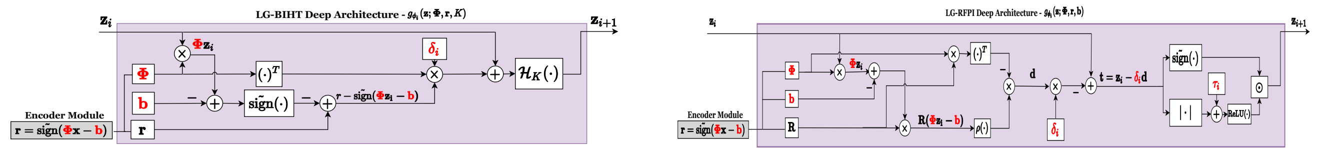
$$f_{\mathbf{\Upsilon}_2}^{\text{Decoder}}(\boldsymbol{z}_0) = g_{\phi_L} \circ g_{\phi_{L-1}} \circ \cdots \circ g_{\phi_1} \circ g_{\phi_0}(\boldsymbol{z}_0;\mathbf{\Phi},\boldsymbol{R},K). \tag{40}$$

We again observe the slight over-parametrization of the L-BIHT algorithm during the unfolding process. Namely, at each iteration we are introducing the per-iteration step-sizes δ_i to be learned which further enhances the performance of our iterations (see (38)). In this case, the decoder function is parameterized only on the gradient step-sizes, i.e. $\mathbf{\Upsilon}_2 = \{\delta_i\}_{i=0}^{L-1}$. The L-BIHT iterations have an additional $(L-1)$ degrees of freedom compared to that of the original BIHT iterations. Fig. 1(b) illustrates the computation dynamics of the i-th layer of the proposed L-BIHT deep architecture.

C.3. LG-RFPI-Based Compressive Autoencoding: We consider the unfolding of iterations of the Learned Generalized RFPI method according to (22). As previously discussed, in the generalized iterations of both the RFPI and BIHT algorithms, the encoder module can be expressed as:

$$f_{\mathbf{\Upsilon}_1}^{\text{Encoder}}(\boldsymbol{x}) = \tilde{\text{sign}}(\mathbf{\Phi}\boldsymbol{x} - \boldsymbol{b}), \tag{41}$$

where $\mathbf{\Upsilon}_2 = \{\mathbf{\Phi}, \boldsymbol{b}\}$, and $\boldsymbol{b}$ represents the tunable vector of quantization thresholds. We follow a similar approach to the


LG-BIHT Deep Architecture - $g_{\phi_i}(\mathbf{z}; \mathbf{\Phi}, \mathbf{r}, K)$
$\mathbf{z}_i$
$\mathbf{z}_{i+1}$
$\mathbf{\Phi}\mathbf{z}_i$
$\mathbf{\Phi}$
$\mathbf{b}$
$\mathbf{r}$
$(\cdot)^T$
δ_i
$\tilde{\text{sign}}(\cdot)$
$r - \tilde{\text{sign}}(\mathbf{\Phi}\mathbf{z}_i - \mathbf{b})$
$\mathcal{H}_K(\cdot)$
Encoder Module
$\mathbf{r} = \tilde{\text{sign}}(\mathbf{\Phi}\mathbf{x} - \mathbf{b})$
LG-RFPI Deep Architecture - $g_{\phi_i}(\mathbf{z}; \mathbf{\Phi}, \mathbf{r}, \mathbf{b})$
$\mathbf{z}_i$
$\mathbf{z}_{i+1}$
$\mathbf{\Phi}$
$\mathbf{b}$
$\mathbf{R}$
$\mathbf{\Phi}\mathbf{z}_i$
$(\cdot)^T$
$\mathbf{R}(\mathbf{\Phi}\mathbf{z}_i - \mathbf{b})$
$\rho(\cdot)$
$\mathbf{d}$
δ_i
$\mathbf{t} = \mathbf{z}_i - \delta_i \mathbf{d}$
$\tilde{\text{sign}}(\cdot)$
$|\cdot|$
τ_i
ReLU$(\cdot)$
$\odot$
Encoder Module
$\mathbf{r} = \tilde{\text{sign}}(\mathbf{\Phi}\mathbf{x} - \mathbf{b})$

(a) (b)

Fig. 2. An illustration of the computation dynamics in the i-th layer of the proposed (a) LG-BIHT and (b) LG-RFPI deep architectures, in which the trainable parameters are highlighted in red color.

proposed L-RFPI-Based deep architecture and slightly over-parameterize the iterations in (22a)-(22c), leading to the design of the decoder function:

$$g_{\phi_i}(\boldsymbol{z}; \boldsymbol{\Phi}, \boldsymbol{R}, \boldsymbol{b}) = \tilde{\text{sign}}\left(\boldsymbol{v}\right) \odot \text{ReLU}\left(|\boldsymbol{v}| - \boldsymbol{\tau}_i\right), \quad \text{with} \tag{42a}$$

$$\boldsymbol{v} = \boldsymbol{z} - \delta_i \boldsymbol{d}, \tag{42b}$$

$$\boldsymbol{d} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\left(\boldsymbol{\Phi}\boldsymbol{z} - \boldsymbol{b}\right)\right), \tag{42c}$$

where $\phi_i = \{\boldsymbol{\tau}_i, \delta_i\}$ represents the parameters of the function g_{ϕ_i}, $\boldsymbol{\tau}_i \in \mathbb{R}^n$ denotes the sparsity inducing thresholds vector, and δ_i represents the gradient step-size at iteration i. Hence, the proposed decoder function $f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2}(\boldsymbol{z}_0)$ can be represented in the same way as in (37), with $\boldsymbol{\Upsilon}_2 = \{\phi_i\}_{i=0}^{L-1}$. Note that by incorporating the non-zero quantization thresholds, there is no need for an additional normalization term at each iteration. The above iterations (comprising the decoder function) have the same degree of freedom as L-RFPI iterations—an additional $L(n+1) - 2$ model parameters compared to that of the baseline G-RFPI iterations. Also, note the additional m degrees of freedom that the encoder function offers in terms of tunable quantization thresholds vector $\boldsymbol{b}$ (in addition to the sensing matrix). Fig. 2(b) illustrates the computation dynamics of the i-th layer of the proposed L-BIHT deep architecture.

C.4. LG-BIHT-Based Compressive Autoencoding: We consider an encoder function $f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}$ of the form (41), where $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}, \boldsymbol{b}\}$ denotes the learnable sensing matrix and arbitrary quantization thresholds. Additionally, we present an over-parameterization of the Genralized BIHT iterations (see Eqs. (33)) and consider the resulting unfolded network as the blueprint of our decoder. Namely, we define $g_{\phi_i} : \mathbb{R}^m \mapsto \mathbb{R}^n$ as:

$$g_{\phi_i}(\boldsymbol{z}; \boldsymbol{\Phi}, \boldsymbol{r}, \boldsymbol{b}, K) = \mathcal{H}_K\left(\boldsymbol{v}\right), \quad \text{with} \tag{43a}$$

$$\boldsymbol{v} = \boldsymbol{z} + \delta_i \boldsymbol{\Phi}^T \left(\boldsymbol{r} - \tilde{\text{sign}}(\boldsymbol{\Phi}\boldsymbol{z} - \boldsymbol{b})\right), \tag{43b}$$

where $\phi_i = \{\delta_i\}$ denotes the set of parameters of the function g_{ϕ_i}. Note that due to employing a non-zero thresholds vector, we do not need the additional normalization layer as in (39) for this case. Consequently, the decoder function $f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2}$ can be expressed in a similar manner as in (40), with $\boldsymbol{\Upsilon}_2 = \{\delta_i\}_{i=0}^{L-1}$. These iterations, similar to L-BIHT case, have an additional $(L-1)$ degrees of freedom compared to that of the baseline G-BIHT iterations; whereas, the encoder function has an additional m tunable parameters in terms of the one-bit quantization thresholds compared to that of the L-BIHT-based AE. Fig. 2(a) illustrates the computation dynamics of the i-th layer of the proposed LG-BIHT deep architecture.

In the next section, we discuss the training process of the above proposed one-bit compressive autoencoders. Particularly, we formulate a proper loss function that facilitates the training of such unfolded deep architectures, and for each model, we seek to jointly learn the set of parameters of the entire network (i.e., the encoder and decoder function) in a end-to-end manner using the available deep learning techniques.

D. Loss Function Characterization and Training Method

The output of an autoencoder is the reconstructed signal from the compressed measurements, i.e.

$$\hat{\boldsymbol{x}} = f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2} \circ f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x}),$$

where $\boldsymbol{x} \sim \mathcal{D}(\boldsymbol{x})$ and $\hat{\boldsymbol{x}}$ denotes the input and output of the AE, respectively. The training of an AE should be carried out by defining a proper loss function $\mathcal{G}\left(\boldsymbol{x}, f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2} \circ f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x})\right)$ that provides a measure of the similarity between the input and the output of the AE. The goal is to minimize the distance between the input target signal $\boldsymbol{x}$ and the recovered signal $\hat{\boldsymbol{x}}$ according to a similarity criterion. A widely-used option for the loss function is the output MSE loss, i.e., $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{x})}\left\{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2\right\}$, and hence, the training loss of such a system can be formulated as:

$$\mathcal{G}(\boldsymbol{x}; \hat{\boldsymbol{x}}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{x})}\left\{\|\boldsymbol{x} - f^{\text{Decoder}}_{\boldsymbol{\Upsilon}_2} \circ f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x})\|_2^2\right\}$$

that is to be minimized over $\boldsymbol{\Upsilon}_1$ and $\boldsymbol{\Upsilon}_2$. Nevertheless, in deep architectures with a high number of layers and parameters, such a simple choice of the loss function makes it difficult to back-propagate the gradients; in fact, the vanishing gradient problem arises. Therefore, for the training of the proposed AE, a better choice for the loss function is to consider the cumulative MSE loss of the layers. As a result, one can also feed-forward the decoder function for only $l < L$ layers (a lower complexity decoding), and consider the output of the l-th layer as a good approximation of the target signal. For training, one needs to consider the constraint that the gradient step-sizes $\{\delta_i\}_{i=0}^{L-1}$, and the shrinkage thresholds $\{\boldsymbol{\tau}_i\}_{i=0}^{L-1}$ must be non-negative. By parameterizing the decoder function on the step-sizes and the shrinkage step thresholds, we need to regularize the training loss function ensuring that the network chooses positive step sizes and shrinkage thresholds at each layer. With this in mind, we suggest the following loss function for training the proposed one-bit compressive AE.

Let $\tilde{g}_i = g_{\phi_i} \circ g_{\phi_{i-1}} \circ \cdots \circ g_{\phi_0} \circ f_{\boldsymbol{\Upsilon}_1}^{\text{Encoder}}(\boldsymbol{x})$, and define the loss function for training as

$$\mathcal{G}_L(\boldsymbol{x};\hat{\boldsymbol{x}}) = \underbrace{\sum_{i=0}^{L-1} w_i ||\boldsymbol{x} - \tilde{g}_i(\boldsymbol{x}_i)||_2^2}_{\text{accumulated MSE loss of all layers}} + \underbrace{\lambda \sum_{i=0}^{L-1} \operatorname{ReLU}(-[\boldsymbol{\delta}]_i) + \lambda \sum_{i=0}^{nL-1} \operatorname{ReLU}(-[\boldsymbol{\tau}]_i)}_{\text{regularization term for the step-sizes and shrinkage thresholds}}, \tag{44}$$

where w_i denotes the importance weight of choice for the output of each layer, $\lambda > 0$, $[\boldsymbol{\delta}]_i = \delta_i$, and $\boldsymbol{\tau} = [\boldsymbol{\tau}_0^T, \ldots, \boldsymbol{\tau}_{L-1}^T]$. Note that as the information flows through the network, one expects that as we progress layer by layer, the reconstruction shows improvement. A reasonable weighting scheme for designing the importance weights w_i is to gradually increase the importance weights as we proceed through the layers. In this work, we consider a logarithmic weighting scheme, i.e. $w_i = \log(i+2)$, for $i \in \{0, \ldots, L-1\}$. Moreover, in training the autoencoders based on the BIHT algorithm, we exclude the last term in (44) as there is no shrinkage thresholds required for these models.

As for the training procedure, our numerical investigations showed that an incremental learning approach is most effective for training of the proposed networks. The details of the incremental learning method that we employed are as follows. During the l-th increment round (for $l \in \{0, \ldots, L-1\}$), we seek to optimize the cost function $\mathcal{G}_l(\boldsymbol{x}, \hat{\boldsymbol{x}})$ by learning the set of parameters $\boldsymbol{\Upsilon}_l = \boldsymbol{\Upsilon}_1 \cup \{\phi_i\}_{i=0}^{l}$. At each round l, we perform a batch learning with mini-batches of size B. After finishing the l-th round of training, the $(l+1)$-th layer will be added to the network, and the objective function will be changed to $\mathcal{G}_{l+1}(\boldsymbol{x}, \hat{\boldsymbol{x}})$. Next, the entire network will go through another batch-learning phase. Interestingly, in this method of training, the learned parameters from the l-th round $\boldsymbol{\Upsilon}_l$ will be used as the initial values of the same parameters in the next round.

IV. Numerical Results

In this section, we present various simulation results to investigate the performance of the proposed one-bit compressive AEs and to further show the effectiveness of our training. For training purposes, we randomly generate K-sparse signals of length $n = 128$, i.e. $\boldsymbol{x} \in \mathbb{R}^{128}$ where the non-zero elements are sampled from $\mathcal{N}(0,1)$. Furthermore, we fix the total number of layers of the decoder function to $L = 30$; equivalent of performing only 30 optimization iterations of the form (36), (38), (42), and (43). As for the sensing matrix (to be learned), we assume $\boldsymbol{\Phi} \in \mathbb{R}^{512 \times 128}$. The results presented here are averaged over 128 realizations of the system parameters. Similar to [2], we consider the case that $m > n$, due to the focus of this study on one-bit sampling where usually a large number of one-bit samples are available, as opposed to the usual infinite-precision CS settings.

The proposed one-bit CS AEs are implemented using the PyTorch library [40]. The Adam optimizer [41] with a learning rate of 10^{-3} is utilized for optimization of parameters of the proposed deep architectures. Due to the importance of reproducible research, we have made all the codes implemented publicly available along with this paper.[1]

As it was previously discussed in Sec. III-D, we employ an incremental batch-learning approach with mini-batches of size 64 at each round $l < 30$, and a total number of 200 epochs per round. For training of the the proposed AEs that are based on the RFPI iterations, i.e., the L-RFPI and LG-RFPI deep architectures, we uniformly sample the sparsity level of the source signal from the set $K \in \{16, 24, 32\}$ for each training point in the mini-batch. We evaluate the performance of the proposed methods on target signals with $K \in \{16, 32\}$, as well as $K \in \{8, 40\}$ (which was not presented to the network during the training phase). Moreover, due to the fact that the BIHT method and the corresponding one-bit AEs (L-BIHT and LG-BIHT) require the knowledge of the sparsity level of the source signal a priori, there is no need to train the network on various sparsity levels; i.e., the corresponding deep architectures can be trained for a particular K. Hence, for the L-BIHT and LG-BIHT deep architectures, we train the network for source signals with $K = 16$, and evaluate the performace of the resulted networks on $K \in \{16, 24\}$.

In the sequel, we refer to $\boldsymbol{s}^{\text{d}} = \boldsymbol{s}/\|\boldsymbol{s}\|_2$ as the *normalized* version of the vector $\boldsymbol{s}$. In all scenarios, in order to have a fair comparison between the algorithms, the initial starting point $\boldsymbol{z}_0$ of the optimization algorithms are the same.

Performance of the proposed L-RFPI AE: In this part, we investigate the performance of the proposed L-RFPI-based AE in recovering the normalized source signal $\boldsymbol{x}$, i.e., recovering $\boldsymbol{x}_i^{\text{d}}$.

Fig. 3 illustrates Mean Squared Error (MSE) for normalized version of the recovered signal $\hat{\boldsymbol{x}}_i^{\text{d}}$ versus total number of optimization iterations i, for $i \in \{0, \ldots, 29\}$, and for sparsity levels (a) $K = 8$, (b) $K = 16$, (c) $K = 32$, and (d) $K = 40$. We compare the performance of the proposed L-RFPI algorithm with the standard RFPI iterations in (8a)-(8d), in the following scenarios:

- *Case 1*: The RFPI algorithm with a randomly generated sensing matrix whose elements are i.i.d. and sampled from $\mathcal{N}(0,1)$, and fixed values for δ and α.
- *Case 2:* The RFPI algorithm where the learned $\boldsymbol{\Phi}$ is utilized, and the values for δ and α are fixed as in the previous case.
- *Case 3:* The RFPI algorithm with a randomly generated $\boldsymbol{\Phi}$ (same as Case 1), however, the learned shrinkage thresholds vector $\{\boldsymbol{\tau}_i\}_{i=0}^{L-1}$ is utilized with a fixed step-size.
- *Case 4:* The proposed one-bit L-RFPI AE method corresponding to the iterations of the form (36a)-(36d), with learned $\boldsymbol{\Phi}$, $\{\delta_i\}_{i=0}^{L-1}$, and $\{\boldsymbol{\tau}_i\}_{i=0}^{L-1}$.

To have a fair comparison, we fine-tuned the parameters of the standard RFPI method (Case 1), i.e., the step-size δ and the shrinkage threshold α, using a grid-search method. It can be seen from Fig. 3 that in all cases of $K \in \{8, 16, 32, 40\}$, the proposed L-RFPI method demonstrates a significantly better performance than that of the RFPI algorithm (described in Case 1)—an improvement of ~ 10 times in MSE outcome. Furthermore, the effectiveness of the learned $\boldsymbol{\Phi}$ (Case 2), and

[1]The code is also available at: https://github.com/skhobahi/deep1bitVAE

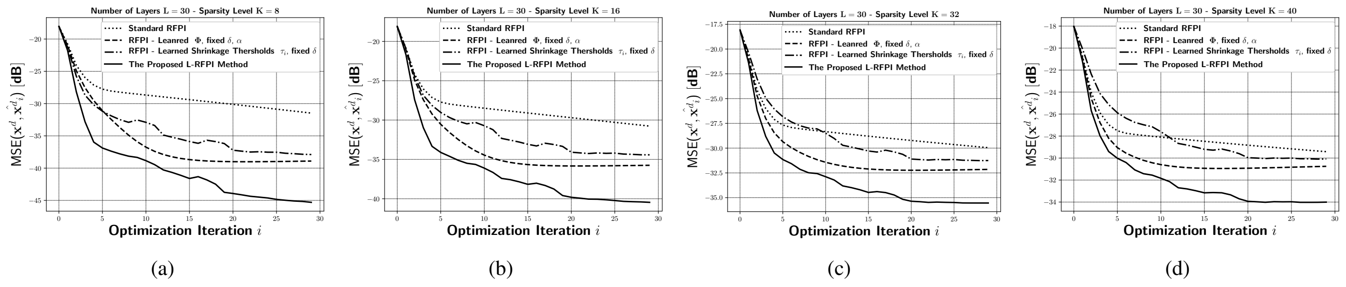


Fig. 3. The performance of the proposed L-RFPI method compared to the RFPI algorithm for sparsity levels: (a) $K = 8$, (b) $K = 16$, (c) $K = 32$, and (d) $K = 40$.

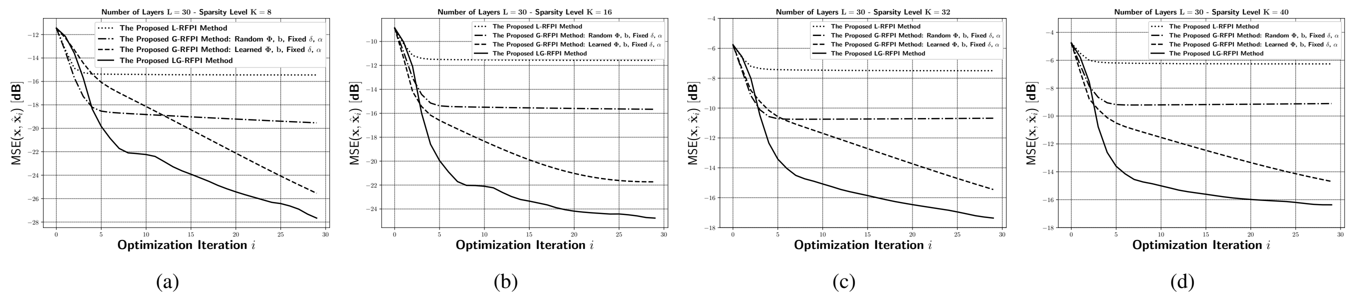


Fig. 4. The performance of the proposed LG-RFPI AE and the proposed G-RFPI method in recovering the amplitude information of the K- sparse signals for sparsity levels: (a) $K = 8$, (b) $K = 16$, (c) $K = 32$, and (d) $K = 40$.

the learned $\{\boldsymbol{\tau}_i\}$ (Case 3) compared to the base algorithm (Case 1), are clearly evident, as both algorithms with learned parameters significantly outperform the original RFPI. Finally, although we trained the network for $K \in \{16, 24, 32\}$ sparse signals, it still shows very good generalization properties even for $K \in \{8, 40\}$ (see Fig. 3 (a) and (d)). This is presumably due to the fact that the proposed L-RFPI-based AE is a *hybrid* model-based data-driven approach that exploits the existing domain knowledge of the problem as well as the available data at hand. Furthermore, note that the proposed method achieves a high accuracy very quickly and does not require solving (7) for several instances as opposed to the original RFPI algorithm—thus showing great potential for usage in real-time applications.

Performance of the proposed LG-RFPI AE :
Next, we investigate the performance of the proposed LG-RFPI AE (see Eqs. (42a)-(42c)) and the G-RFPI algorihtm (see Eqs. (22a)-(22c)) that we specifically designed for incorporating arbitrary quantization thresholds at data acquisiton. We investigate the performance of the proposed method in both cases of recovering the amplitude information as well as the normalized signal.

Fig. 4 illustrates the MSE between the source signal $\boldsymbol{x}$ and the recovered signal $\hat{\boldsymbol{x}}_i$ versus total number of optimization iterations i, for $i \in \{0, \ldots, 29\}$, and for sparsity levels (a) $K = 8$, (b) $K = 16$, (c) $K = 32$, and (d) $K = 40$. Similar to the previous case, we consider the following scenarios:

- *Case 1*: The proposed G-RFPI algorithm with a randomly generated sensing matrix and quantization threhsolds vector, whose elements are i.i.d. and sampled from $\mathcal{N}(0, 1)$, and fixed values for δ and α.
- *Case 2*: The proposed G-RFPI algorithm where the learned sensing matrix $\boldsymbol{\Phi}$ and quantization thresholds vector $\boldsymbol{b}$ are utilized, and the values for δ and α are fixed as in the previous case.
- *Case 3*: The proposed one-bit LG-RFPI AE method corresponding to the iterations of the form (42a)-(42c), with the learned $\boldsymbol{\Phi}$, $\boldsymbol{b}$, $\{\delta_i\}_{i=0}^{L-1}$, and $\{\boldsymbol{\tau}_i\}_{i=0}^{L-1}$.

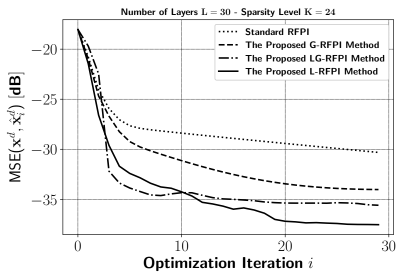


Fig. 5. The performance of the proposed LG-RFPI AE and the proposed G-RFPI method in recovering the normalized K-sparse signals for sparsity level $K = 24$.

Note that the focus of this part is on recovering the amplitude information of the underlying K-sparse signal by means of using arbitrary quantization thresholds. Although the RFPI method and the proposed L-RFPI AE can only recover the normalized signal $\boldsymbol{x}^d = \boldsymbol{x}/\|\boldsymbol{x}\|$, we further provide the performance of the L-RFPI method (that significantly outperforms the RFPI method) in recovering the amplitude information for comparison purposes. It can be observed from Fig. 4 that the proposed G-RFPI algorithm with randomly generated sensing matrix and quantization thresholds (Case 1) provides good accuracy in recovering the amplitude information of the true signal for sparsity levels $K \in \{8, 16, 32, 40\}$. This

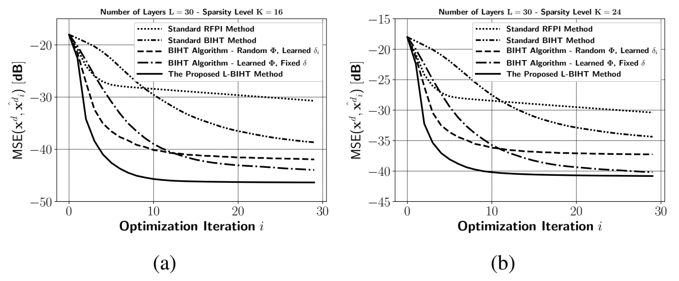


Fig. 6. The performance of the proposed L-BIHT method compared to the base-line BIHT algorithm for sparsity levels: (a) $K = 16$ and (b) $K = 24$.

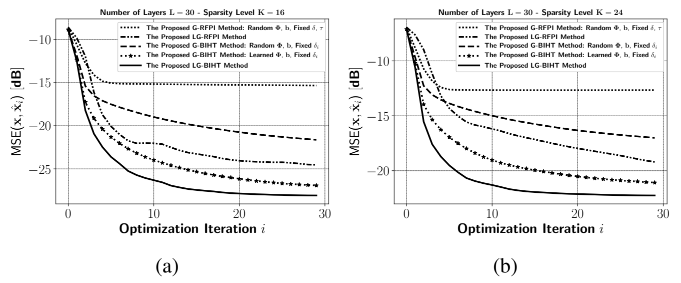


Fig. 7. The performance of the proposed G-BIHT and the corresponding LG-BIHT AE in recovering the amplitude information of the signal for sparsity levels: (a) $K = 16$, and (b) $K = 24$.

is in contrast to the RFPI algorithm and the corresponding L-RFPI AE where the amplitude information is lost due to zero quantization thresholds. More precisely, the proposed G-RFPI algorithm outperforms the RFPI and the L-RFPI algorithm in terms of recovering the amplitude information of the signal. One can observe that even with a randomly generated quantization thresholds (i.e., without learning them), the proposed G-RFPI method achieve a significantly lower MSE in terms of recovering the amplitude information of the source signal as compared to the RFPI and the proposed L-RFPI method. Hence, the proposed G-RFPI method can be used as an stand-alone algorithm for one-bit compressive sensing settings with non-zero quantization thresholds, where both finding the direction of the source signal and the amplitude information is of great interest. Next, we explore the effect of learning the distribution-specific (data-driven) sensing matrix and the quantization thresholds (Case 2). It is evident from Fig. 4 that compared to the vanilla G-RFPI method, one can significantly achieve a lower MSE in terms of recovering the amplitude information by learning a proper sensing matrix and the quantization thresholds and utilizing them during the data-acquisition process. Finally, it can be seen from Fig. 4 that the proposed LG-RFPI AE (Case 3) significantly outperforms its counterparts by achieving a much lower MSE very quickly. Moreover, the proposed LG-RFPI AE shows strong generalization properties for unseen sparsity levels $K \in \{8, 40\}$ (see Fig. 4 (b) and (d)). The fact that such architectures show great performance in generalization is due to the model-driven nature of the proposed deep networks.

We conclude this part by comparing the performance of the proposed LG-RFPI, G-RFPI, and L-RFPI AEs in recovering the normalized version of the signal $\boldsymbol{x}$. Fig. 5 illustrates the MSE between the normalized source signal and the recovered signal versus number of iterations i, i.e. MSE($\boldsymbol{x}^d$,$\hat{\boldsymbol{x}}_i^d$), for a sparsity level of $K = 24$. It can be observed from Fig. 4 that the proposed methods outperform the standard RFPI iterations and achieve a high accuracy in recovering $\boldsymbol{x}^d$. Moreover, the proposed L-RFPI AE shows a slightly better performance than that of the LG-RFPI method. This is presumably due to the fact that the L-RFPI iterations and the corresponding deep architecture are specifically designed and tuned for recovering the normalized source signal while the proposed G-RFPI and LG-RFPI algorithms are designed for recovering the amplitude information of the source signal. Nevertheless, the MSE difference between the LG-RFPI and L-RFPI methods in recovering $\boldsymbol{x}^d$ is negligible, and hence, in a non-zero quantization thresholds setting, it is beneficial to use the proposed LG-RFPI AE as it shows significant improvement in the performance of recovering the amplitude information while maintaining a high performance in recovering $\boldsymbol{x}^{\mathrm{d}}$ as well.

Performance of the proposed L-BIHT AE:

In this part, we investigate the performance of the proposed L-BIHT AE, and compare our results with the standard BIHT algorithm. Note that similar to the RFPI method and the proposed L-RFPI AE, the BIHT algorithm considers $\boldsymbol{b} = \mathbf{0}$ at the time of data acquisition. Hence, we investigate the performance of the proposed method in recovering the normalized source signal, i.e. $\boldsymbol{x}^{\mathrm{d}}$. In particular, we provide the simulation results for the following cases:

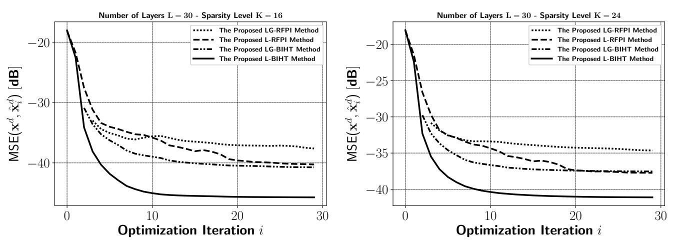
• *Case 1*: The BIHT algorithm with a randomly generated sensing matrix whose elements are i.i.d. and sampled from $\mathcal{N}(0, 1)$, and fixed value for δ.

• *Case 2*: The BIHT algorithm with a randomly generated $\mathbf{\Phi}$ (same as Case 1); however, learned gradient step-sizes δ_i are used at each iteration i.

• *Case 3*: The BIHT algorithm where the learned $\mathbf{\Phi}$ is utilized and the value for the step-size δ is fixed as in Case 1.

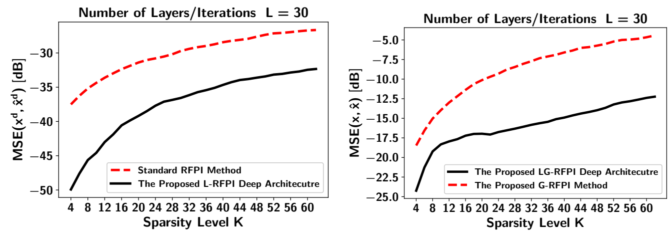
• *Case 4*: The proposed one-bit L-BIHT AE method corresponding to the iterations of the form (38a)-(38b), with the learned $\mathbf{\Phi}$ and $\{\delta_i\}_{i=0}^{L-1}$.

Fig. 6 demonstrates the MSE between normalized source signal $\boldsymbol{x}^d$, and the recovered signal $\hat{\boldsymbol{x}}_i^{\mathrm{d}}$ versus the number of optimization iterations i, for signals with sparsity levels (a) $K = 16$ and (b) $K = 24$. Note that for learning the parameters of the proposed L-BIHT algorithm, we trained the corresponding deep architecture on the sparsity level $K = 16$, and we check the generalization performance of the learned parameters for the case of $K = 24$. It can be seen from Fig. 6 that in both cases of $K \in \{16, 24\}$ the proposed L-BIHT algorithm demonstrates a significantly better performance than that of the standard BIHT algorithm (Case 1). Moreover, the effectiveness of the learned step-sizes $\{\delta_i\}_{i=0}^{L-1}$ (Case 2), and the learned sensing matrix $\mathbf{\Phi}$ (Case 3) compared to the base-line vanilla BIHT algorithm (Case 1) are evident. In particular, the learned step-sizes (Case 2) results in a fast descent while the learned $\mathbf{\Phi}$ (Case 3) leads to a lower MSE compared to Case 2. In addition, we provided the performance of the standard RFPI algorithm for comparison purposes. It



(a) (b)

Fig. 8. The performance of the proposed G-BIHT and the corresponding LG-BIHT AE in recovering the normalized signal, i.e. $\boldsymbol{x}^{\mathrm{d}}$, for sparsity levels: (a) $K = 16$, and (b) $K = 24$.



(a) (b)

Fig. 9. The generalization performance of (a) the proposed L-RFPI deep architecture compared to the standard RFPI method in recovering the direction information, and (b) the proposed LG-RFPI deep architecture and the proposed G-RFPI method in recovering the amplitude information of the K-sparse signals for sparsity levels $K \in \{4, 5, 6, \cdots, 64\}$.

can be seen from Fig. 6 that the BIHT algorithm with and without the learned parameters achieves a better accuracy in recovering the direction of the source signal compared to the RFPI method. Also, a comparison between Fig. 6 (a) and Fig. 3 (b) reveals the fact that the proposed L-BIHT AE demonstrates a far better performance than that of the proposed L-RFPI AE. This is due to the fact that the BIHT algorithm and the corresponding proposed L-BIHT AE, exploits the knowledge of the sparsity level K of the source signal (note the mapping function $\mathcal{H}_K$ used in (38a) and (24b)). One can further observe that even for the unseen case of $K = 24$, the proposed method generalizes very well and maintains its accuracy. This is due the model-driven nature of the proposed L-BIHT AE architecture. It is worth mentioning that it can be observed from Fig. 6 that the proposed L-BIHT method converges very fast (in 10 iterations), achieving a high accuracy—making it a great candidate for real-time applications. Of course, the trade-off between using the L-RFPI and L-BIHT is implicit in the knowledge of the sparsity level of the signal. For applications where K is known beforehand, the proposed L-BIHT can be used in that it shows higher accuracy compared to the other methods. However, the L-RFPI methodology is more flexible as it does not require knowing the sparsity level of the signal a priori.

Performance of the proposed LG-BIHT AE:

Finally, we investigate the performance of the proposed G-BIHT method (see Eqs. (33a)-(33b)) and the corresponding one-bit compressive LG-BIHT AE (see Eqs. (43a)-(43b)) that are specifically designed to handle non-zero quantization thresholds $\boldsymbol{b}$. In particular, we are interested in evaluating the performance of the proposed methods in recovering the amplitude information of the source K-sparse signal. Hence, for this part, we check the MSE between the true signal $\boldsymbol{x}$, and the recovered signal $\hat{\boldsymbol{x}}_i$ from the G-BIHT and LG-BIHT methods for each iteration i. In addition, we provide the results for recovering the direction of the source signal $\boldsymbol{x}^{\mathrm{d}}$ as well. Specifically, we provide the simulation results for the following cases:

• *Case 1*: The proposed G-BIHT algorithm with a randomly generated sensing matrix and quantization thresholds vector where the elements of both are i.i.d. and sampled from $\mathcal{N}(0, 1)$, and fixed value for $\{\delta_i\}_{i=0}^{L-1}$.

• *Case 2*: The proposed G-BIHT algorithm, where the learned sensing matrix $\boldsymbol{\Phi}$ and quantization thresholds $\boldsymbol{b}$ are utilized and the values for $\{\delta_i\}_{i=0}^{L-1}$ are fixed as in the previous case.

• *Case 3*: The proposed one-bit LG-BIHT AE method corresponding to the iterations of the form (43a)-(43b), with learned $\boldsymbol{\Phi}$, $\boldsymbol{b}$ and $\{\delta_i\}_{i=0}^{L-1}$.

Fig. 7 illustrates the MSE between the true signal $\boldsymbol{x}$ and the recovered signal $\hat{\boldsymbol{x}}_i$ versus optimization iteration i for sparsity levels (a) $K = 16$ and (b) $K = 24$. We further provide the numerical results for the proposed LG-RFPI AE and the proposed G-RFPI iterations for comparison. It can be seen from Fig. 7 that the proposed G-BIHT algorithm with randomly generated latent-variables (Case 1) significantly outperforms its G-RFPI counterpart, and achieves a high accuracy very quickly. On the other hand, the proposed LG-RFPI still achieves a lower MSE compared to the vanilla G-RFPI method. In addition, a comparison between the performance of the proposed G-BIHT algorithm with learned $\boldsymbol{\Phi}$ and $\boldsymbol{b}$ (Case 2) and the proposed LG-RFPI AE and vanilla G-BIHT (Case 1) reveals the effectiveness of the learned parameters and the power of the proposed G-BIHT algorithm. Namely, by utilizing only the learned $\boldsymbol{\Phi}$ and $\boldsymbol{b}$ and by using a fixed step size for the G-BIHT algorithm, one can achieve a superior performance than that of the LG-RFPI (where all of the learned variables are in use) and the vanilla G-BIHT method. Finally, it can be observed from 6(a)-(b) that the proposed LG-BIHT algorithm (Case 3) significantly outperforms the other methods as it achieves a much lower MSE very quickly, specifically, compared to the proposed LG-RFPI AE. The superior performance of the LG-BIHT algorithm and the corresponding LG-BIHT AE is due the fact that we are exploiting the knowledge of the sparsity level K present in the signal. As discussed before, if the sparsity level is known a priori, it is beneficial to use either the G-BIHT algorithm (when one do not wish to perform any learning) or the proposed LG-BIHT methodology. It is worth mentioning that similar to the previously investigated methods, the proposed LG-BIHT generalizes very well for $K = 24$ (see Fig. 7(b)) even though the sparsity level was not revealed to the network during the training phase.

Fig. 8 demonstrate the MSE between the direction of the source signal, i.e. $\boldsymbol{x}^{\mathrm{d}}$, and the recovered direction $\hat{\boldsymbol{x}}^{\mathrm{d}}$ versus optimization iteration i, for sparsity levels of (a) $K = 16$ and (b) $K = 24$. It can be seen from Fig. 8 that the proposed LG-BIHT method outperforms both the LG-RFPI method, and furthermore, it achieves a similar MSE to that

TABLE I

Metrics	**L-BIHT**	**LG-BIHT**	**L-RFPI**	**LG-RFPI**	**Random $\boldsymbol{\Phi}$**
$\mu(\boldsymbol{\Phi})$	0.0415	0.0904	0.0760	0.0852	0.2444
$\|\boldsymbol{M}-\boldsymbol{I}\|_F^2$	2.1716	8.7711	6.6117	8.0244	31.3032

of the proposed L-RFPI method. However, the convergence of LG-BIHT is much faster than that of the L-RFPI method. Furthermore, the proposed L-BIHT algorithm still achieves a superior performance than that of the other methods both in terms of convergence speed and accuracy. This is presumably due to the fact that the L-BIHT method is specifically designed and learned to have a high accuracy in finding normalized true signal $\boldsymbol{x}^{\mathrm{d}}$.

Generalization of the LG-RFPI and L-RFPI Methods: In this part, we analyze the generalization performance of the proposed LG-RFPI and L-RFPI deep architectures. We consider the same simulation setup as in the previous cases, i.e. both architectures are assumed to have $L = 30$ layers. We performed the training of both architectures on a dataset consisting of K-sparse signals where the sparsity level is sampled uniformly from the set $K \in \{4, 8, 12, 16, 20\}$, and we evaluate the generalization performance of both architectures on K-source signals with sparsity levels $K \in \{4, 5, 6, \cdots, 64\}$.

Fig. 9(a) demonstrates the MSE between the direction of the true source signal, i.e. $\boldsymbol{x}^{\mathrm{d}}$, and the recovered direction $\hat{\boldsymbol{x}}^{\mathrm{d}}$ versus the sparsity level $K = \|\boldsymbol{x}\|_0$ for the proposed L-RFPI deep architecture. Moreover, Fig. 9(b) demonstrates the MSE between the the true source signal $\boldsymbol{x}$ and the recovered signal $\hat{\boldsymbol{x}}$ versus the sparsity level K for the proposed LG-RFPI deep architecture, and the proposed base-line G-RFPI algorithm (provided here for comparison purposes), both specifically designed to recover the amplitude information of the source signal. It can be seen from both Figs. 9(a) and 9(b) that the performance of the proposed L-RFPI and LG-RFPI methodologies is far superior to that of the standard model-based RFPI methods and the proposed G-RFPI algorithm across all sparsity levels. Interestingly, although the proposed deep architectures have been trained only on the sparsity levels $K \in \{4n\}_{n=1}^{5}$, they generalized very well to higher sparsity levels as well, while outperforming the model-based algorithms. Such a good generalization performance is expected due to the model-based nature of the proposed architecture. This is in contrast to the conventional black-box deep learning models where the generalization performance usually degrades significantly as the input deviates from the distribution of the data points considered for training.

Coherence analysis of the learned sensing matrices: In this part, we give a detailed analysis of the quality of the learned sensing matrices obtained by employing each of the proposed methodologies.

In order to quantify the quality of the learned task-specific sensing matrices, we make use of the mutual coherence metric defined in (2) as a figure of merit for the learned sensing matrices by the proposed methodologies. Fig. 10 demonstrates the distribution of the mutual coherence coefficients of the sensing matrices obtained by the proposed deep architecture as well as the mutual coherence coefficients of a randomly generated sensing matrix (used in the previous numerical results). Furthermore, a detailed numerical analysis of $\mu(\boldsymbol{\Phi})$ and the Gram matrix $\boldsymbol{M}$ is provided in Table I. It can be clearly observed from Fig. 10 and Table I that the proposed methodologies result in task-specific sensing matrices with considerably lower coherence coefficients as compared to that of a randomly generated one. In Particular, the mutual coherence $\mu(\boldsymbol{\Phi})$ is significantly lower for the proposed methodologies as compared to a purely random $\boldsymbol{\Phi}$. These observations also support the superior performance of the proposed methodologies in terms of signal reconstruction accuracy. In addition, the Gram matrix associated with the learned sensing matrices admits a structure far closer to the identity as compared to a random matrix (see Table I). As it was previously mentioned, the design of sensing matrices with low mutual coherence is the subject of numerous works in various fields and directly carrying out such a design is a difficult task in general. Interestingly, *although the framework does not rely on explicit regularization or a tailored optimization objective to reduce the mutual coherence, the proposed methodology learns sensing matrices with a very low mutual coherence, i.e., the proposed methodology is implicitly biased towards learning high-quality task-specific sensing matrices*. This is presumably due to the model-based nature of the proposed deep architectures.

V. Conclusion

In this paper, we considered the problem of one-bit compressive sensing and proposed a novel hybrid *model-driven* and *data-driven* autoencoding scheme that allows us to *jointly* learn the parameters of the measurment module (i.e., the sensing matrix and the quantization thresholds) and the latent-variables of the decoder (estimator) function, based on the underlying distribution of the data. In broad terms, we proposed a novel methodology that combines the traditional compressive sensing techniques with model-based deep learning—resulting in interpretable deep architectures for the problem of one-bit compressive sensing. In addition, the proposed method can handle the recovery of the amplitude information of the signal using the learned and optimized quantization thresholds. Our simulation results demonstrated that the proposed hybrid methodology is superior to the state-of-the-art methods for the problem of one-bit CS in terms of both computional efficiency and accuracy.

References

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

[2] P. T. Boufounos and R. G. Baraniuk, "1-bit compressive sensing," in *2008 42nd Annual Conference on Information Sciences and Systems*, March 2008, pp. 16–21.

[3] J. N. Laska, Z. Wen, W. Yin, and R. G. Baraniuk, "Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5289–5301, Nov 2011.

[4] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *IEEE Transactions on Information Theory*, vol. 59, no. 4, pp. 2082–2102, April 2013.

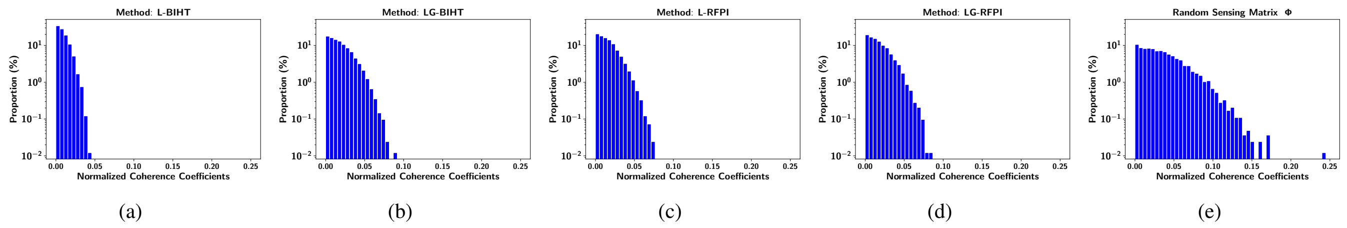


Fig. 10. Distribution of the mutual coherence coefficients associated with the sensing matrix learned through (a) the proposed L-BIHT method (b) the proposed L-GBIHT method, (c) the proposed L-RFPI and (d) the proposed L-GBRFPI method, where (e) demonstrates the distribution of the mutual coherence coefficient of a random generated sensing matrix $\mathbf{\Phi}$ used in the numerical sections. It can be observed that the proposed methodologies implicitly learn sensing matrices with very low mutual coherence as compared to that of a randomly generated $\mathbf{\Phi}$.

[5] A. Movahed, A. Panahi, and G. Durisi, "A robust RFPI-based 1-bit compressive sensing reconstruction algorithm," in *2012 IEEE Information Theory Workshop*, Sep. 2012, pp. 567–571.

[6] M. Yan, Y. Yang, and S. Osher, "Robust 1-bit compressive sensing using adaptive outlier pursuit," *IEEE Transactions on Signal Processing*, vol. 60, no. 7, pp. 3868–3875, July 2012.

[7] P. Xiao, B. Liao, and J. Li, "One-bit compressive sensing via Schur-concave function minimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 16, pp. 4139–4151, Aug 2019.

[8] K. Knudson, R. Saab, and R. Ward, "One-bit compressive sensing with norm estimation," *IEEE Transactions on Information Theory*, vol. 62, no. 5, pp. 2748–2758, 2016.

[9] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[10] A. S. Bandeira, E. Dobriban, D. G. Mixon, and W. F. Sawin, "Certifying the restricted isometry property is hard," *IEEE transactions on information theory*, vol. 59, no. 6, pp. 3448–3450, 2013.

[11] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[12] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[13] S. Foucart and H. Rauhut, "A mathematical introduction to compressive sensing," *Bull. Am. Math*, vol. 54, pp. 151–165, 2017.

[14] M. Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*. New York, NY: Springer, 2010.

[15] ——, "Optimized projections for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[16] T. Strohmer and R. W. Heath, "Grassmannian frames with applications to coding and communication," *Applied and Computational Harmonic Analysis*, vol. 14, no. 3, pp. 257–275, 2003.

[17] Y. C. Eldar and G. D. Forney, "Optimal tight frames and quantum measurement," *IEEE Transactions on Information Theory*, vol. 48, no. 3, pp. 599–610, 2002.

[18] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3475–3506, 2012.

[19] H. Hu, M. Soltanalian, P. Stoica, and X. Zhu, "Locating the few: Sparsity-aware waveform design for active radar," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 651–662, 2016.

[20] Z. Li, W. Xu, X. Zhang, and J. Lin, "A survey on one-bit compressed sensing: theory and applications," *Frontiers of Computer Science*, vol. 12, no. 2, pp. 217–230, Apr 2018. [Online]. Available: https://doi.org/10.1007/s11704-017-6132-7

[21] Y. Plan and R. Vershynin, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Transactions on Information Theory*, vol. 59, no. 1, pp. 482–494, 2012.

[22] U. S. Kamilov, A. Bourquard, A. Amini, and M. Unser, "One-bit measurements with adaptive thresholds," *IEEE Signal Processing Letters*, vol. 19, no. 10, pp. 607–610, 2012.

[23] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," *arXiv preprint arXiv:1601.06892*, 2016.

[24] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deep fully-connected networks for video compressive sensing," *Digital Signal Processing*, vol. 72, pp. 9 – 18, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1051200417302130

[25] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep. 2015, pp. 1336–1343.

[26] N. Shlezinger, Y. C. Eldar, and M. R. Rodrigues, "Hardware-limited task-based quantization," *IEEE Transactions on Signal Processing*, vol. 67, no. 20, pp. 5223–5238, 2019.

[27] Y. Liao, N. Farsad, N. Shlezinger, Y. C. Eldar, and A. J. Goldsmith, "Deep neural network symbol detection for millimeter wave communications," *arXiv preprint arXiv:1907.11294*, 2019.

[28] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Viterbinet: A deep learning based Viterbi algorithm for symbol detection," *arXiv preprint arXiv:1905.10750*, 2019.

[29] Y. Wu, M. Rosca, and T. Lillicrap, "Deep compressed sensing," *arXiv preprint arXiv:1905.06723*, 2019.

[30] J.-T. Chien and C.-H. Lee, "Deep unfolding for topic models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 318–331, 2017.

[31] S. Khobahi, N. Naimipour, M. Soltanalian, and Y. C. Eldar, "Deep signal recovery with one-bit quantization," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 2987–2991.

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

[33] S. Wisdom, J. Hershey, J. Le Roux, and S. Watanabe, "Deep unfolding for multichannel source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 121–125.

[34] O. Solomon, R. Cohen, Y. Zhang, Y. Yang, Q. He, J. Luo, R. J. G. van Sloun, and Y. C. Eldar, "Deep unfolded robust PCA with application to clutter suppression in ultrasound," *IEEE transactions on medical imaging*, 2019.

[35] S. Khobahi, A. Bose, and M. Soltanalian, "Deep radar waveform design for efficient automotive radar sensing," in *2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE, 2020, pp. 1–5.

[36] Y. Plan and R. Vershynin, "One-bit compressed sensing by linear programming," *Communications on Pure and Applied Mathematics*, vol. 66, no. 8, pp. 1275–1297, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21442

[37] Y. Shen, J. Fang, H. Li, and Z. Chen, "A one-bit reweighted iterative algorithm for sparse signal recovery," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 5915–5919.

[38] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[39] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and computational harmonic analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[40] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.