A Robust Deep Unfolded Network for Sparse Signal Recovery from Noisy Binary Measurements

Yuqing Yang[‡], Peng Xiao[‡⋆], Bin Liao[⋆], Nikos Deligiannis[‡†]

[‡]*Department of Electronics and Informatics, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium*
[⋆]*Department of Electronics and Information Engineering, Shenzhen University, China*
[†]*imec, Kapeldreef 75, B-3001 Leuven, Belgium*
Email: yuqing.yang@vub.be, pxiao@etrovub.be, binliao@szu.edu.cn, ndeligia@etrovub.be

Abstract—We propose a novel deep neural network, coined DeepFPC-ℓ_2, for solving the 1-bit compressed sensing problem. The network is designed by unfolding the iterations of the fixed-point continuation (FPC) algorithm with one-sided ℓ_2-norm (FPC-ℓ_2). The DeepFPC-ℓ_2 method shows higher signal reconstruction accuracy and convergence speed than the traditional FPC-ℓ_2 algorithm. Furthermore, we compare its robustness to noise with the previously proposed DeepFPC network—which stemmed from unfolding the FPC-ℓ_1 algorithm—for different signal to noise ratio (SNR) and sign-flipped ratio (flip ratio) scenarios. We show that the proposed network has better noise immunity than the previous DeepFPC method. This result indicates that the robustness of a deep-unfolded neural network is related with that of the algorithm it stems from.

Index Terms—1-bit compressed sensing, Deep unfolding, Gaussian noise, sign flipping.

I. INTRODUCTION

Compressive sensing (CS) is a low-rate signal acquisition method for sparse signals [1]. Provided that a signal is sparse, or has a sparse representation in a set of bases or learned dictionary, CS can recover it with a sampling rate notably lower than what the Shannon-Nyquist theory dictates. CS has found many applications in image processing [2], wireless communications [3] and direction-of-arrival (DOA) estimation [4], to name a few. In practice, the measurements of CS must be quantized to discrete values; 1-bit CS [5], in particular, considers 1-bit quantized measurements. Compared with traditional CS, 1-bit CS can further reduce the sampling and system complexity, i.e., by realizing the sampling with a simple comparator. Yet, a challenge for the 1-bit CS problem is how to solve the related optimization problem with high accuracy.

A variety of algorithms have been proposed to solve the 1-bit CS problem. The fixed point continuation (FPC) method in [6] can be deployed by computing the one-sided ℓ_1- or ℓ_2-norm penalty on the unit sphere $\|\boldsymbol{x}\|_2 = 1$ and applying a re-normalization step, resulting in the FPC-based algorithms FPC-ℓ_1 [7] and FPC-ℓ_2 [5], respectively. Alternatively, the matching sign pursuit (MSP) [8], the binary iterative hard thresholding (BIHT) [9] and the History algorithm [10] have been proposed, achieving high reconstruction performance. FPC-based algorithms do not require knowing the level of sparsity in advance, knowledge that might not be available in a practical setting. However, the performance of FPC-based algorithms is relatively poor compared to other algorithms that need the sparsity level as an input. Thus, a more accurate reconstruction method that does not require knowledge on the signal sparsity level is desired.

Deep neural networks (DNNs) have demonstrated state-of-the-art performance in many inference problems [11], [12], including signal recovery tasks [13]. Nevertheless, DNNs are considered as black-box models and do not incorporate prior knowledge about the signal structure. Deep unfolding [14] is expected to bridge the gap between analytical methods and deep learning-based methods by designing DNNs as unrolled iterations of optimization algorithm. Deep unfolding networks are interpretable (as opposed to traditional DNNs) and have proven to be superior to traditional optimization-based methods and DNN models (because they integrate prior knowledge about the signal structure). Examples of such networks include LISTA [14], ℓ_1-ℓ_1-RNN [15], and LeSITA [16]. In a recent study we presented a DeepFPC network [17] designed by unfolding the iterations of the FPC-ℓ_1 algorithm. DeepFPC achieved good performance in solving the 1-bit CS problem without requiring knowing the signal sparsity level. It was, however, shown that the reconstruction performance of DeepFPC [17] degrades when the measurements are contaminated by high noise. This calls for a robust deep unfolded network showing improved performance in the presence of noise.

The contribution of this paper is as follows: (i) we design a novel DeepFPC-ℓ_2 network by corresponding each layer of an unfolded deep network to an iteration of the FPC-ℓ_2 algorithm; (ii) we propose extended matrices to reduce the time complexity and improve the speed and efficiency of the designed network; (iii) we compare the robustness of the previous DeepFPC [17] and the proposed DeepFPC-ℓ_2 for different signal to noise ratios (SNR) and sign-flipped to signal ratios (flip ratio), and shows that the latter is more robust to noise. This indicates that the robustness of a unfolded network is related to the that of the algorithm it stems from.

In what follows, the next section provides a brief background on the FPC-based method and compares the robustness to noise of the one-sided ℓ_1-norm and the one-sided ℓ_2-norm formulations. Section III introduces the proposed DeepFPC-ℓ_2 model and Section IV reports simulation results. Finally, Section V concludes the paper.
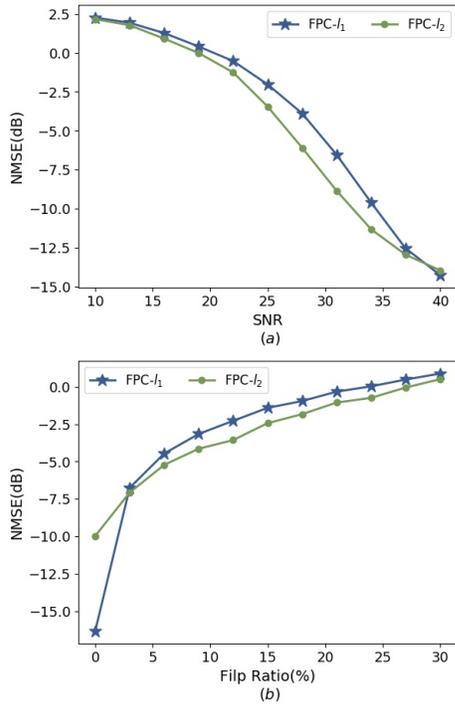
Fig. 1. Reconstruction NMSE for FPC-ℓ_1 and FPC-ℓ_2; the sign flips of measurements are caused by (a) measurement noise; (b) transmission errors.

Notation: Throughout the paper, $\boldsymbol{y} = [y_1, y_2, ..., y_M]^T \in \{-1,1\}^M$ is the vector of measurements, $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ is the sensing matrix, and $\boldsymbol{x} \in \mathbb{R}^N$ is the sparse signal with sparsity $K \ll N$. Moreover, $\tilde{\boldsymbol{X}} = [\mathbf{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_L] \in \mathbb{R}^{N \times L}$ combines L sparse signal vectors into a signal matrix, $\tilde{\boldsymbol{Y}} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_L] \in \mathbb{R}^{M \times L}$ is the measurement matrix combining L measurement vectors and $\boldsymbol{Y}_l = \text{diag}(\boldsymbol{y}_l)$ is a square matrix with $\boldsymbol{y}_l$ being the diagonal entries. Furthermore, $\odot$ and $\otimes$ respectively denote the matrix-multiply product and the Hadamard product, $(\cdot)^T$ is the transpose operation, and $\|\cdot\|_p$ represents the p-norm and $|\cdot|$ is the element-wise absolute operator.

II. BACKGROUND ON FPC-BASED METHODS

In this section, two FPC-based algorithms, i.e., FPC-ℓ_1 and FPC-ℓ_2, are introduced. Moreover, we review their robustness to noise, which forms our motivation to design the DeepFPC-ℓ_2 model.

A. The FPC-ℓ_1 and FPC-ℓ_2 Algorithms

In 1-bit CS, the measurements are acquired as:

$$\boldsymbol{y} = \text{sign}(\boldsymbol{\Phi x}) \quad (1)$$

where the sign$(\cdot)$ function is applied element-wise to the vector $\boldsymbol{\Phi x}$. The FPC-based algorithm for 1-bit CS recovery solves the following problem:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\text{argmin}} \|\boldsymbol{x}\|_1 + \lambda \sum_{i=1}^{M} f((\boldsymbol{Y}\boldsymbol{\Phi x})_i) \text{ s.t. } \|\boldsymbol{x}\|_2 = 1 \quad (2)$$

where λ is a tuning parameter ($\lambda > 0$) and $f(x)$ can be specified as the one-sided ℓ_1-norm:

$$f_{\ell_1}(x) = \begin{cases} 0, & x \geq 0 \\ |x|, & x < 0 \end{cases} \quad (3)$$

or the one-sided ℓ_2-norm penalty:

$$f_{\ell_2}(x) = \begin{cases} 0, & x \geq 0 \\ \frac{x^2}{2}, & x < 0 \end{cases} \quad (4)$$

The convexity and smoothness of this function allows the use of FPC method to perform the minimization. The FPC method solves Problem (2) by updating $\boldsymbol{x}$ iteratively as:

$$\boldsymbol{u} = S_\nu \left(\boldsymbol{x}^{(r)} - \tau \mathbf{g}(\boldsymbol{x}^{(r)}) \right), \quad (5a)$$

$$\boldsymbol{x}^{(r+1)} = \boldsymbol{u}/\|\boldsymbol{u}\|_2, \quad (5b)$$

where $\tau > 0$, $\nu = \tau/\lambda$ and $S_\nu(\cdot) \triangleq \text{sign}(\cdot) \odot \max\{|\cdot| - \nu, 0\}$ is the soft-thresholding operator. Furthermore, $g(\boldsymbol{x})$ is the gradient of the consistent term. When the one-sided ℓ_1-norm is considered, the gradient is given by

$$g_{\ell_1}(\boldsymbol{x}) = \boldsymbol{\Phi}^T(\text{sign}(\boldsymbol{\Phi x}) - \boldsymbol{y}), \quad (6)$$

whereas, when the one-sided ℓ_2-norm is considered, the gradient becomes:

$$g_{\ell_2}(\boldsymbol{x}) = \boldsymbol{\Phi}^T \boldsymbol{Y} \bar{f}'_{\ell_2}(\boldsymbol{Y}\boldsymbol{\Phi x}), \quad (7)$$

where $\boldsymbol{Y} = \text{diag}(\boldsymbol{y})$ and

$$\bar{f}'_{\ell_2}(x) = \begin{cases} 0, & x > 0 \\ x, & x \leq 0 \end{cases} \quad (8)$$

Given proper values for τ, λ and initialization $\boldsymbol{x}^{(0)}$, the sparse signal can be reconstructed by these two FPC-based algorithms.

B. The FPC-ℓ_1 and FPC-ℓ_2 Under Noisy Measurements

Due to noise or transmission errors, the binary measurements could suffer from sign flips, which will degrade the reconstruction performance of 1-bit CS. To address this issue, a robust recovery method is required. The authors of [9] showed that the BIHT-ℓ_2 algorithm outperforms the BIHT-ℓ_1 algorithm when the measurements contain sign flips. This means that the one-sided ℓ_2 objective has better robustness against noise than the one-sided ℓ_1. We therefore conjecture that the FPC-ℓ_2 algorithm has better noise robustness than the FPC-ℓ_1 algorithm. To test this hypothesis, we compare their performance for two kinds of sign flips (in the measurement process and in the transmission process) and show the results in Fig.1. It can be seen that FPC-ℓ_2 has indeed better recovery performance than FPC-ℓ_1. Thus, we propose to design an DeepFPC-ℓ_2 network by unfolding the FPC-ℓ_2 algorithm to improve the recovery performance under noise.

III. THE PROPOSED DEEPFPC-ℓ_2 MODEL

In this section, we present our DeepFPC-ℓ_2 architecture (Section III-A) and describe how to build extended matrices for parallel computing (Section III-B).
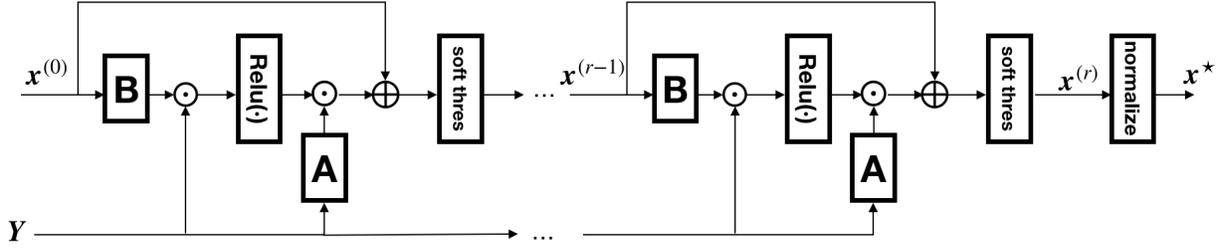
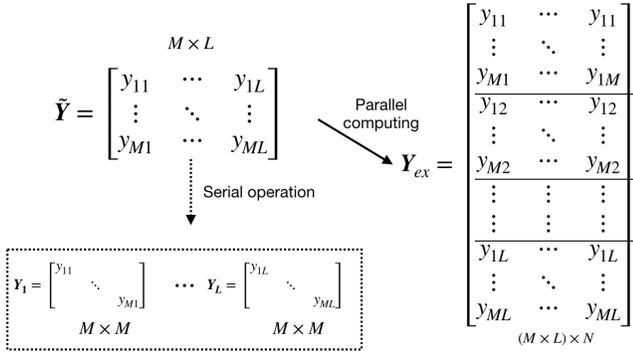Fig. 2. The block diagram of proposed DeepFPC-ℓ_2 network.

Fig. 3. Comparison of the serial and parallel operation.

A. The Proposed Deep Unfolded Network Architecture

Substituting $g_{\ell_2}(\boldsymbol{x})$ in (7) into the iteration (5a), the update becomes:

$$\boldsymbol{u} = S_\nu(\boldsymbol{x}^{(r)} - \boldsymbol{A}\boldsymbol{Y} \bar{f}^{\,l}_{\ell_2}(\boldsymbol{Y}\boldsymbol{B}\boldsymbol{x})) \qquad (9)$$

where $\boldsymbol{A} = \tau\boldsymbol{\Phi}^{\mathrm{T}}$ and $\boldsymbol{B} = \boldsymbol{\Phi}$. Following the principles in [14], the update can be presented in the form of neural-network layers. An initialization $\boldsymbol{x}^{(0)}$ and a diagonal matrix containing the binary measurement vector $\boldsymbol{Y}$ are as inputs to the network. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are trainable linear weights, which may change per layer. Furthermore, $S_\nu(\cdot)$ and $\bar{f}^{\,l}_{\ell_2}(\cdot)$ play the role of non-linear activation functions in the network.

Unlike the traditional FPC-ℓ_2 algorithm, the DeepFPC-ℓ_2 network does not need to apply a normalization step after every layer, because the network is trained with signal examples and it will not learn to generate zero solutions of finite layers. Hence, normalization is performed after the final layer. Moreover, $\bar{f}^{\,l}_{\ell_2}(\cdot)$ in (8) can be expressed as the relu$(\cdot)$ function [18], which leads (9) to:

$$\boldsymbol{x}^{(r+1)} = S_\nu(\boldsymbol{x}^{(r)} + \boldsymbol{A}\boldsymbol{Y}\mathrm{relu}(\boldsymbol{Y}\bar{\boldsymbol{B}}\boldsymbol{x}^{(r)})) \qquad (10)$$

where $\bar{\boldsymbol{B}} = -\boldsymbol{B} = -\boldsymbol{\Phi}$. In the last layer, the final estimate is normalized as $\boldsymbol{x}^\star = \boldsymbol{x}^{(R)}/\|\boldsymbol{x}^{(R)}\|_2$.

The block diagram of the proposed DeepFPC-ℓ_2 is shown in Fig.2. Specifically, the input to the soft-thresholding non-linearity is fed through a shortcut connection from the output of the previous layer. This implies that DeepFPC-ℓ_2 can undergo deep training.

B. Built with Extended Matrices

As illustrated in (10) and Fig. 2, the input $\boldsymbol{Y}$ of the DeepFPC-ℓ_2 network is the diagonal matrix of one measurement vector $\boldsymbol{y}$. For the measurement matrix $\tilde{\boldsymbol{Y}}$, each column, i.e., one measurement vector $\boldsymbol{y}_l$, has to be diagonalized firstly as $\boldsymbol{Y}_l$ (shown in Fig.3, serial operation). Consequently, $\boldsymbol{x}_l$ is recovered from each input $\boldsymbol{Y}_l$ one by one, and the operators ($\boldsymbol{A}\boldsymbol{Y}_l$ and $\boldsymbol{Y}_l\bar{\boldsymbol{B}}\boldsymbol{x}$) have to be computed multiple times, which is quite time-consuming. Hence, we aim to use parallel computing to reduce time complexity and improve the speed and efficiency of the proposed network.

Our strategy is to expand the matrix to perform parallel computing. Firstly, $\boldsymbol{Y}_{ex}$ is designed by extending the measurement matrix $\tilde{\boldsymbol{Y}}$ (shown in Fig. 3 as *parallel computing*) and then used it as the network input. At the same time, Hadamard product operations are used between extended matrices, instead of the matrix-multiply product. In this way, we can still get the same result with paralleled sum and reshape operations. Specifically, considering that $\boldsymbol{Y}_l$ is a diagonal matrix, we can write:

$$\begin{aligned}
\boldsymbol{Y}_l\boldsymbol{B} &= [y_{1l}\boldsymbol{b}_1^T, y_{2l}\boldsymbol{b}_2^T, \ldots, y_{Ml}\boldsymbol{b}_M^T]^T \\
&= [\hat{\boldsymbol{y}}_{l1}^T \otimes \boldsymbol{b}_1^T, \hat{\boldsymbol{y}}_{l2}^T \otimes \boldsymbol{b}_2^T, \ldots, \hat{\boldsymbol{y}}_{lM}^T \otimes \boldsymbol{b}_M^T]^T \\
&= \hat{\boldsymbol{Y}}_l \otimes \boldsymbol{B},
\end{aligned} \qquad (11)$$

where $\boldsymbol{b}_m$ is the m-th row vector of $\boldsymbol{B}$, $\hat{\boldsymbol{y}}_{ml}$ is an N-dimensional row vector with all entries equal to y_{ml}, and $\hat{\boldsymbol{Y}}_l = [\hat{\boldsymbol{y}}_{1l}^T, \hat{\boldsymbol{y}}_{1l}^T, \ldots, \hat{\boldsymbol{y}}_{1l}^T]^T$. This implies that the matrix-multiply product is replaced by the Hadamard product. We define the following extended matrices as $\boldsymbol{Y}_{ex} = [\hat{\boldsymbol{Y}}_1^T, \hat{\boldsymbol{Y}}_2^T, \ldots, \hat{\boldsymbol{Y}}_N^T]^T \in \mathbb{R}^{(M \times L) \times N}$ and $\boldsymbol{B}_{ex} = [\boldsymbol{B}^T, \boldsymbol{B}^T, \ldots, \boldsymbol{B}^T]^T \in \mathbb{R}^{(M \times L) \times N}$, i.e., we reshape $\tilde{\boldsymbol{Y}}$ into a column vector and copy it N times horizontally to build the extended matrix $\boldsymbol{Y}_{ex}$. Moreover, we copy $\boldsymbol{B}$ L times vertically to build the extended matrix $\boldsymbol{B}_{ex}$. Then, we can compute all $\boldsymbol{Y}_l\boldsymbol{B}$ at the same time by a Hadamard product $\boldsymbol{Y}_{ex} \otimes \boldsymbol{B}_{ex}$, as shown in (12).

Similarly, $\boldsymbol{X}_{ex} \in \mathbb{R}^{N \times (M \times L)}$ is generated by copying each sample of $\boldsymbol{x}_l$ horizontally M times [see (13)], and $\boldsymbol{A}_{ex} \in \mathbb{R}^{N \times (M \times L)}$ is obtained by copying $\boldsymbol{A}$ horizontally L times (i.e., $\boldsymbol{B}_{ex}^T$). Then, the results of the parallel computing are calculated through the sum and reshape operations.

$$\boldsymbol{Y}_{ex} = \begin{bmatrix} y_{11} & \cdots & y_{11} \\ \vdots & \ddots & \vdots \\ y_{M1} & \cdots & y_{M1} \\ y_{12} & \cdots & y_{12} \\ \vdots & \ddots & \vdots \\ y_{M2} & \cdots & y_{M2} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ y_{1L} & \cdots & y_{1L} \\ \vdots & \ddots & \vdots \\ y_{ML} & \cdots & y_{ML} \end{bmatrix}, \boldsymbol{B}_{ex} = \begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1N} \\ \vdots & \ddots & \vdots \\ \Phi_{M1} & \cdots & \Phi_{MN} \\ \Phi_{11} & \cdots & \Phi_{1N} \\ \vdots & \ddots & \vdots \\ \Phi_{M1} & \cdots & \Phi_{MN} \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \Phi_{11} & \cdots & \Phi_{1N} \\ \vdots & \ddots & \vdots \\ \Phi_{M1} & \cdots & \Phi_{MN} \end{bmatrix}$$

$$\boldsymbol{Y}_{ex} \otimes \boldsymbol{B}_{ex} = \begin{bmatrix} \boldsymbol{Y}_1 \boldsymbol{B}; \boldsymbol{Y}_2 \boldsymbol{B}; \cdots; \cdots; \boldsymbol{Y}_L \boldsymbol{B} \end{bmatrix} \quad (12)$$

$$\boldsymbol{X}_{ex} = \begin{bmatrix} x_{11} & \cdots & x_{11} & \cdots & x_{1L} & \cdots & x_{1L} \\ \vdots & \cdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{N1} & \cdots & x_{N1} & \cdots & x_{NL} & \cdots & x_{NL} \end{bmatrix} \quad (13)$$

In this way, the time complexity of each layer for DeepFPC-ℓ_2 is reduced from $O(n^3)$ to $O(n^2)$, and the extended matrix effectively improves the network construction speed.

IV. EXPERIMENTAL RESULTS

A. Sparse Signal Reconstruction

The performance of the DeepFPC-ℓ_2 model is assessed in this section. The model is implemented using TensorFlow in Python and trained with the ADAM optimizer using an exponentially decaying step-size. A set of $\{\boldsymbol{y}^l, \boldsymbol{x}^l\}$ ($l = 1, 2, ..., L$) pairs (L = 100) is used for training and a different set of 100 pairs is used for testing. We set the signal dimension to $N = 100$, and its sparsity level to $K = 10$. Moreover, we choose the support locations from the uniform distribution. Their values are drawn $i.i.d$ from the standard normal and the measurement matrix $\boldsymbol{\Phi}$ is randomly drawn from the Gaussian distribution $\mathcal{N}(0, 1/M)$. The measurement vectors' dimension is $M = 300$ and are generated by computing (1) with the signal and $\boldsymbol{\Phi}$. The normalized mean square error (NMSE), defined as NMSE(dB) $\triangleq 10 \log_{10} \left(\|\boldsymbol{x}^\star - \boldsymbol{x}\|_2^2 / \|\boldsymbol{x}\|_2^2 \right)$, is used as the metric, with $\boldsymbol{x}^\star$ being the reconstructed vector.

Table I shows the performance comparison between the FPC-ℓ_2 algorithm and the DeepFPC-ℓ_2 network in terms of the averaged NMSE for the first 20 iterations/layers. It is clear that the DeepFPC-ℓ_2 model significantly improves the corresponding algorithm. The FPC-ℓ_2 algorithm will converge to -14.39dB after 150 iterations. However, after 8 layers, DeepFPC-ℓ_2 has achieved this performance and it continues to reduce the NMSE as the number of layers increases. In this 20-layer network, the best result is better than the original algorithm with about 2dB margin.

TABLE I
PERFORMANCE COMPARISON BETWEEN THE FPC-ℓ_2 ALGORITHM AND
THE PROPOSED DEEPFPC-ℓ_2 NETWORK.

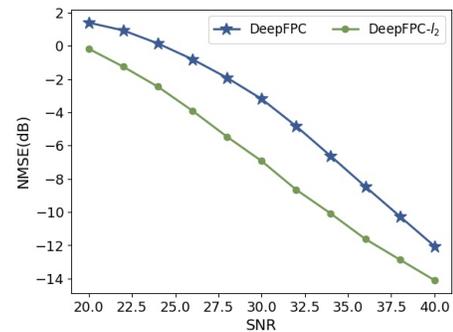| Layers/ Iterations	FPC-l_2 [dB]	DeepFPC-l_2 [dB]	Layers/ Iterations	FPC-l_2 [dB]	DeepFPC-l_2 [dB]
1	-4.53	-12.27	11	-6.34	-14.64
2	-4.80	-12.32	12	-6.48	-15.07
3	-5.03	-12.67	13	-6.62	-15.16
4	-5.23	-12.71	14	-6.76	-15.34
5	-5.42	-13.03	15	-6.89	-15.51
6	-5.59	-13.87	16	-7.02	-15.84
7	-5.76	-14.00	17	-7.16	-15.90
8	-5.91	-14.41	18	-7.29	-15.99
9	-6.06	-14.53	19	-7.42	-16.00
10	-6.20	-14.64	20	-7.55	-16.16

Fig. 4. Comparison of the robustness to noise for DeepFPC [17] and the proposed DeepFPC-ℓ_2 for SNR = 20dB to 40dB

B. Reconstruction in the Noisy Scenario

In this section, we compare DeepFPC-ℓ_2 with DeepFPC [17] in terms of their performance in the presence of noise in the measurements. The same settings as Section IV-A are designed for training the DeepFPC [17] model. Two kinds of noise, i.e., noise resulting from the measurement and the transmission process, are considered.

1) Measurements contaminated by Gaussian noise: Unlike the training set, the measurements in the testing set are contaminated by Gaussian noise. The signal-to-noise ratio (SNR) in dB is defined as:

$$SNR = 10 \log_{10} \left(\frac{P_S}{P_N} \right), \quad (14)$$

where P_S and P_N are the power of the signal and noise, respectively. The amplitude of noise A_N can be calculated as:

$$A_N = \sqrt{P_S / 10^{(\frac{SNR}{10})}} \quad (15)$$

The Gaussian noise follows the distribution $\mathcal{N}(0, A_N^2)$ and is added to the measurement matrix; different SNR levels are considered. DeepFPC and DeepFPC-ℓ_2 are trained with a noiseless sampling set, and then they are tested with the noisy measurements. The results are shown in Fig.4. It is clear that the recovery performance of the proposed DeepFPC-ℓ_2 network is better than that of our previous DeepFPC model, which means that DeepFPC-ℓ_2 has better robustness to noise.

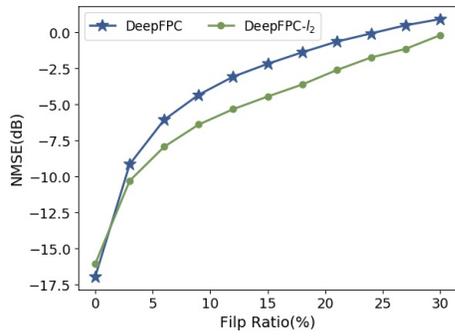Furthermore, we also tried to recover signals under low SNR conditions (less than 20dB). Although the NMSE of

Fig. 5. Comparison of the robustness to noise for DeepFPC and DeepFPC-ℓ_2 on ratio = 0% to 30%

DeepFPC-ℓ_2 is still less than that of the DeepFPC, neither of them recover the signals with high accuracy. Improving the accuracy of DeepFPC-based model in the low SNR regime is still an open problem, which we plan to address in subsequent work.

2) Measurements contaminated by transmission error: We now consider sign flips, which are caused by bit errors in the transmission process. In order to model sign flips, the measurement vector y is multiplied by a vector $\xi \in \{-1, 1\}^M$ to control flips. When an entry in the measurement vector is multiplies by -1, the sign of the entry flips. The locations of -1 are chosen randomly from 1 to 300 ($M = 300$) and the number of -1 is determined by the flip ratio (see Fig.5).

In Fig.5 we see that DeepFPC-ℓ_2 has a better signal reconstruction performance than the previous DeepFPC network when the flip ratio is equal to 3% or higher; this shows that the DeepFPC-ℓ_2 has better robustness to noise than the DeepFPC. However, we can also notice that the recovery performance of the DeepFPC is better than the DeepFPC-ℓ_2 when the flip ratio is small enough. This indicates that DeepFPC outperforms DeepFPC-ℓ_2 in the noiseless scenario, and the reason could be that the associated one-sided ℓ_1-norm algorithm has superior performance in the noiseless case compared to the one-sided ℓ_2-norm [9] algorithm.

To sum up, the proposed DeepFPC-ℓ_2 network outperforms the traditional FPC-ℓ_2 algorithm for sparse signal recovery. Moreover, compared with the previous DeepFPC [17] model, it has better robustness when the measurements are contaminated by noise, for a wide range of SNRs.

V. Conclusion

DeepFPC-ℓ_2, a deep neural network designed by unfolding the FPC-ℓ_2 recovery algorithm for 1-bit CS, is proposed in this paper. Our result demonstrates that the DeepFPC-ℓ_2 not only improves the recovery performance of the original algorithm, but also greatly improves the convergence speed. Moreover, we explore the robust property of the DeepFPC-ℓ_2 network, and we find that the DeepFPC-ℓ_2 network has better noise immunity than the previous DeepFPC model. The results also indicate that the robust property of a deep unfolding network relies on the algorithm it stems from. When the measurements are polluted by noise, DeepFPC-ℓ_2 is a better option to obtain more accurate results. This approach can be applied to many 1-bit CS applications which can be affected by noise, such as DOA estimation, image processing and wireless communications.

References

[1] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] J. Romberg, "Imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 14–20, 2008.

[3] W. Bajwa, J. Haupt, A. Sayeed, and R. Nowak, "Compressive wireless sensing," in *Proceedings of the 5th international conference on Information processing in sensor networks*, 2006, pp. 134–142.

[4] D. Malioutov, M. Cetin, and A. S. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE transactions on signal processing*, vol. 53, no. 8, pp. 3010–3022, 2005.

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

[6] E. T. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing," *CAAM TR07-07, Rice University*, vol. 43, p. 44, 2007.

[7] P. Xiao, B. Liao, X. Huang, and Z. Quan, "1-bit compressive sensing with an improved algorithm based on fixed-point continuation," *Signal Processing*, vol. 154, pp. 168–173, 2019.

[8] P. T. Boufounos, "Greedy sparse signal reconstruction from sign measurements," in *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*. IEEE, 2009, pp. 1305–1309.

[9] 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, 2013.

[10] B. Sun, H. Feng, and X. Xu, "History: An efficient and robust algorithm for noisy 1-bit compressed sensing," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 10, pp. 2566–2573, 2016.

[11] J. Mansanet, A. Albiol, and R. Paredes, "Local deep neural networks for gender recognition," *Pattern Recognition Letters*, vol. 70, pp. 80–86, 2016.

[12] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable bayesian optimization using deep neural networks," in *International conference on machine learning*, 2015, pp. 2171–2180.

[13] D. M. Nguyen, E. Tsiligianni, and N. Deligiannis, "Deep learning sparse ternary projections for compressed sensing of images," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 1125–1129.

[14] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010, pp. 399–406.

[15] H. D. Le, H. Van Luong, and N. Deligiannis, "Designing recurrent neural networks by unfolding an l1-l1 minimization algorithm," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 2329–2333.

[16] E. Tsiligianni and N. Deligiannis, "Deep coupled-representation learning for sparse linear inverse problems with side information," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1768–1772, 2019.

[17] P. Xiao, B. Liao, and N. Deligiannis, "Deepfpc: A deep unfolded network for sparse signal recovery from 1-bit measurements with application to doa estimation," *Signal Processing*, 2020, doi: https://doi.org/10.1016/j.sigpro.2020.107699.

[18] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.