





Phase Retrieval With Learning Unfolded Expectation Consistent Signal Recovery Algorithm

Chang-Jen Wang , Chao-Kai Wen , *Member, IEEE*, Shang-Ho Tsai , *Senior Member, IEEE*,
and Shi Jin , *Senior Member, IEEE*

Abstract—Phase retrieval algorithms are now an important component of many modern computational imaging systems. A recently proposed scheme called generalized expectation consistent signal recovery (GEC-SR) shows better accuracy, speed, and robustness than numerous existing methods. Decentralized GEC-SR (deGEC-SR) addresses the scalability issue in high-resolution images. However, the convergence speed and stability of these algorithms heavily rely on the settings of several handcrafted tuning factors with inefficient turning process. In this work, we propose deGEC-SR-Net by unfolding the iterative deGEC-SR algorithm into a learning network architecture with trainable parameters. The parameters of deGEC-SR-Net are determined by data-driven training. Numerical results show that deGEC-SR-Net provides substantially faster convergence than deGEC-SR and exhibits superior robustness to noise and prior mis-specifications.

Index Terms—Phase retrieval, deep neural network, unfolding, decentralized algorithm.

I. INTRODUCTION

THE reconstruction of a complex vector from its linear transform magnitude involves numerous imaging applications, such as ptychography [1], optical crystallography [2], [3], and inverse scattering [4]. From these applications, the phase retrieval (PR) problem arises. To solve this problem, numerous algorithms are developed, such as Gerchberg-Saxton [5], Fienup [6], PhaseLift [7], Wirtinger Flow (WF) [8], PhaseMax [9], PhaseLamp [10], re-weighted WF [11], prGAMP [12], prSAMP [13], prVAMP [14], and generalized expectation consistent signal recovery (GEC-SR) [15].

Manuscript received February 29, 2020; revised April 13, 2020; accepted April 20, 2020. Date of publication April 27, 2020; date of current version June 3, 2020. The work of Chang-Jen Wang and Shang-Ho Tsai was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 108-2218-E009-027. The work of Chao-Kai Wen was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST 108-2218-E-110-014- and Grant MOST 108-2628-E-110-001-MY3. The work of Shi Jin was supported in part by the National Science Foundation of China under Grant 61941104. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Feifei Gao. (*Corresponding author: Chao-Kai Wen.*)

Chang-Jen Wang is with the Institute of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: dkman0988@gmail.com).

Chao-Kai Wen is with the Institute of Communications Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan (e-mail: chaokai.wen@mail.nsysu.edu.tw).

Shang-Ho Tsai is with the Department of Electrical Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: shanghot@mail.nctu.edu.tw).

Shi Jin is with the National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China (e-mail: jinshi@seu.edu.cn).

Digital Object Identifier 10.1109/LSP.2020.2990767

More than 16 PR algorithms are compared in [14], [15]. Among the algorithms, GEC-SR, prVAMP, and prSAMP demonstrate the best performance in various transformation matrices and signal-to-noise ratios (SNRs). However, the speed of prSAMP is extremely slow because it makes parallel updates sequential. The accuracy of GEC-SR and prVAMP is comparable to that of prSAMP, but GEC-SR and prVAMP run fast. GEC-SR slightly outperforms prVAMP and is the best by far due to effective updating. Decentralized GEC-SR (deGEC-SR) [15] has been proposed recently to address scalability issues in high-resolution images.

GEC-SR and deGEC-SR are iterative message-passing algorithms. In certain cases, such algorithms have several unexpected numerical issues and divergences. The damping method [16] is used generally to improve robustness of these algorithms. A conservative damping mechanism with slow update rate can prevent instabilities while slowing the algorithm to obtain a stationary solution [12]–[14]. In [15], incrementally increasing the update rate with the number of iterations is suggested to accelerate convergence speed. The damping factor determines the speed of the algorithm and controls instabilities. However, the tuning process of the parameters is customized and demonstrates low efficiency.

Considering these developments, we use algorithm unfolding technique [17]–[20] to propose a trainable deGEC-SR algorithm called deGEC-SR-Net. It is a learning neural network (NN) whose architecture is based on an unfolded deGEC-SR algorithm. All modules are computed as a solution to the data-free Bayesian estimation problem, thereby leaving only the damping factors for data-driven learning. This architecture considerably simplifies/speeds up training because only a few parameters must be optimized. The simulation results reveal that deGEC-SR-Net can obtain the same accuracy as the original deGEC-SR using fewer iterations. deGEC-SR-Net exhibits strong robustness under varying input distributions or noise levels.

II. PROBLEM SETUP

The PR problem generally aims to recover the unknown signal $\mathbf{x} \in \mathbb{C}^N$ by observing $\mathbf{y} \in \mathbb{R}_+^M$ through linear transform matrix $\mathbf{A} \in \mathbb{C}^{M \times N}$ with noise interference \mathbf{w} , as expressed by

$$\mathbf{y} = \mathbf{Q}(\mathbf{A}\mathbf{x} + \mathbf{w}), \quad (1)$$

where operator $\mathbf{Q}(\cdot)$ takes the element-wise $|\cdot|$. We consider the case where signal $\mathbf{x} \in \mathbb{C}^N$ is generated following a prior distribution $p(\mathbf{x})$, and noise $\mathbf{w} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbf{1})$. We assume that transform matrix \mathbf{A} , prior distribution $p(\mathbf{x})$, and noise level

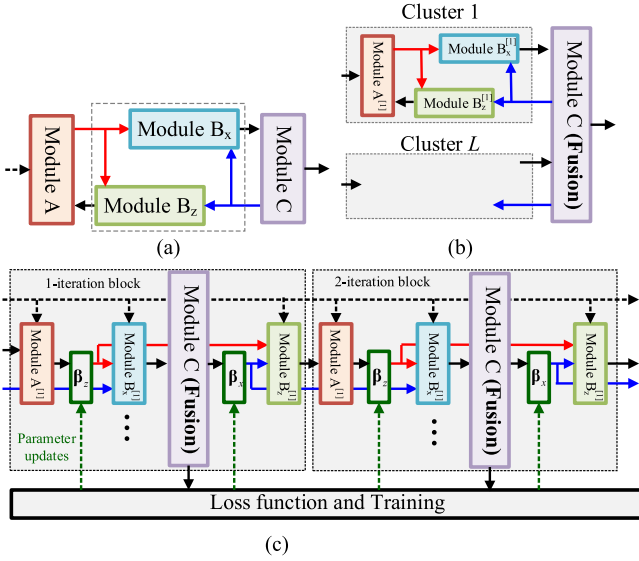


Fig. 1. Block diagrams of (a) GEC-SR, (b) deGEC-SR, and (c) deGEC-SR-Net.

σ_w^2 are known. Our goal is to develop an efficient algorithm for recovering signal \mathbf{x} from \mathbf{y} .

III. ALGORITHMS VIA DEEP UNFOLDING

A. GEC-SR and deGEC-SR

To demonstrate learning unfolding, we briefly review GEC-SR and deGEC-SR [15] in this subsection. Although GEC-SR is derived on basis of a sophisticated expectation consistent approximation [21], we present it in a modular-based approach [22], [23] for ease of developing unfolding. We first define a hidden vector $\mathbf{z} = \mathbf{A}\mathbf{x}$. As such, measurements \mathbf{y} are related to \mathbf{x} following mapping procedure $\mathbf{x} \rightarrow \mathbf{z} \rightarrow \mathbf{y}$. The inference of \mathbf{x} will then follow a reverse procedure, that is, $\mathbf{y} \rightarrow \mathbf{z} \rightarrow \mathbf{x}$, which is achieved through three modules, namely, Modules A, B, and C (Fig. 1).

- Module A computes the unbiased estimates (i.e., mean and variance) of \mathbf{z} . This process is interpreted as *de-nonlinear* because \mathbf{z} is estimated through the de-nonlinear process from nonlinear measurements \mathbf{y} .
- Module B provides the unbiased estimates of (\mathbf{z}, \mathbf{x}) based on the linear transformation of $\mathbf{z} = \mathbf{A}\mathbf{x}$. The inputs of Module B are (\mathbf{z}, \mathbf{x}) , wherein the priors of \mathbf{z} and \mathbf{x} are from Modules A and C, respectively. The output of Module B is either \mathbf{x} or \mathbf{z} , depending on its processing direction. We use Modules B_x and B_z to indicate the outputs of Module B being \mathbf{x} and \mathbf{z} , respectively. In Module B, (\mathbf{z}, \mathbf{x}) are assumed to be Gaussian distributed to facilitate the estimates, and their true prior distributions are not used.
- Given the estimated \mathbf{x} from Module B, Module C further computes the unbiased estimates of \mathbf{x} on the basis of its true prior distribution $p(\mathbf{x})$. This process is interpreted as *de-noising*.

Using the aforementioned procedure (feed-forward direction), we obtain the first inference of \mathbf{x} . We then push \mathbf{x} back to \mathbf{y} in a backward direction. The three modules

are executed iteratively in a circular manner $\mathbf{A} \rightarrow \mathbf{B}_x \rightarrow \mathbf{C} \rightarrow \mathbf{B}_z \rightarrow \mathbf{A} \rightarrow \mathbf{B}_x \rightarrow \mathbf{C} \rightarrow \mathbf{B}_z \rightarrow \dots$ until convergence.

To address the scalability issue, [15] proposed to divide measurements \mathbf{y} into L independent clusters with small pieces of $M_l = M/L$. The measurements (1) in cluster l can be written as

$$\mathbf{y}_l = \mathbf{Q}(\mathbf{A}_l \mathbf{x} + \mathbf{w}_l), \quad (2)$$

where $\mathbf{y}_l, \mathbf{w}_l \in \mathbb{C}^{M_l}$ denote the l th subset of \mathbf{y} and \mathbf{w} , respectively, and transform matrix $\mathbf{A} \in \mathbb{C}^{M \times N}$ is split into L sub-matrices $\mathbf{A}_l \in \mathbb{C}^{M_l \times N}$. The subset of measurements \mathbf{z} is denoted by $\mathbf{z}_l = \mathbf{A}_l \mathbf{x}$. Given this data partition, deGEC-SR performs local inference using the original GEC-SR for each cluster separately, and combines the results to obtain a global estimate.

A block diagram of deGEC-SR is illustrated in Fig. 1(b). We use subscript $^{[l]}$ to indicate the estimated parameters in the l -th cluster. As the measurements \mathbf{y}_l 's go through Modules $A^{[l]}$ and $B_x^{[l]}$, we obtain the unbiased estimates $\mathbf{x}^{[l]}$. After feeding the estimates $\mathbf{x}^{[l]}$ for $l = 1, \dots, L$ into Module C, we add constraint $\mathbf{x}^{[1]} = \mathbf{x}^{[2]} = \dots = \mathbf{x}^{[L]}$ to establish the global estimates of \mathbf{x} . Similar to that in the original GEC-SR, the unbiased estimates of \mathbf{x} is sent to Modules $B_z^{[l]}$ and $A^{[l]}$ for each cluster, and the procedure is repeated.

B. deGEC-SR-Net

We develop a learning network on the basis of deGEC-SR. Each iteration of deGEC-SR comprises Modules $A^{[l]}$, $B_x^{[l]}$, C, and $B_z^{[l]}$. As shown in Fig. 1(c), the modules in one iteration step are unfolded into a single network layer. We subsequently form a deep network by mapping each iteration to a network layer and stacking the layers together. This process is equivalent to the execution of a deGEC-SR iteration multiple times. We fix the total number of iterations T (i.e., T layers) and use (t) to represent the estimated parameters in the t -th layer of the network.

Each module can be considered a black box NN and trained end-to-end using real datasets. However, we continue module performance of unbiased estimation and introduce only two trainable damping operations after Modules $A^{[l]}$ and C. This strategy is based on the argument that the modules (or estimators) are well-developed mathematically with few uncertainties. Making the modules trainable prevents remarkable gains while increasing training complexity. We leverage deep learning to adjust the update to the right amount at each iteration because inadequate and conservative updates result in divergence and unnecessarily slows convergence, respectively.

Each module in deGEC-SR-Net (i.e., Modules $A^{[l]}$, $B_x^{[l]}$, C, and $B_z^{[l]}$) comprises the Bayesian posterior estimator, followed by an extrinsic (or debias) operation. To describe these operations in a generic form, we let “ a ” and “ b ” be the random variables of either \mathbf{x} or \mathbf{z} . Of note, \mathbf{a} and \mathbf{b} can refer to the same random variable. Given the mean and variance $(\boldsymbol{\mu}_{1a}, \mathbf{v}_{1a})$ of the previous module, the posterior mean and variance $(\hat{\boldsymbol{\mu}}_{1b}, \hat{\mathbf{v}}_{1b})$ are computed by taking the expectations of \mathbf{b} w.r.t.

$$f(\mathbf{b}|\mathbf{a}) \cdot \mathcal{N}_{\mathbb{C}}(\mathbf{a}; \boldsymbol{\mu}_{1a}, \mathbf{v}_{1a}) / \mathcal{Z}, \quad (3)$$

where $\mathcal{Z} = \int f(\mathbf{b}|\mathbf{a})\mathcal{N}_C(\mathbf{a}; \boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}})d\mathbf{a}$ is the normalization. In (3), $f(\mathbf{b}|\mathbf{a})$ is the likelihood function of \mathbf{b} condition on \mathbf{a} . For ease of notation, we express the posterior estimates in pairs as

$$(\widehat{\boldsymbol{\mu}}_{1\mathbf{b}}, \widehat{\mathbf{v}}_{1\mathbf{b}}) = \mathbb{E}\{\mathbf{b}|\boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}}; f(\mathbf{b}|\mathbf{a})\}. \quad (4)$$

For efficient message passing, each module passes only unbiased estimates (or extrinsic messages) to the next module rather than posterior estimates. We use subscripts $(\cdot)_1$ and $(\cdot)_2$ to specify input priors and extrinsic messages, respectively. If the input priors are denoted by $(\cdot)_1$, then their corresponding extrinsic messages are denoted by $(\cdot)_2$, and vice versa. The extrinsic mean and variance of (4) are calculated by excluding the prior mean and variance $(\boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}})$, which are given as

$$\boldsymbol{\mu}_{2\mathbf{a}} = \mathbf{v}_{2\mathbf{a}} \left(\frac{\widehat{\boldsymbol{\mu}}_{1\mathbf{a}}}{\widehat{\mathbf{v}}_{1\mathbf{a}}} - \frac{\boldsymbol{\mu}_{1\mathbf{a}}}{\mathbf{v}_{1\mathbf{a}}} \right), \quad \mathbf{v}_{2\mathbf{a}} = \left(\frac{\mathbf{1}}{\widehat{\mathbf{v}}_{1\mathbf{a}}} - \frac{\mathbf{1}}{\mathbf{v}_{1\mathbf{a}}} \right)^{-1}. \quad (5)$$

We simply denote the extrinsic operation as

$$(\boldsymbol{\mu}_{2\mathbf{a}}, \mathbf{v}_{2\mathbf{a}}) \Leftarrow (\widehat{\boldsymbol{\mu}}_{1\mathbf{a}}, \widehat{\mathbf{v}}_{1\mathbf{a}}) \setminus (\boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}}). \quad (6)$$

Using (4) and (6), Modules $A^{[l]}$, $B_x^{[l]}$, C , and $B_z^{[l]}$ perform the following operations

$$\begin{aligned} \text{Module } A^{[l]} : (\widehat{\boldsymbol{\mu}}_{1z}^{[l]}, \widehat{\mathbf{v}}_{1z}^{[l]}) &= \mathbb{E}\left\{\mathbf{z}^{[l]} \middle| \boldsymbol{\mu}_{1z}^{[l]}, \mathbf{v}_{1z}^{[l]}; f_A\right\}, \\ (\boldsymbol{\mu}_{2z}^{[l]}, \mathbf{v}_{2z}^{[l]}) &\Leftarrow (\widehat{\boldsymbol{\mu}}_{1z}^{[l]}, \widehat{\mathbf{v}}_{1z}^{[l]}) \setminus (\boldsymbol{\mu}_{1z}^{[l]}, \mathbf{v}_{1z}^{[l]}), \end{aligned} \quad (7a)$$

$$\begin{aligned} \text{Module } B_x^{[l]} : (\widehat{\boldsymbol{\mu}}_{2x}^{[l]}, \widehat{\mathbf{v}}_{2x}^{[l]}) &= \mathbb{E}\left\{\mathbf{x}^{[l]} \middle| \boldsymbol{\mu}_{2z}^{[l]}, \mathbf{v}_{2z}^{[l]}; f_B\right\}, \\ (\boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]}) &\Leftarrow (\widehat{\boldsymbol{\mu}}_{2x}^{[l]}, \widehat{\mathbf{v}}_{2x}^{[l]}) \setminus (\boldsymbol{\mu}_{2z}^{[l]}, \mathbf{v}_{2z}^{[l]}), \end{aligned} \quad (7b)$$

$$\begin{aligned} \text{Module } C : (\widehat{\boldsymbol{\mu}}_{1x}, \widehat{\mathbf{v}}_{1x}) &= \mathbb{E}\left\{\mathbf{x} \middle| \boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]}, \forall l; f_C\right\}, \\ (\boldsymbol{\mu}_{2x}^{[l]}, \mathbf{v}_{2x}^{[l]}) &\Leftarrow (\widehat{\boldsymbol{\mu}}_{1x}, \widehat{\mathbf{v}}_{1x}) \setminus (\boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]}), \end{aligned} \quad (7c)$$

$$\begin{aligned} \text{Module } B_z^{[l]} : (\widehat{\boldsymbol{\mu}}_{2z}^{[l]}, \widehat{\mathbf{v}}_{2z}^{[l]}) &= \mathbb{E}\left\{\mathbf{z}^{[l]} \middle| \boldsymbol{\mu}_{2x}^{[l]}, \mathbf{v}_{2x}^{[l]}; f_B\right\}, \\ (\boldsymbol{\mu}_{1z}^{[l]}, \mathbf{v}_{1z}^{[l]}) &\Leftarrow (\widehat{\boldsymbol{\mu}}_{2z}^{[l]}, \widehat{\mathbf{v}}_{2z}^{[l]}) \setminus (\boldsymbol{\mu}_{2x}^{[l]}, \mathbf{v}_{2x}^{[l]}), \end{aligned} \quad (7d)$$

where $f_{A/B/C}$ are the likelihood functions of the concerned estimates given their inputs. To better understand how the operations work, we take Module $A^{[l]}$ as an example. We remove subscript $^{[l]}$ in the following description for ease of notation. In Module A, the likelihood function of each element of \mathbf{y} is given as follows [12, (8)]:

$$f_A := f(y|z) = \frac{2y}{\sigma_w^2} e^{-\frac{y^2 + |z|^2}{\sigma_w^2}} I_0\left(\frac{2y|z|}{\sigma_w^2}\right), \quad (8)$$

where $I_0(\cdot)$ is the 0th-order modified Bessel function of the first kind. By substituting $f(\mathbf{b}|\mathbf{a}) := f(y|z)$ and $(\boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}}) := (\boldsymbol{\mu}_{1z}, \mathbf{v}_{1z})$ into (3), we can obtain the posterior mean and variance of \mathbf{z} as the outputs of Module A, that is, $(\widehat{\boldsymbol{\mu}}_{1z}, \widehat{\mathbf{v}}_{1z})$. The estimates can be expressed explicitly [15, Table I]. Subsequently, we obtain the extrinsic messages $(\boldsymbol{\mu}_{2z}, \mathbf{v}_{2z})$ by substituting $(\widehat{\boldsymbol{\mu}}_{1z}, \widehat{\mathbf{v}}_{1z}) := (\widehat{\boldsymbol{\mu}}_{1z}, \widehat{\mathbf{v}}_{1z})$ and $(\boldsymbol{\mu}_{1\mathbf{a}}, \mathbf{v}_{1\mathbf{a}}) := (\boldsymbol{\mu}_{1z}, \mathbf{v}_{1z})$ into (5). The same operations can be applied to the other modules. The details of this process are provided in [15].

As the measurements \mathbf{y}_l 's go through Modules $A^{[l]}$ and $B_x^{[l]}$, we obtain $(\boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]})$. After feeding $(\boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]})$, $l = 1, \dots, L$, into Module C, the module introduces constraint

$\mathbf{x} = \mathbf{x}^{[1]} = \mathbf{x}^{[2]} = \dots = \mathbf{x}^{[L]}$ and computes the global posterior information $(\widehat{\boldsymbol{\mu}}_{1x}, \widehat{\mathbf{v}}_{1x})$ by considering the true prior of \mathbf{x} . The extrinsic information of $\mathbf{x}^{[l]}$ for each cluster is computed by excluding the prior mean and variance $(\boldsymbol{\mu}_{1x}^{[l]}, \mathbf{v}_{1x}^{[l]})$. The extrinsic information $(\boldsymbol{\mu}_{2x}^{[l]}, \mathbf{v}_{2x}^{[l]})$ is sent to Module $B_z^{[l]}$.

As shown in Fig. 1(c), we introduce layer-dependent damping factors $\beta_z(t)$ and $\beta_x(t)$ to slow the update of Modules $A^{[l]}$ and C , respectively, as follows:

$$(\boldsymbol{\mu}_{2z}^{[l]}(t), \mathbf{v}_{2z}^{[l]}(t)) := \text{Damp}(\boldsymbol{\mu}_{2z}^{[l]}(t), \mathbf{v}_{2z}^{[l]}(t); \beta_z(t)), \quad (9a)$$

$$(\boldsymbol{\mu}_{2x}^{[l]}(t), \mathbf{v}_{2x}^{[l]}(t)) := \text{Damp}(\boldsymbol{\mu}_{2x}^{[l]}(t), \mathbf{v}_{2x}^{[l]}(t); \beta_x(t)), \quad (9b)$$

for $t = 1, \dots, T$, where $\beta_z(t), \beta_x(t) \in [0, 1]$ and

$$\text{Damp}(\mathbf{a}(t), \mathbf{b}(t); \beta(t)) = \begin{bmatrix} \beta(t)\mathbf{a}(t-1) + (1-\beta(t))\mathbf{a}(t) \\ \beta(t)\mathbf{b}(t-1) + (1-\beta(t))\mathbf{b}(t) \end{bmatrix}.$$

We denote the damping factors for Modules $A^{[l]}$ and C by $\boldsymbol{\beta}_z = [\beta_z(1), \dots, \beta_z(T)]$ and $\boldsymbol{\beta}_x = [\beta_x(1), \dots, \beta_x(T)]$, respectively. The structure of deGEC-SR-Net is generic. Compared with conventional generic NNs, deGEC-SR-Net contains fewer parameters because it encodes the estimation knowledge through unfolding. Moreover, deGEC-SR-Net reduces to deGEC-SR if the damping factors are handcrafted without training.

Training is performed on the basis of S samples using the data structure as $(\mathbf{y}^s, \mathbf{A}^s, \mathbf{x}^s)$ for $s = 1, 2, \dots, S$, where transform matrix \mathbf{A}^s and signal \mathbf{x}^s are randomly generated for each sample, and \mathbf{y}^s is obtained using (1). By feeding $(\mathbf{y}^s, \mathbf{A}^s)$ into the network, deGEC-SR-Net generates $\widehat{\boldsymbol{\mu}}_{1x}^s(t)$ at the t -th layer and eventually outputs $\widehat{\boldsymbol{\mu}}_{1x}^s(T)$ at the T -th layer as the reconstructed signal. An optimizer is used to tune damping factors $(\boldsymbol{\beta}_z, \boldsymbol{\beta}_x)$ through back-propagation to minimize the mean square error (MSE) loss function between the true signal \mathbf{x}^s and the estimates $\widehat{\boldsymbol{\mu}}_{1x}^s(t)$ of every layer. The PR problem is disturbed by the global phase rotation, that is, if \mathbf{x} is the solution, then $e^{j\phi}\mathbf{x}$ with any rotation angle ϕ is also the solution. Therefore, the ambiguity of each estimate must be removed by

$$\text{dis}(\mathbf{x}^s, \widehat{\boldsymbol{\mu}}_{1x}^s(t)) = e^{j\phi_t^s} \widehat{\boldsymbol{\mu}}_{1x}^s(t), \quad (10)$$

where $\phi_t^s = \angle((\widehat{\boldsymbol{\mu}}_{1x}^s(t))^H \mathbf{x}^s)$. Consequently, the loss function is defined as

$$\mathcal{L}(\boldsymbol{\beta}_z, \boldsymbol{\beta}_x) = \frac{1}{S} \sum_{s=1}^S \sum_{t=1}^T \|\mathbf{x}^s - \text{dis}(\mathbf{x}^s, \widehat{\boldsymbol{\mu}}_{1x}^s(t))\|_2^2. \quad (11)$$

IV. NUMERICAL RESULTS

We compare deGEC-SR-Net and deGEC-SR in terms of reconstruction accuracy and convergence speed.¹ We consider the PR problem with a size of $(M, N) = (4000, 1000)$. The training and testing sets contain 100 and 1,000 samples, respectively, where each sample $(\mathbf{y}^s, \mathbf{A}^s, \mathbf{x}^s)$ is generated as follows. Transform matrix \mathbf{A}^s is randomly generated from i.i.d. complex Gaussian distribution with zero mean and variance $1/N$. The elements of \mathbf{x}^s are generated by i.i.d. Gaussian-Bernoulli distribution $p(x) = (1-\rho)\delta(x) + \rho\mathcal{N}_C(x; 0, \rho^{-1})$ with sparsity rate $\rho = 0.5$. We set the noise level σ_w^2 to satisfy the specification by using the SNR defined by $\text{SNR} = \mathbb{E}\{\|\mathbf{A}\mathbf{x}\|_2^2\} / (M\sigma_w^2)$. We set

¹An implementation of the algorithm is available on GitHub <https://github.com/Wangchangjen/DeGEC-SR-Net>.

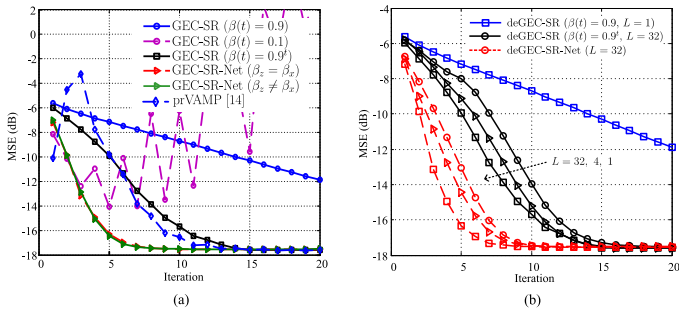


Fig. 2. MSE versus the iteration for (a) GEC-SR and (b) deGEC-SR. The corresponding performance of prVAMP [14], which is by far the most comparable algorithm to GEC-SR, is also shown. Detailed comparisons are available in [15].

TABLE I

LEARNED DAMPING FACTORS FOR DIFFERENT DECENTRALIZATIONS

$\beta(t)$	1	2	3	4	5	6	7	8	9	10
$L = 1$	0.32	0.29	0.26	0.27	0.24	0.20	0.14	0.12	0.12	0.12
$L = 4$	0.42	0.38	0.32	0.33	0.25	0.21	0.15	0.13	0.13	0.13
$L = 32$	0.46	0.41	0.35	0.36	0.25	0.21	0.15	0.13	0.13	0.13

SNR = 20 dB during training, deGEC-SR-Net is implemented using Tensorflow and trained using a PC with an NVIDIA GeForce GTX 2080 Ti GPU. An Adam optimizer with a learning rate of 0.05 and a batch size of 100 is used to identify the damping factors for minimizing the loss function (11). The training procedure is fast because only few samples are required to train the damping factors.

A. Convergence Acceleration

We compare GEC-SR-Net with the original GEC-SR to demonstrate the advantages of the learning parameters through learning unfolding. We use $\beta(t)$ to denote the damping factor if $\beta_z(t) = \beta_x(t)$. Five damping strategies, including constant damping with $\beta(t) = 0.1$ or $0.9 \forall t$, exponentially decreasing damping $\beta(t) = 0.9^t$, and learned damping with $\beta_z = \beta_x$ or $\beta_z \neq \beta_x$, are considered. Fig. 2(a) displays the corresponding MSE performances as a function of the number of iterations. GEC-SR shows instabilities under settings with fast updates (i.e., $\beta(t) = 0.1$). Among the customized tuning processes, exponentially decreasing damping $\beta(t) = 0.9^t$ demonstrates a better trade-off between instabilities and speed than constant damping with $\beta(t) = 0.9$. Therefore, GEC-SR (or deGEC-SR) with exponential damping is used as the benchmark in the remaining experiments. GEC-SR-Net converges rapidly owing to trainable damping factors and can obtain the same accuracy as GEC-SR. Moreover, the performances of GEC-SR-Net with $\beta_z = \beta_x$ and $\beta_z \neq \beta_x$ are comparable. This finding indicates that the number of training parameters can be reduced by half, which speeds up the training of GEC-SR-Net. Therefore, we use $\beta_z = \beta_x$ in GEC-SR-Net in the following experiments.

The corresponding MSE performances of deGEC-SR and deGEC-SR-Net for decentralized cases with $L = \{1, 4, 32\}$ are shown in Fig. 2(b). deGEC-SR-Net can obtain the same accuracy as the original deGEC-SR using fewer iterations. The learned damping factors are listed in Table I and they vary for different layers and decentralizations. In general, the damping

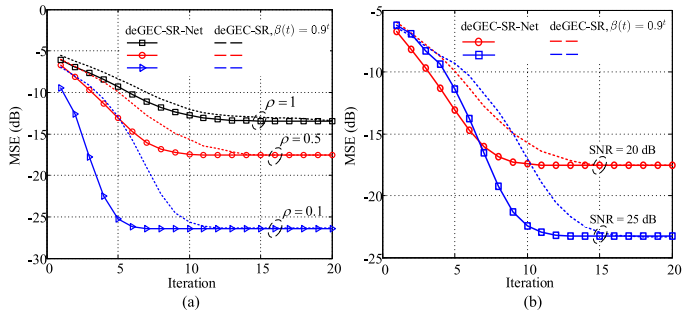


Fig. 3. MSE versus the iteration for deGEC-SR-Net when $L = 32$ under (a) sparsity and (b) SNR mismatch. deGEC-SR-Net is trained under SNR = 20 dB and $\rho = 0.5$.

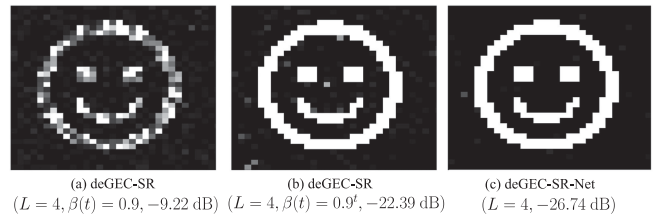


Fig. 4. Reconstruction of a 32×32 image from i.i.d. standard Gaussian measurements for SNR = 10 dB and $T = 10$.

factor for each setting decreases with iterations and reaches a fixed constant after $t = 8$. This result is consistent with the finding that deGEC-SR-Net converges after $t = 8$. In addition, the damping factor increases with decentralization level L . This result is reasonable because adequate damping is necessary to prevent divergence of high decentralizations.

B. Robustness

In the previous experiments, deGEC-SR-Net is trained and tested under the similar scenarios. In this subsection, we examine the robustness of deGEC-SR-Net by testing the previous experiments under different sparsity rates and SNRs. Figs. 3(a) and 3(b) illustrate the corresponding MSEs of deGEC-SR-Net under mismatched sparsity and SNRs. The damping factors are not trained using the same statistical data, which result in the slower convergence speed of deGEC-SR-Net at SNR = 25 dB than at SNR = 20 dB. Nevertheless, the convergence speed of deGEC-SR-Net is higher than that of deGEC-SR. Subsequently, we test deGEC-SR-Net by using the signals obtained from a practical image instead of from the Gaussian-Bernoulli distribution. The reconstruction results of the evaluated algorithms under 10 iterations are shown in Fig. 4. The result indicates that deGEC-SR-Net performs best.

V. CONCLUSION

We develop a learning NN called deGEC-SR-Net to address the PR problem by unfolding the deGEC-SR algorithm. deGEC-SR-Net demonstrates excellent performance in terms of accuracy and speed and is robust to input distribution or noise level mismatch.

REFERENCES

- [1] L. Bian, J. Suo, G. Zheng, K. Guo, F. Chen, and Q. Dai, "Fourier ptychographic reconstruction using Wirtinger flow optimization," *Opt. Express*, vol. 23, no. 4, pp. 4856–4866, 2015.
- [2] R. P. Millane, "Phase retrieval in crystallography and optics," *J. Opt. Soc. America*, vol. 7, no. 3, pp. 394–411, Mar. 1990.
- [3] DL Misell, "A method for the solution of the phase problem in electron microscopy," *J. Phys. D: Appl. Phys.*, vol. 6, no. 1, 1973.
- [4] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev, "Phase retrieval with application to optical imaging: A contemporary overview," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 87–109, May 2015.
- [5] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of the phase from image and diffraction plane pictures," *Optik*, vol. 35, no. 2, pp. 237–246, 1972.
- [6] J. R. Fienup, "Reconstruction of an object from the modulus of its fourier transform," *Opt. Lett.*, vol. 3, no. 1, pp. 27–29, 1978.
- [7] E. J. Candes, T. Strohmer, and V. Voroninski, "Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming," *Commun. Pure Applied Math.*, vol. 66, no. 8, pp. 1241–1274, Nov. 2013.
- [8] E. J. Candes, X. Li, and M. Soltanolkotabi, "Phase retrieval via Wirtinger flow: Theory and algorithms," *IEEE Trans. Inf. Theory*, vol. 61, no. 4, pp. 1985–2007, Jan. 2015.
- [9] T. Goldstein and C. Studer, "Phasemax: Convex phase retrieval via basis pursuit," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2675–2689, Apr. 2018.
- [10] C. T. O. Dhifallah and Y. M. Lu, "Phase retrieval via linear programming: Fundamental limits and algorithmic improvements," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, 3–6 Oct. 2017, pp. 1071–1077.
- [11] Z. Yuan and H. Wang, "Phase retrieval via reweighted wirtinger flow," *Appl. Opt.*, vol. 56, no. 9, pp. 2418–2427, 2017.
- [12] P. Schniter and S. Rangan, "Compressive phase retrieval via generalized approximate message passing," *IEEE Trans. Signal Process.*, vol. 63, no. 4, pp. 1043–1055, Feb. 2015.
- [13] B. Rajaei, S. Gigan, F. Krzakala, and L. Daudet, "Robust phase retrieval with the swept approximate message passing (prSAMP) algorithm," *Image Process. Line*, vol. 7, pp. 43–55, Jan. 2017.
- [14] M. Sharma, C. A. Metzler, S. Nagesh, O. Cossairt, R. G. Baraniuk, and A. Veeraraghavan, "Inverse scattering via transmission matrices: Broadband illumination and fast phase retrieval algorithms," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 95–108, May 2020.
- [15] C. Wang, C. Wen, S. Tsai, and S. Jin, "Decentralized expectation consistent signal recovery for phase retrieval," *IEEE Trans. Signal Process.*, vol. 68, pp. 1484–1499, 2020.
- [16] M. Pretti, "A message-passing algorithm with damping," *J. Stat. Mech.*, vol. 2005, Nov. 2005, Art. no. P11008.
- [17] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 399–406.
- [18] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," 2019, *arXiv:1912.10557*.
- [19] H. He, S. Jin, C. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 5, pp. 77–83, Oct. 2019.
- [20] D. Ito, S. Takabe, and T. Wadayama, "Trainable ISTA for sparse signal recovery," *IEEE Trans. Signal Process.*, vol. 67, no. 12, pp. 3113–3125, Dec. 2019.
- [21] A. Fletcher, M. Sahraee-Ardakan, S. Rangan, and P. Schniter, "Expectation consistent approximate inference: Generalizations and convergence," in *Proc. IEEE Int. Symp. Inf. Theory*, Barcelona, Spain, 10–15 Jul. 2016, pp. 190–194.
- [22] X. Meng, S. Wu, and J. Zhu, "A unified bayesian inference framework for generalized linear models," *IEEE Signal Process. Lett.*, vol. 25, no. 3, pp. 398–402, Mar. 2018.
- [23] J. Zhu, Q. Yuan, C. Song, and Z. Xu, "Phase retrieval from quantized measurements via approximate message passing," *IEEE Signal Process. Lett.*, vol. 26, no. 7, pp. 986–990, Jul. 2019.