DEEP ONE-BIT COMPRESSIVE AUTOENCODING

Shahin Khobahi, Arindam Bose and Mojtaba Soltanalian*

ECE Department, University of Illinois at Chicago, Chicago, IL 60607

ABSTRACT

Parameterized mathematical models play a central role in understanding and design of complex information systems. However, they often cannot take into account the intricate interactions innate to such systems. On the contrary, purely data-driven approaches do not need explicit mathematical models for data generation and have a wider applicability at the cost of interpretability. In this paper, we consider the design of a one-bit compressive autoencoder, and propose a novel hybrid model-based and data-driven methodology that allows us to not only design the sensing matrix for one-bit data acquisition, but also allows for learning the latent-parameters of an iterative optimization algorithm specifically designed for the problem of one-bit sparse signal recovery. Our results demonstrate a significant improvement compared to state-of-the-art model-based algorithms.

Index Terms— Compressive sensing, low-resolution signal processing, deep unfolding, deep neural networks, autoencoders

1. INTRODUCTION

In the past decade, compressive sensing (CS) has shown significant potential in enhancing signal sensing and recovery performance with simpler hardware resources, 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 that is needed in Nyquist sampling [1]—where the signals are typically reconstructed by finding the sparsest solution of an under-determined system of equations using various available means.

Note that in a practical settings, each measurement needs to be digitized into finite-precision values for further processing and storage purposes, which inevitably introduces a quantization error. This error is generally dealt with as measurement noise possessing limited energy; an approach that does not perform well in extreme cases. 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, *i.e.*, $+1$ or -1 [2–7]. 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).

1.1. Relevant Prior Art

The current one-bit CS recovery algorithms generally exploit the *consistency principle*, which assumes that the element-wise product of the sparse signal and the corresponding measurement is always positive [2]. In [2], the authors introduce a regularizer based recovery algorithm called *renormalized fixed point iteration* (RFPI) using a convex barrier function as a regularizer for the consistency principle. We discuss the formulation of RFPI in Section 2 in details. 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. Compared to RFPI algorithm, RSS has three important advantages: provable convergence, improved consistency, and feasible performance [12]. 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 needs the sparsity level of the signal as input. Both RFPI and BIHT, however, perform very poorly in the presence of measurement noise, when *bit flips* occur. 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 robust against bit flips in the measurement vector.

We note that model-based algorithms, such as the ones discussed above, often do not take into account the intricate interactions and the latent-parameters innate to complex signal processing systems. Therefore, there has recently been a high demand for developing effective real-time signal processing algorithms that use the data to achieve improved performance [13–15]. In particular, the data-driven approaches relying on deep neural architectures such as convolutional neural networks [13], deep fully connected networks [14], and stacked denoising autoencoders [15], have been studied for sparse signal recovery in generic quantized CS setting. Such data-driven approaches do not need explicit mathematical models for data generation and have a wider applicability.

* Corresponding author (e-mail: *skhoba2@uic.edu*). This work was supported in part by U.S. National Science Foundation Grants CCF-1704401.

On the other hand, they lack the interpretability and trustability that comes with model-based signal processing techniques. The advantages associated with both model-based and data-driven methods show the need for developing approaches that enjoy the benefits of both frameworks [16, 17].

In this paper, we bridge the gap between the data-driven and model-based approaches in the one-bit CS area and propose a specialized, yet hybrid, methodology for the purpose of sparse signal recovery from one-bit measurements.

2. PROBLEM FORMULATION

In a one-bit CS scenario, the dynamics of the data acquisition process (*i.e.*, the encoder module) can be formulated as:

$$\text{Encoder Module:} \qquad \boldsymbol{r} = \operatorname{sign}\left(\boldsymbol{\Phi}\boldsymbol{x}\right), \tag{1}$$

where $\boldsymbol{\Phi}^{m\times n}$ denotes the sensing matrix, and $\boldsymbol{x} \in \mathbb{R}^n$ is assumed to be a K-sparse signal. Having the one-bit measurements of the form (1), one can pose the problem of sparse signal recovery from one-bit measurements $\boldsymbol{r}$ by solving the following non-convex program:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0, \;\; \text{s.t.} \;\; \boldsymbol{r} = \operatorname{sign}\left(\boldsymbol{\Phi}\boldsymbol{x}\right), \tag{2}$$

where the constraint in (2) is imposed to ensure a consistent reconstruction from the one-bit information. Further note that, the one-bit measurement consistency principle can be equivalently expressed as $\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x} \succeq \boldsymbol{0}$, where $\boldsymbol{R} = \operatorname{Diag}\left(\boldsymbol{r}\right)$ and $\succeq$ is an element-wise matrix inequality operator. Inspired by the CS literature, the above non-convex optimization problem can be further reformulated as a non-convex ℓ_1-minimization program on the unit sphere:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1, \;\; \text{s.t.} \;\; \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x} \succeq \boldsymbol{0}, \;\; \|\boldsymbol{x}\|_2 = 1, \tag{3}$$

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 significantly reduces the feasible set of the optimization problem, and second, it avoids the trivial solution of $\hat{\boldsymbol{x}} = \boldsymbol{0}$. There exists an extensive body of research on solving the above non-convex optimization problem (*e.g.*, see [2, 3, 5, 7, 18, 19], and the references therein). The most notable methods utilize a regularization term $\mathcal{R}(\boldsymbol{s})$ to enforce the consistency principle as a penalty term for the ℓ_2-objective function, viz.

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 + \alpha\mathcal{R}(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}), \;\; \text{s.t.} \;\; \|\boldsymbol{x}\|_2 = 1, \tag{4}$$

where $\alpha > 0$ is the penalty factor. In this paper, we build upon the work done in [2] and its proposed RFPI algorithm as a base-line to design the decoder function of the proposed one-bit compressive autoencoder (AE). In particular, we unfold the iterations of a renormalized fixed-point algorithm onto the layers of a 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 (*i.e.*, parameters of the RFP iterations) and the encoder (*i.e.*, the sensing matrix $\boldsymbol{\Phi}$) functions of the proposed compressive AE.

Let $\boldsymbol{y} = \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}$. In order to enforce the first constraint in (4), the RFPI algorithm utilizes the regularization term $\rho(\boldsymbol{y}) \triangleq \max\{-\boldsymbol{y}, \boldsymbol{0}\}$. Note that the function $\rho(\cdot)$ can be expressed in terms of the well-known ReLU function extensively used by the deep learning research community, *i.e.*, $\rho(\boldsymbol{y}) = \mathrm{ReLU}(-\boldsymbol{y})$. 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, the gradient step-size δ and a shrinkage thresholds α, at each iteration i, the estimated signal $\boldsymbol{x}_i$ is obtained using the following update steps:

$$\boldsymbol{d}_i = \nabla_{\boldsymbol{x}}\mathcal{R}(\boldsymbol{y})\big|_{\boldsymbol{x}=\boldsymbol{x}_{i-1}} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{x}_{i-1}\right), \tag{5a}$$

$$\boldsymbol{t}_i = \left(1 + \delta\boldsymbol{d}_i^T\boldsymbol{x}_{i-1}\right)\boldsymbol{x}_{i-1} - \delta\boldsymbol{d}_i, \tag{5b}$$

$$\boldsymbol{v}_i = \operatorname{sign}\left(\boldsymbol{t}_i\right) \odot \mathrm{ReLU}\left(|\boldsymbol{t}_i| - (\delta/\alpha)\boldsymbol{1}\right), \tag{5c}$$

$$\boldsymbol{x}_i = \frac{\boldsymbol{v}_i}{\|\boldsymbol{v}_i\|_2}. \tag{5d}$$

After the descent in (5a)-(5b), the update step in (5c) 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.

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 straightforward how to choose the fixed step-size and the shrinkage threshold, that may depend on the latent-parameters in the information 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 the next section, we show how this tuning can be done by learning from the data. In particular, we slightly modify and over-parameterize the above updating steps of the RFPI algorithm and unfold them onto the layers of a deep neural network, and define a decoder function based on the unfolded iterations, and seek to jointly learn the parameters of the proposed AE.

3. THE PROPOSED ONE-BIT COMPRESSIVE AUTOENCODER

We pursue the design of a novel *one-bit compressive sensing-based autoencoder* architecture that allows us to jointly design the parameters of both the encoder and the decoder module when one-bit quantizers are employed in the data acquisition process (*i.e.*, the encoding module) for a K-sparse input signal $\boldsymbol{x} \in \mathbb{R}^n$. Briefly speaking, an 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 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 functionals: *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 AE 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 $\boldsymbol{\Phi}$ 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 form (5a)-(5d) 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. In particular, we define the encoder module of the proposed AE as follows:

$$f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x}) = \tilde{\text{sign}}(\boldsymbol{\Phi}\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{\Upsilon}_1 = \{\boldsymbol{\Phi}\}$ denotes the set of parameters of the encoder function, and $\tilde{\text{sign}}(\boldsymbol{x}) = \tanh(c \cdot \boldsymbol{x})$, for some choice of $c > 0$ (c was set to 100 in numerical investigations). Note that we replaced the original $\text{sign}(\cdot)$ function with a smooth approximation of it based on the hyperbolic tangent function. The reason for such a replacement is that the $\text{sign}(\cdot)$ function is not continuous and its gradient is zero everywhere except at the origin, and hence, the use of it would cripple any stochastic gradient-based optimization method (used in backpropagation method in deep learning). As for the decoder function, define $g_{\phi_i} : \mathbb{R}^m \mapsto \mathbb{R}^n$ as follows:

$$g_{\phi_i}(\boldsymbol{z}; \boldsymbol{\Phi}, \boldsymbol{R}) = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}, \quad \text{with} \tag{7a}$$

$$\boldsymbol{v} = \tilde{\text{sign}}(\boldsymbol{t}) \odot \text{ReLU}(|\boldsymbol{t}| - \boldsymbol{\tau}_i), \tag{7b}$$

$$\boldsymbol{t} = \left(1 + \delta_i \boldsymbol{d}^T \boldsymbol{z}\right) \boldsymbol{z} - \delta_i \boldsymbol{d}, \tag{7c}$$

$$\boldsymbol{d} = -\left(\boldsymbol{R}\boldsymbol{\Phi}\right)^T \rho\left(\boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{z}\right), \tag{7d}$$

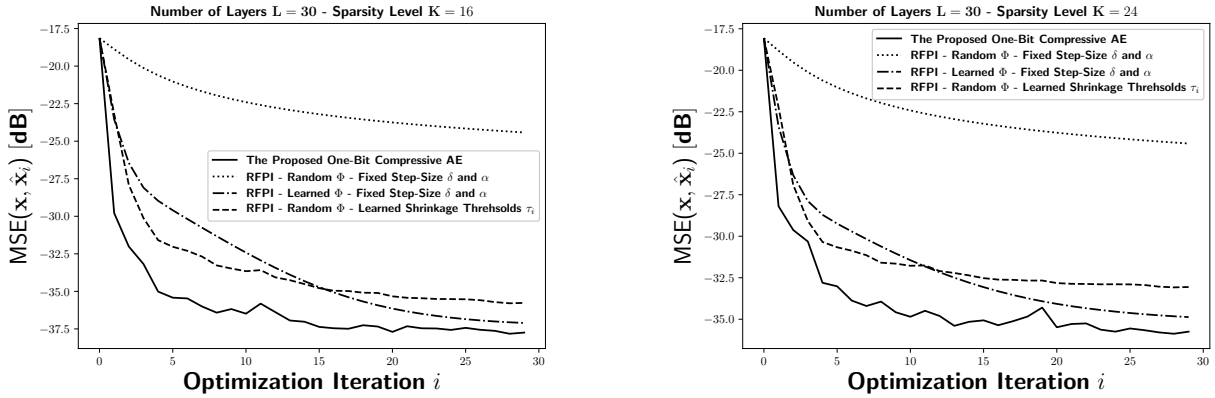
where $\phi_i = \{\boldsymbol{\tau}_i, \delta_i\}$ represents the parameters of the function g_{ϕ_i}, and $\boldsymbol{\tau}_i \in \mathbb{R}n$ denotes the sparsity inducing thresholds vector. Next, we define the proposed composite decoder function as follows:

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

where $\boldsymbol{\Upsilon}_2 = \{\phi_i\}_{i=0}^{L-1}$ represents the learnable (tunable) parameters of the decoder function. Note that we have over-parameterized the iterations of the RFPI algorithm by introducing a new variable $\boldsymbol{\tau}_i$ at each iteration for the sparsity inducing step (*i.e.*, Eq. (7b)). Moreover, in contrast with the original iterations, we have introduced a new step-size δ_i at each step of the iteration as well. Hence, 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.*, $\{\boldsymbol{\tau}_i, \delta_i\}_{i=0}^{L-1}$. Hence, the proposed decoder function is much more expressive than that of the iterations of RFPI algorithm.

3.1. Loss Function Characterization and Training

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.*, $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}(\boldsymbol{x})}\left\{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2\right\}$. 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, and hence, 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 all 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 $\{\delta_i\}_{i=0}^{L-1}$ 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 thresholds at each layer. With this in mind, we suggest the



(a) (b)

Fig. 1. The performance of the proposed method compared to the RFPI algorithm for sparsity levels: (a) $K = 16$, and (b) $K = 24$.

following loss function for training the proposed one-bit compressive AE. Let $\tilde{g}_i = g_{\phi_i} \circ g_{\phi_{i-1}} \circ \cdots \circ g_{\phi_0} \circ f^{\text{Encoder}}_{\boldsymbol{\Upsilon}_1}(\boldsymbol{x})$, and define the loss function for training as

$$\mathcal{G}(\boldsymbol{x};\hat{\boldsymbol{x}}) = \underbrace{\sum_{i=0}^{L-1} w_i ||\boldsymbol{x} - \tilde{g}_i(\boldsymbol{x}_i)||_2^2}_{\text{accumulated MSE loss of all layers}} + \underbrace{\lambda \sum_{i=0}^{L-1} \text{ReLU}(-[\boldsymbol{\delta}]_i) + \lambda \sum_{i=0}^{nL-1} \text{ReLU}(-[\boldsymbol{\tau}]_i)}_{\text{regularization term for the step-sizes and shrinkage thresholds}}, \tag{8}$$

where $\lambda > 0$, $[\boldsymbol{\delta}]_i = \delta_i$, and $\boldsymbol{\tau} = [\boldsymbol{\tau}_0^T, \ldots, \boldsymbol{\tau}_{L-1}^T]$.

4. NUMERICAL RESULTS

In this section, we present the simulation results that investigate the performance of the proposed one-bit compressive AE. For training purposes, we randomly generate K-sparse signals of length $n = 128$, on the unit sphere, *i.e.*, $\boldsymbol{x} \in \mathbb{R}^{128}$, and $\|\boldsymbol{x}\|_2 = 1$. 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 (7). As for the sensing matrix to be learned, we assume $\boldsymbol{\Phi} \in \mathbb{R}^{512\times128}$. The results presented here are averaged over 128 realizations of the system parameters. Although 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 CS settings.

The proposed one-bit CS AE is implemented using the PyTorch library [20]. The Adam algorithm [21] with a learning rate of 10^{-3} is utilized for optimization of parameters of the proposed AE. We perform a batch-learning approach with mini-batches of size 64, and a total number of 8000 epochs. For training of the the proposed AE, we fix $K = 16$, and evaluate the performance of the proposed method on target signals with $K = 16$, as well as $K = 24$ (which was not shown to the network during the training phase). In all scenarios, the initial starting point of the optimization algorithms are the same.

Fig. 1 illustrates MSE for the recovered signal versus total number of optimization iterations i, for $i = 0, \ldots, 29$, and for sparsity levels (a) $K = 16$ and (b) $K = 24$. We compare our algorithm with the RFPI iterations in (5a)-(5d), 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 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 CS AE method corresponding to the iterations of the form (7a)-(7d), with the learned $\boldsymbol{\Phi}$, $\{\delta_i\}_{i=1}^{L-1}$, and $\{\boldsymbol{\tau}_i\}_{i=0}^{L-1}$.

4.1. Discussion and Concluding Remarks

It can be seen from Fig. 1 that in both cases of $K \in \{16, 24\}$, the proposed 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 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 $K = 16$ sparse signals, it still shows good generalization properties even for $K = 24$ (see Fig. 1 (b)). This is presumably due to the fact that the proposed 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 high accuracy very quickly and does not require solving (4) for several instances as opposed to the original RFPI algorithm—thus showing great potential for usage in real-time applications.

5. REFERENCES

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

[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.

[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] L. Zhang, J. Yi, and R. Jin, "Efficient algorithms for robust one-bit compressive sensing," in *International Conference on Machine Learning*, 2014, pp. 820–828.

[8] S. Khobahi and M. Soltanalian, "Signal recovery from 1-bit quantized noisy samples via adaptive thresholding," in *52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 1757–1761.

[9] A. Ameri, A. Bose, J. Li, and M. Soltanalian, "One-bit radar processing with time-varying sampling thresholds," *IEEE Transactions on Signal Processing*, vol. 67, no. 20, pp. 5297–5308, 2019.

[10] K. Roth, H. Pirzadeh, A. Lee Swindlehurst, and J. A. Nossek, "A comparison of hybrid beamforming and digital beamforming with low-resolution ADCs for multiple users and imperfect CSI," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 3, pp. 484–498, 2018.

[11] S. J. Zahabi, M. M. Naghsh, M. Modarres-Hashemi, and J. Li, "One-bit compressive radar sensing in the presence of clutter," *IEEE Transactions on Aerospace and Electronic Systems*, 2019.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] S. Khobahi and M. Soltanalian, "Model-based deep learning for one-bit compressive sensing," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5292–5307, 2020.

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

[18] 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.

[19] 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.

[20] 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.

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