Deep unrolling networks with recurrent momentum acceleration for nonlinear inverse problems

Zhou, Qingping; Qian, Jiayu; Tang, Junqi; Li, Jinglai

DOI:
10.1088/1361-6420/ad35e3

OPEN ACCESS
IOP Publishing

Inverse Problems **40** (2024) 055014 (20pp) https://doi.org/10.1088/1361-6420/ad35e3

Deep unrolling networks with recurrent momentum acceleration for nonlinear inverse problems

Qingping Zhou[1], Jiayu Qian[1], Junqi Tang[2] and Jinglai Li[2,*]

[1] School of Mathematics and Statistics, HNP-LAMA, Central South University, Changsha 410083, People's Republic of China
[2] School of Mathematics, University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom

E-mail: j.li.10@bham.ac.uk

Received 19 September 2023; revised 8 March 2024
Accepted for publication 20 March 2024
Published 2 April 2024

Abstract

Combining the strengths of model-based iterative algorithms and data-driven deep learning solutions, deep unrolling networks (DuNets) have become a popular tool to solve inverse imaging problems. Although DuNets have been successfully applied to many linear inverse problems, their performance tends to be impaired by nonlinear problems. Inspired by momentum acceleration techniques that are often used in optimization algorithms, we propose a recurrent momentum acceleration (RMA) framework that uses a long short-term memory recurrent neural network (LSTM-RNN) to simulate the momentum acceleration process. The RMA module leverages the ability of the LSTM-RNN to learn and retain knowledge from the previous gradients. We apply RMA to two popular DuNets—the learned proximal gradient descent (LPGD) and the learned primal-dual (LPD) methods, resulting in LPGD-RMA and LPD-RMA, respectively. We provide experimental results on two nonlinear inverse problems: a nonlinear deconvolution problem, and an electrical impedance tomography problem with limited boundary measurements. In the first experiment we have observed that the improvement due to RMA largely increases with respect to the nonlinearity of the problem. The results of the

[*] Author to whom any correspondence should be addressed.

second example further demonstrate that the RMA schemes can significantly improve the performance of DuNets in strongly ill-posed problems.

Keywords: inverse problems, deep unrolling networks,
momentum acceleration, learned primal-dual,
learned proximal gradient descent, recurrent neural network

1. Introduction

Many image processing tasks can be cast as an inverse problem, i.e. to recover an unknown image x from indirect measurements y

$$y = \mathcal{A}(x) + \epsilon, \tag{1}$$

where $x \in X$, $y \in Y$, $\mathcal{A}$ represents a forward measurement operator and ϵ is the observation noise. Problems that can be formulated with equation (1) include denoising [18], compressive sensing [37], computed tomography reconstruction [2, 3], phase retrieval [29], optical diffraction tomography [30], electrical impedance tomography [10, 28, 33] and so on.

A common challenge in solving the inverse problems is that they are typically ill-posed and as such regularization techniques are often used to ensure a unique and stable solution. Classical regularization techniques introduce an explicit regularization term in the formulation, yielding a variational regularization problem that can be solved with iterative algorithms, such as the alternating direction method of multipliers (ADMM) [5] and the primal-dual hybrid gradient (PDHG) method [6]. Although such methods can obtain well-posed solutions of the inverse problems, they have a number of limitations: most notably, inaccuracy regularizing assumption, need for parameter tuning, and mathematical inflexibility. A possible remedy to these limitations is to implicitly regularize the inverse problem by replacing certain modules in the iterative procedure with deep neural networks [1, 2, 34, 37], a type of method commonly referred to as the deep unrolling networks (DuNets).

DuNets, pioneered by Gregor and LeCun in [13], have achieved great empirical success in the field of inverse problems and image processing [26, 32, 39] in the past few years. DuNets combine traditional model-based optimization algorithms with learning-based deep neural networks, yielding an interpretable and efficient deep learning framework for solving inverse imaging problems. It should be noted that most of the aforementioned studies have focused on solving inverse problems with linear or linearized forward operators. On the other hand, there has been relatively little research on applying DuNets to nonlinear inverse problems, such as optical diffraction tomography and electrical impedance tomography (EIT). This paper attempts to bridge this gap by applying DuNets to the nonlinear inverse problems. In such problems, nonlinearity may pose additional difficulty for the DuNet methods, as the gradient of the forward operator varies significantly as the iterations proceed. Intuitively, the performance of DuNets may be improved if the previous gradient information is included. In this work we will draw on momentum acceleration (MA), an acceleration strategy commonly used in optimization algorithms, and the recurrent neural networks (RNN) techniques to improve the performance of the deep unrolling networks. Specifically, we propose a recurrent momentum acceleration (RMA) module that utilizes a long short-term memory recurrent neural network (LSTM-RNN) to represent the momentum acceleration term. The RMA module exploits the LSTM-RNN's capacity to remember previous inputs over extended periods and learn from them, thereby providing information from the entire gradient history. In particular we apply it to two popular DuNets: the learned proximal gradient descent (LPGD) [25] and the learned primal-dual (LPD) [2]. We refer to the resulting algorithms as LPGD-RMA and LPD-RMA respectively. It should be noted that several existing works propose to use the iteration history to improve the performance of the unrolling algorithms. For example, [2] extends the state space to allow the algorithm some 'memory' between the iterations, and [17] develops a history-cognizant unrolling of the proximal gradient descent where the outputs of all the previous regularization units are used. As a comparison, our RMA method employs the previous gradients that are combined via a flexible RNN model learned from data.

The remainder of this paper is organized as follows. In section 2, we review two widely used types of deep unrolling models: the LPGD and the LPD methods. In section 3, we present our RMA formulation, and incorporate it with both LPGD and LPD, yielding LPGD-RMA and LPD-RMA. Numerical experiments performed on two nonlinear inverse problems are reported in section 4. Finally section 5 offers some concluding remarks.

2. Deep unrolling networks

We start with the variational formulation for solving inverse problems of the form (1). These methods seek to solve the following minimization problem that includes a data consistency term $\mathcal{D}(\cdot,\cdot):Y\times Y\to\mathbb{R}$ and regularization term $\mathcal{R}(\cdot):X\to\mathbb{R}$:

$$\arg\min_{x\in X}\mathcal{D}\left(\mathcal{A}\left(x\right),y\right)+\lambda\mathcal{R}\left(x\right), \tag{2}$$

where λ is the regularization parameter balancing $\mathcal{R}$ against $\mathcal{D}$. The regularizer $\mathcal{R}$ encodes prior information on x representing desired solution properties. Common regularization functions include Tikhonov regularization, total variation (TV), wavelets, and sparsity promoting dictionary [4], to name a few. Besides Tikhonov, all the aforementioned regularization functions are non-smooth, and as a result equation (2) is typically solved by the first-order algorithms, such as the proximal gradient descent (PGD) algorithm [11], the variable splitting scheme [5] and the primal-dual hybrid gradient (PDHG) method [6]. These algorithms typically involve rather expensive and complex iterations. The basic idea of DuNets is to use a learned operator, represented by a deep neural network, to model the iterations. We here focus on two archetypes of the deep unrolling models: the LPGD method with both shared [24, 25] and independent weights [7, 15], and the LPD method [2].

2.1. Learned proximal gradient descent method

Starting from an initial value x_0, PGD performs the following iterates until convergence:

$$s_t=x_{t-1}-\alpha_t\nabla_{x_{t-1}}\mathcal{D}\left(\mathcal{A}\left(x_{t-1}\right),y\right), \tag{3a}$$

$$x_t=\mathcal{P}_{\lambda\mathcal{R}}\left(s_t\right), \tag{3b}$$

where α_t is the step size and the proximal gradient descent $\mathcal{P}_{\lambda\mathcal{R}}(\cdot)$ is defined by

$$\mathcal{P}_{\lambda\mathcal{R}}\left(x\right)=\arg\min_{x'\in X}\frac{1}{2}\|x'-x\|_X^2+\lambda\mathcal{R}\left(x'\right).$$

Simply speaking, the LPGD method replaces the proximal operators $\mathcal{P}_{\lambda\mathcal{R}}$ with a neural network Ψ_{θ_t} [7, 15, 36], and thus it allows one to learn how to effectively combine the previous update with the gradient update direction instead of a pre-determined updating scheme. The complete algorithm of LPGD is outlined in algorithm 1. In the standard LPGD algorithm, the network in each iteration uses its own weights θ_t, and a popular variant of the method is to use shared weights over all the networks, i.e. restricting $\theta_1 = \ldots = \theta_T$. We refer to [24, 25] for more details.

Algorithm 1. LPGD algorithm.

```
Input: $x_0 \in X$
Output: $x_T$
for $t = 1, \ldots, T$ do
    $g_{t-1} = \nabla_{x_{t-1}} \mathcal{D}(\mathcal{A}(x_{t-1}), y)$
    $x_t = \Psi_{\theta_t}(x_{t-1}, g_{t-1})$
end for
```

2.2. *Learned primal-dual method*

Adler and Öktem first introduced the partially learned primal-dual approach as an extension of iterative deep neural networks in [1] and further elaborated it into the LPD approach in [2]. The method is based on PDHG, another popular algorithm for solving the non-differentiable optimization problem (2). The PDHG iteration is given by

$$\begin{cases} u_{t+1} = \operatorname{prox}_{\sigma \mathcal{D}^*}\left(u_t + \sigma \mathcal{A}\left(\bar{x}_t\right)\right) \\ x_{t+1} = \operatorname{prox}_{\tau \mathcal{R}}\left(x_t - \tau \left[\partial \mathcal{A}\left(x_t\right)\right]^* \left(u_{t+1}\right)\right) \\ \bar{x}_{t+1} = x_{t+1} + \gamma\left(x_{t+1} - x_t\right), \end{cases} \tag{4}$$

where σ, τ, γ are predefined parameters, $\mathcal{D}^*$ denotes the Fenchel conjugate of $\mathcal{D}$, and $\left[\partial \mathcal{A}\left(x_t\right)\right]^*$ is the adjoint of the Fréchet derivative of $\mathcal{A}$ at point x_t. The LPD method is built upon equation (4), and the main idea is to replace $\operatorname{prox}_{\sigma \mathcal{D}^*}$ and $\operatorname{prox}_{\tau \mathcal{R}}$ with neural network models that are learned from data. Once the models are learned, the reconstruction proceeds via the following iterations:

$$\begin{cases} u_t = \Gamma_{\theta_t^d}\left(u_{t-1}, \mathcal{A}\left(x_{t-1}\right), y\right) \\ x_t = \Lambda_{\theta_t^p}\left(x_{t-1}, \left[\partial \mathcal{A}\left(x_{t-1}\right)\right]^* \left(u_t\right)\right), \end{cases} \quad \text{for} \quad t = 1, \ldots, T. \tag{5}$$

The complete LPD algorithm is given in algorithm 2. It is important to note that the LPD method often enlarges both the primal and dual spaces to allow some 'memory' between iterations [2]. In particular, it defines $x_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(N_{\text{primal}})}] \in X^{N_{\text{primal}}}$ and $u_t = [u_t^{(1)}, u_t^{(2)}, \ldots, u_t^{(N_{\text{dual}})}] \in Y^{N_{\text{dual}}}$. Additionally, $\Lambda_{\theta_t^p} : X^{N_{\text{primal}}} \times X^{N_{\text{primal}}} \to X^{N_{\text{primal}}}$ corresponds to dual and primal networks, which have different learned parameters but with the same architecture for each iteration. A typical initialization is $x_0 = [0, \ldots, 0]$ and $u_0 = [0, \ldots, 0]$, where 0 is the zero element in the primal or dual space. We refer to [2] for details on the LPD method.

Algorithm 2. LPD algorithm.

```
Input: $x_0 \in X^{N_{\text{primal}}}, u_0 \in Y^{N_{\text{dual}}}$
Output: $x_T^{(1)}$
for $t = 1, \ldots, T$ do
    $u_t = \Gamma_{\theta_t^d}\left(u_{t-1}, \mathcal{A}(x_{t-1}^{(2)}), y\right)$
    $g_t = \left[\partial \mathcal{A}(x_{t-1}^{(1)})\right]^* (u_t^{(1)})$
    $x_t = \Lambda_{\theta_t^p}\left(x_{t-1}, g_t\right)$
end for
```

3. Deep unrolling networks with momentum acceleration

The conventional DuNets, exemplified by LPGD and LPD, only use the current gradient, ignoring a large amount of historical gradient data. As has been discussed, these methods can be improved by adopting the momentum acceleration (MA) strategies that are frequently used in optimization methods. In this section we will present such momentum accelerated DuNet methods.

3.1. Momentum acceleration methods

We here discuss the conventional explicit MA scheme and the one based on RNN.

3.1.1. Explicit momentum acceleration. Momentum-based acceleration methods, like Nesterov's accelerated gradient [27] and adaptive moment estimation [20], are well-established algorithms for speeding up the optimization procedure and have vast applications in machine learning [31]. The classical gradient descent with MA utilizes the previous 'velocity' v_{t-1} at each iteration to perform extrapolation and generates the new update as:

$$v_t = \gamma v_{t-1} - \eta g_{t-1} \tag{6a}$$

$$x_t = x_{t-1} + v_t, \tag{6b}$$

where g_{t-1} is the gradient of the objective function evaluated at x_{t-1}, η is the step size, and $\gamma \in [0,1)$ is the momentum coefficient controlling the relative contribution of the current gradient and the previous velocity. Equation (6) can be rewritten as:

$$\begin{aligned} v_t &= \gamma v_{t-1} - \eta g_{t-1} = \gamma\left(\gamma v_{t-2} - \eta g_{t-2} - \eta g_{t-1}\right) \\ &= \ldots = \gamma^t v_0 - \gamma^{t-1}\eta g_0 - \ldots - \eta g_{t-1}, \end{aligned} \tag{7}$$

which shows that the current velocity is essentially a weighted average of all the gradients (assuming $v_0 = 0$). As mentioned above, the momentum coefficient γ controls how much information from previous iterations is used to compute the new velocity v_t, and therefore needs to be chosen carefully for a good performance of the method. However, the optimal value for the parameter is problem-specific and typically requires manual tuning [23, 31].

3.1.2. Momentum acceleration via RNN. The conventional momentum method utilizes a fixed formula (a linear combination of all the gradients) to calculate the present velocity v_t. In this section we introduce a more flexible scheme that uses the recurrent neural networks to learn the velocity term [16], which we refer to as the RMA method. In particular we use the LSTM based RNN, which is briefly described as follows. At each time step t we employ a neural network to compute the 'velocity' v_t. The neural network has three inputs and three outputs. These inputs include the current gradient input g_{t-1}, the cell-state c_{t-1} (carrying long-memory information) and the hidden state h_{t-1} (carrying short-memory information), where the latter two inputs are both inherited from the previous steps. The outputs of the network are v_t, h_t and c_t. We formally write this neural network model as,

$$\left(v_t, h_t, c_t\right) = \Xi_\vartheta\left(g_{t-1}, h_{t-1}, c_{t-1}\right), \tag{8}$$

where ϑ is the neural network parameters, and leave the details of it in appendix. As one can see this network integrates the current gradient g_{t-1} and the information from previous step h_{t-1} and c_{t-1} to produce the velocity v_t that is used in DuNets. Finally we note that other RNN

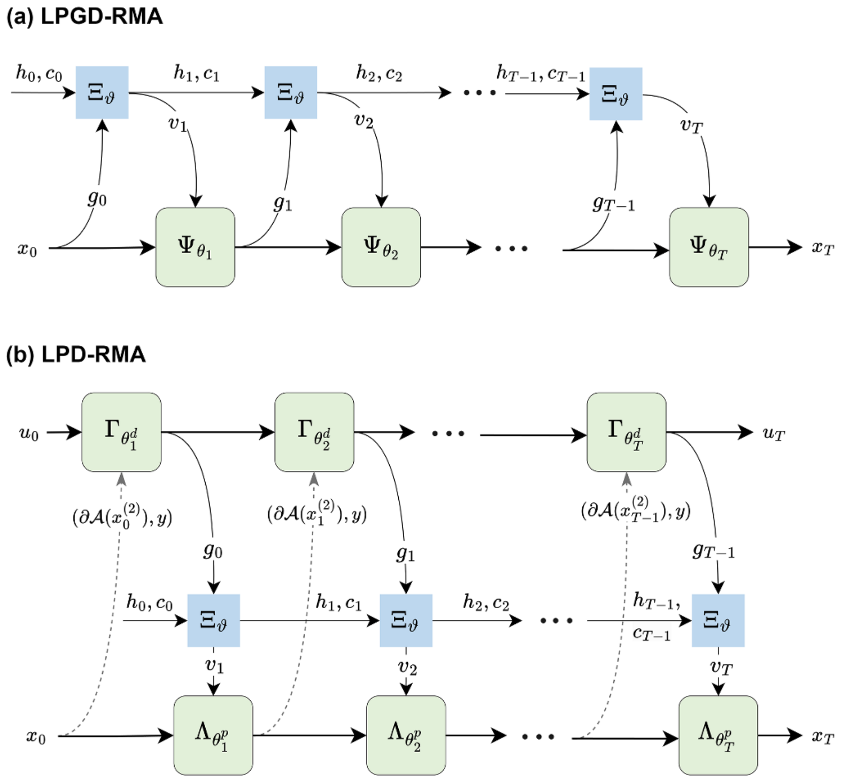


Figure 1. The model architectures of DuNets-RMA. The RMA module is constructed via a deep LSTM-RNN, denoted as Ξ_ϑ.

models such as the Gated Recurrent Unit (GRU) [8] can also be used here. In our numerical experiments we have tested both LSTM and GRU, and found no significant difference between the performances of the two approaches, which is consistent with some existing works, e.g. [9, 12]. As such, we only report our experimental results with LSTM-RNN in this work.

3.2. LPGD and LPD with MA

Inserting the MA module into the DuNets algorithms is rather straightforward. In this section, we use LPGD and LPD as examples, while noting that they can be implemented in other DuNets in a similar manner.

3.2.1. LPGD. The explicit MA can be easily incorporated with LPGD. A notable difference is that in the unrolling methods, g_t is not the gradient of the objective function, which may be nondifferentiable or not explicitly available. In LPGD, g_t is taken to be the gradient of the data fidelity term only, i.e. $g_t = \nabla_{x_{t-1}}\mathcal{D}(\mathcal{A}(x_{t-1}), y)$. The main idea here is to replace g_{t-1} in algorithm 1 by v_t calculated via equation (6*a*), yielding the LPGD-MA method (algorithm 3.). Similarly by inserting the LSTM model into algorithm, (1), we obtain the LPGD-RMA algorithm (algorithm 4), as illustrated in figure 1(a). Finally, recall that LPGD also has a shared-weights version (referred to as LPGDSW), and correspondingly we have LPGDSW-MA and LPGDSW-RMA, which will also be tested in our numerical experiments.

Algorithm 3. LPGD-MA.

```
Input: $x_0 \in X$, $v_0 = 0$
Output: $x_T$
for $t = 1, \ldots, T$ do
    $g_{t-1} = \nabla_{x_{t-1}} \mathcal{D}(\mathcal{A}(x_{t-1}), y)$
    $v_t = \gamma v_{t-1} - \eta g_{t-1}$
    $x_t = \Psi_{\theta_t}(x_{t-1}, v_t)$
end for
```

Algorithm 4. LPGD-RMA algorithm.

```
Input: $x_0 \in X, h_0 = 0, c_0 = 0$
Output: $x_T$
for $t = 1, \ldots, T$ do
    $g_{t-1} = \nabla_{x_{t-1}} \mathcal{D}(\mathcal{A}(x_{t-1}), y)$
    $(v_t, h_t, c_t) = \Xi_{\vartheta}(g_{t-1}, h_{t-1}, c_{t-1})$
    $x_t = \Psi_{\theta_t}(x_{t-1}, v_t)$
end for
```

3.2.2. LPD. The integration of the MA schemes and LPD is a bit different. Namely, in LPGD, the velocity is constructed based on the gradient of the data fidelity term, while in LPD, we build it upon $g_{t-1} = [\partial \mathcal{A}(x_{t-1}^{(1)})]^* u_t^{(1)}$. By inserting the explicit MA formula (6*a*) into algorithm 2 we obtain the LPD-MA method (algorithm 5). The LPD-RMA method depicted in figure 1(b) can be constructed similarly: one simply replaces the explicit MA formula in algorithm 2 with the RMA module equation (8), and the complete algorithm is outlined in algorithm 6.

Algorithm 5. LPD-MA algorithm.

```
Input: $x_0 \in X^{N_{\text{primal}}}, u_0 \in Y^{N_{\text{dual}}}$
Output: $x_T^{(1)}$
for $t = 1, \ldots, T$ do
    $u_t = \Gamma_{\theta_t^d}\left(u_{t-1}, \mathcal{A}(x_{t-1}^{(2)}), y\right)$
    $g_{t-1} = \left[\partial \mathcal{A}(x_{t-1}^{(1)})\right]^* u_t^{(1)}$
    $v_t = \gamma v_{t-1} - \eta g_{t-1}$
    $x_t = \Lambda_{\theta_t^p}(x_{t-1}, v_t)$
end for
```

Algorithm 6. LPD-RMA algorithm.

```
Input: $x_0 \in X^{N_{\text{primal}}}, u_0 \in Y^{N_{\text{dual}}}, h_0 = 0, c_0 = 0$
Output: $x_T^{(1)}$
for $t = 1, \ldots, T$
    $u_t = \Gamma_{\theta_t^d}\left(u_{t-1}, \mathcal{A}(x_{t-1}^{(2)}), y\right)$
    $g_{t-1} = \left[\partial \mathcal{A}(x_{t-1}^{(1)})\right]^* (u_t^{(1)})$
    $(v_t, h_t, c_t) = \Xi_{\vartheta}(g_{t-1}, h_{t-1}, c_{t-1})$
    $x_t = \Lambda_{\theta_t^p}(x_{t-1}, v_t)$
end for
```

4. Experiments and results

In this section, we present our numerical experiments on two nonlinear inverse problems: a nonlinear deconvolution and an EIT image reconstruction.

4.1. Implementation details

To make a fair comparison, for various DuNet methods, we adjust the number of unrolled iterations to ensure that all the methods have approximately the same of number of training parameters. Specifically, for the LPGD-type of methods, we set the unrolling iterations to 20 for LPGD-RMA and to 43 for both LPGD and LPGD-MA. The outputs of the proximal operator unit are first concatenated with the estimated direction from the RMA module and then combined using a convolutional layer with a 3 × 3 kernel size and 32 output channels before being fed to the subsequent block. The primal subnetwork consists of two convolutional layers of kernel size 3 × 3 and 32 output channels. The convolutional layers are followed by a parametric rectified linear units (PReLU) activation function. The output convolutional layer is designed to match a desired number of channels and does not include any nonlinear activation function. In LPD-RMA, we use 10 unrolling iterations, while in other LPD methods, we adjust it to 22. The number of data that persists between the iterates be $N_{\text{primal}} = 5, N_{\text{dual}} = 5$. The primal subnetwork $\Gamma_{\theta_i^d}$ is the same as that used in LPGD-based methods. The dual subnetwork consists of one convolutional layer of kernel size 3 × 3 and output channels 32, and the other setting is the same as the primal subnetwork.

All networks are trained end-to-end using Adam optimizer [20] to minimize the empirical loss (9). We use a learning rate schedule according to the cosine annealing, i.e. the learning rate in step t is

$$\zeta_t = \frac{\zeta_0}{2}\left(1 + \cos\left(\pi \frac{t}{t_{\max}}\right)\right),$$

where the initial learning rate ζ_0 is set to be 10^{-3}. We also let the parameter β_2 of the ADAM optimizer [20] to be 0.99 and keep all other parameters as default. We perform global gradient norm clipping [38], limiting the gradient norms to 1 to improve training stability. We use a batch size of 32 for the nonlinear convolution example and 1 for the EIT problem. For the DuNets-MA methods, we choose $\gamma = 0.9$ and $\eta = 10^{-3}$. We train all models with 20 epochs and keep a set of trainable parameters that achieve minimal validation losses. We do not enforce any constraint on the trainable parameters during training.

All experiments are run on an Intel Xeon Golden 6248 CPU and an NVIDIA Tesla V100 GPU. The nonlinear deconvolution example is run entirely on the GPU. The forward and adjoint operators in the EIT experiments are run on the CPU as the `pyEIT` toolbox used is not computationally parallelizable and runs faster on the CPU. The training time for a single epoch is approximately 4 min in the nonlinear convolution example with 10 000 training samples, and 60 min for the EIT example with 400 training samples.

We use the ℓ_2 loss function on the outputs from all the stages. Specifically, given the paired samples $\{x_i, y_i\}, i = 1, \ldots, N$, the training objective is defined as:

$$\mathrm{L}(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \|\hat{x}_i - x_i\|^2. \tag{9}$$

Here, $\hat{x}_i$ is the reconstruction, and Θ presents the set of trainable parameters.

4.2. A nonlinear deconvolution problem

4.2.1. Problem setting. We consider a nonlinear deconvolution problem which is constructed largely following [42]. For each input $\mathbf{x} = [x_1, x_2, \ldots, x_n]'$ consisting of n elements, the forward problem is defined as

$$y(\mathbf{x}) = a \cdot \mathbf{x}'\mathbf{W}_2\mathbf{x} + \mathbf{w}_1'\mathbf{x} + b. \tag{10}$$

Here $\mathbf{w}_1 = [w_1^1, w_1^2, \ldots, w_1^n]'$ is the first-order Volterra kernel, which contains the coefficients of the Volterra series' linear part. The second-order Volterra kernel, denoted as $\mathbf{W}_2$, is structured as follows:

$$\mathbf{W}_2 = \begin{bmatrix} w_2^{1,1} & w_2^{1,2} & \cdots & w_2^{1,n} \\ 0 & w_2^{2,2} & \cdots & w_2^{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_2^{n,n} \end{bmatrix}.$$

It is important to note that the parameter a controls the degree of nonlinearity in the deconvolution problem.

4.2.2. Training and testing datasets. We assume that the unknown x is on 53 mesh grid points, and meanwhile we choose the nonlinear kernel with size 9 and stride 4, and the dimension of the observed data y is 12. The first-order and second-order Volterra kernels in equation (10) are derived using the methods described in section 3 of [21]. We consider four sets of experiments with different coefficients in equation (10): $a = 0, 1, 2, 4$. We generate the ground truth by sampling from a TV prior (see chapter 3.3 in [19]) and then obtain the observation data by (10). We employ 12 000 randomly generated paired samples, where 10 000 pairs are used as the training set, and the remaining 1000/1000 pairs are used as the validation/test sets.

4.2.3. Results and discussion

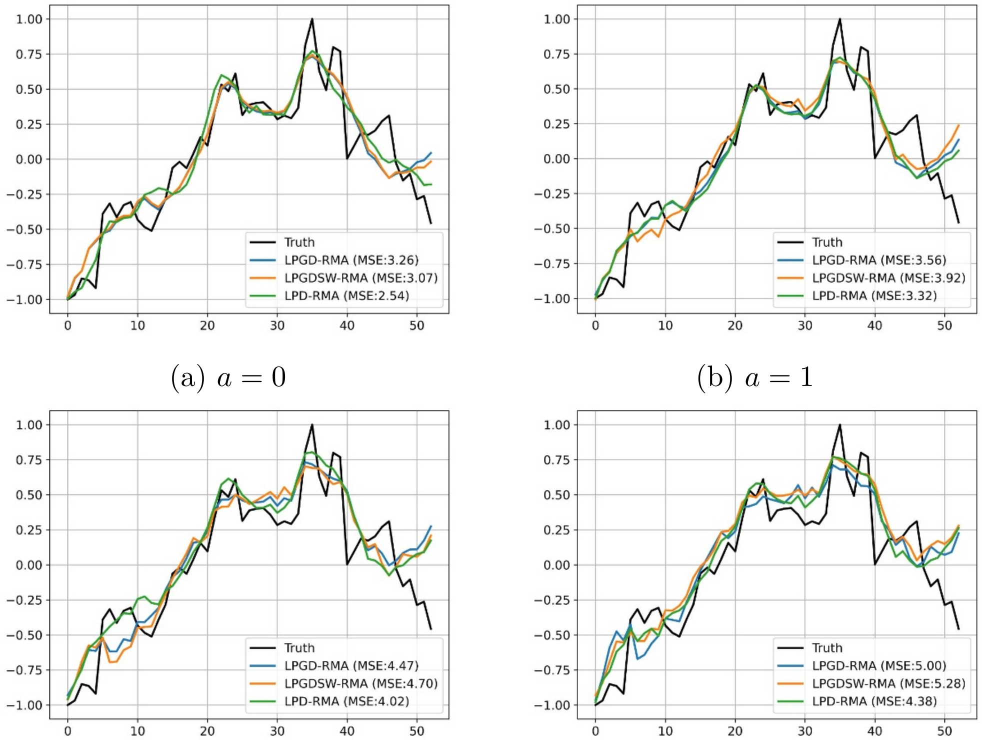
4.2.3.1. Benefit of the RMA scheme. We assess the performance of RMA with the three unrolling methods LPGDSW, LPGD and LPD, and in each method we implement the following three different cases: without acceleration, with the conventional MA module and with the RMA module; as such there are 9 schemes implemented in total. All hyperparameters in the tested methods are manually tuned for optimal performance or automatically chosen as described in the aforementioned references. Table 1 demonstrates the performance of each method in terms of the mean-square error (MSE) on four different settings: $a = 0, 1, 2, 4$. The visual comparison can be found in figure 2. We summarize our findings as the following:

(i) when $a = 0$, the MSE values of each type of DuNets method are almost the same, which is not surprising as the gradient of the forward operator is constant;

(ii) when $a > 0$, DuNets with RMA outperform the state-of-the-art methods by a rather large margin (e.g. LPD-RMA outperforms the LPD method by 8.0%, 12.0%, and 16.0% in terms of MSE for $a = 1, 2, 4$ respectively), suggesting that the RMA module can significantly improve the performance of DuNets, especially for problems that are highly nonlinear;

Table 1. MSE results of the DuNets methods under different a values. The result of LPGD is not reported as it fails to converge. The best results are indicated in orange color.

	$a=0$	$a=1$	$a=2$	$a=4$
LPGD	—	—	—	—
LPGD-MA	3.21×10^{-2}	5.37×10^{-2}	6.49×10^{-2}	7.76×10^{-2}
LPGD-RMA	3.23×10^{-2}	3.56×10^{-2}	4.43×10^{-2}	4.97×10^{-2}
LPGDSW	3.01×10^{-2}	4.61×10^{-2}	5.88×10^{-2}	6.85×10^{-2}
LPGDSW-MA	3.02×10^{-2}	4.54×10^{-2}	5.32×10^{-2}	5.78×10^{-2}
LPGDSW-RMA	3.01×10^{-2}	3.89×10^{-2}	4.68×10^{-2}	5.24×10^{-2}
LPD	2.69×10^{-2}	3.65×10^{-2}	4.61×10^{-2}	5.17×10^{-2}
LPD-MA	2.68×10^{-2}	3.71×10^{-2}	4.65×10^{-2}	5.22×10^{-2}
LPD-RMA	2.67×10^{-2}	3.35×10^{-2}	4.04×10^{-2}	4.33×10^{-2}



(a) $a=0$

(b) $a=1$

(c) $a=2$

(d) $a=4$

Figure 2. Deconvolution results and their corresponding MSE values for all DuNets-RMA methods.

(iii) the conventional MA module can also improve the performance, but RMA clearly outperforms it in all the nonlinear cases;

(iv) LPD-RMA method consistently achieves the best results in terms of MSE.

Table 2. Mean MSE values of the LPD-RMA models with $L = 1,2,3$ and $n = 30,50,70$. Evaluation is done via repeating the experiment 10 times. The number of trainable parameters is also reported below the MSE value in parentheses.

n \ L	1	2	3
30	3.37×10^{-2} (94 193)	3.36×10^{-2} (101 363)	3.34×10^{-2} (108 533)
50	3.35×10^{-2} (103 353)	3.34×10^{-2} (121 303)	3.33×10^{-2} (139 253)
70	3.34×10^{-2} (114 913)	3.35×10^{-2} (148 443)	3.32×10^{-2} (181 973)

Next we will test the sensitivity of the RMA module with respect to both the network structure and the data size. Since the LPGD-type methods are significantly outperformed by the LPD-type ones in this example, we only use the LPD-type methods in these tests.

4.2.3.2. Sensitivity to the RMA structure. We discuss here the choice for the architecture of the RMA modules, i.e. the number of hidden layers L and the hidden size n of the LSTM layer. To avoid overfitting, we limit the ranges of the hidden layers as $L \in \{1,2,3\}$ and the hidden size as $n \in \{30,50,70\}$. Table 2 demonstrates the results of LPD-RMA trained with different network structures in the setting where $a = 1$. We observe that in all these settings the LPD-RMA yields similar results, indicating that the algorithm is rather robust provided that the parameter values are within a reasonably defined range. With extensive numerical tests, we have found that a reasonable choice of L may be $L \in \{1,2\}$ in moderate dimensions, and n can be chosen to be approximately the same as the dimensionality of the unknown variables. We have also tested the methods for $a = 2,4$, where the results are qualitatively similar to those shown in table 2, and hence are omitted here.

4.2.3.3. Sensitivity to data size. To assess the data efficiency of the proposed methods, we train them on different data sizes. The data size is measured as the percentage of the total available training data, and the MSE results are plotted against it in figure 3. The figure shows that LPD-RMA is considerably more data-efficient than LPD and LPD-MA. Interestingly after the data size increases to over 50%, the use of the conventional MA module cannot improve the performance of LPD, while LPD-RMA consistently achieves the best accuracy across the whole range.

4.3. Electrical impedance tomography

4.3.1. Problem setting. EIT is a nondestructive imaging technique that aims at reconstructing the inner conductivity distribution of a medium from a set of voltages registered on the boundary of the domain by a series of electrodes [41].

In this example, we consider a bounded domain $\Omega \subseteq \mathbb{R}^2$ with a boundary $\partial\Omega$ containing certain conducting materials whose electrical conductivity is defined by a positive spatial function $\sigma(x) \in L^\infty(\Omega)$. Next, we assume that L different electrical currents are injected into the

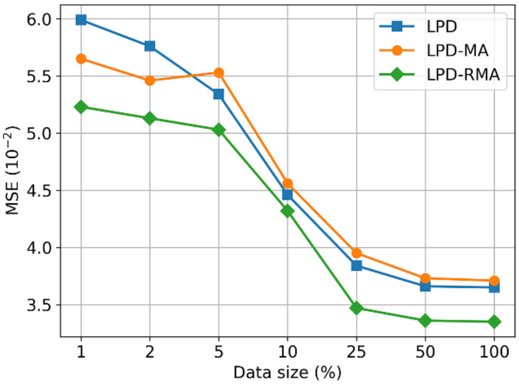


Figure 3. The MSE results plotted against the data size for $a = 1$.

boundary of $\partial\Omega$, and the resulting electrical potential should satisfy the following governing equations with the same coefficient but different boundary conditions:

$$\begin{cases} \nabla\cdot(\sigma\nabla u) = 0 & \text{in } \Omega \\ u + z_l\sigma\frac{\partial u}{\partial e} = V_l & \text{on } E_l, \quad l = 1,\dots,L \\ \int_{E_l}\sigma\frac{\partial u}{\partial e}\,\mathrm{d}s = I_l & \text{on } \Gamma \\ \sigma\frac{\partial u}{\partial e} = 0 & \text{on } \tilde{\Gamma} \end{cases} \tag{11}$$

where $\Gamma(\tilde{\Gamma})$ is the boundary $\partial\Omega$ with (without) electrodes, e is the outer normal direction at the boundary, V_l is the voltage measured by the lth electrode E_l when current I_l is applied, and z_l is the contact impedance.

Numerically, the object domain Ω discretized into n_S subdomains $\{\tau_j\}_{j=1}^{n_S}$ and σ is constant over each of them. One injects a current at a fixed frequency through a pair of electrodes attached to the boundary and measures the voltage differences on the remaining electrode pairs. This process is repeated over all electrodes, and the resulting data is represented as a vector denoted by $y \in \mathbb{R}^{n_Y}$ where n_Y is the number of measurements. We can define a mapping $F : \mathbb{R}^{n_S} \to \mathbb{R}^{n_Y}$ representing the discrete version of the forward operator:

$$y = F(\sigma) + \eta, \tag{12}$$

where η is a zero-mean Gaussian-distributed measurement noise. The EIT problem aims to estimate the static conductivity σ from measurements y. The EIT problem is widely considered to be challenging due to its severe ill-posedness, largely caused by the highly non-linear dependence of the boundary currents on the conductivity.

4.3.2. Training and testing datasets. We run numerical tests on a set of synthetic 2D experiments to evaluate the performance of the reconstruction methods. In the circular boundary ring, $L = 16$ electrodes are equally spaced and located. The conductivity of the background liquid is set to be $\sigma_0 = 1.0\Omega\mathrm{m}^{-1}$. Measurements are simulated by using `pyEIT` [22], a Python-based package for EIT. For each simulated conductivity phantom, the forward EIT problem (11) is solved using FEM with approximately 1342 triangular elements. We explore the following two typical cases to test the methods:

Table 3. The average MSE values of the DuNets methods for Case 1, with the associated standard deviations in parentheses. The average MSE for the GN approach is $10.8\ \times 10^{-3}$ ($\pm 3.16\ \times 10^{-3}$), and for the PDIPM-TV method, it is $5.24\ \times 10^{-3}$ ($\pm 2.78\ \times 10^{-3}$). The best MSE results are indicated in orange color.

Data size	50	200	400
LPGD	—	—	—
LPGD-MA	6.13×10^{-3} ($\pm 16.1 \times 10^{-4}$)	5.17×10^{-3} ($\pm 14.1 \times 10^{-4}$)	4.18×10^{-3} ($\pm 10.5 \times 10^{-4}$)
LPGD-RMA	3.02×10^{-3} ($\pm 1.34 \times 10^{-4}$)	2.48×10^{-3} ($\pm 1.08 \times 10^{-4}$)	2.25×10^{-3} ($\pm 1.41 \times 10^{-4}$)
LPGDSW	4.33×10^{-3} ($\pm 13.7 \times 10^{-4}$)	2.87×10^{-3} ($\pm 1.15 \times 10^{-4}$)	3.07×10^{-3} ($\pm 1.86 \times 10^{-4}$)
LPGDSW-MA	3.81×10^{-3} ($\pm 3.15 \times 10^{-4}$)	2.92×10^{-3} ($\pm 1.40 \times 10^{-4}$)	2.95×10^{-3} ($\pm 1.55 \times 10^{-4}$)
LPGDSW-RMA	3.92×10^{-3} ($\pm 4.79 \times 10^{-4}$)	2.65×10^{-3} ($\pm 1.99 \times 10^{-4}$)	2.63×10^{-3} ($\pm 1.14 \times 10^{-4}$)
LPD	3.25×10^{-3} ($\pm 1.87 \times 10^{-4}$)	2.55×10^{-3} ($\pm 1.34 \times 10^{-4}$)	2.35×10^{-3} ($\pm 2.13 \times 10^{-4}$)
LPD-MA	3.29×10^{-3} ($\pm 1.09 \times 10^{-4}$)	2.71×10^{-3} ($\pm 1.46 \times 10^{-4}$)	2.44×10^{-3} ($\pm 1.64 \times 10^{-4}$)
LPD-RMA	3.11×10^{-3} ($\pm 1.52 \times 10^{-4}$)	2.17×10^{-3} ($\pm 1.36 \times 10^{-4}$)	2.04×10^{-3} ($\pm 1.89 \times 10^{-4}$)

Table 4. The average MSE values of the DuNets methods for Case 2, with the associated standard deviations in parentheses. The average MSE for the GN approach is $12.6\ \times 10^{-3}$ ($\pm 1.0\ \times 10^{-3}$), and for the PDIPM-TV method, it is $7.67\ \times 10^{-3}$ ($\pm 0.36\ \times 10^{-3}$). The best results are indicated in orange color.

Data size	50	200	400
LPGD	—	—	—
LPGD-MA	10.3×10^{-3} ($\pm 1.54 \times 10^{-4}$)	8.60×10^{-3} ($\pm 1.45 \times 10^{-4}$)	7.87×10^{-3} ($\pm 1.15 \times 10^{-4}$)
LPGD-RMA	7.87×10^{-3} ($\pm 1.46 \times 10^{-4}$)	7.43×10^{-3} ($\pm 1.27 \times 10^{-4}$)	6.66×10^{-3} ($\pm 1.28 \times 10^{-4}$)
LPGDSW	7.81×10^{-3} ($\pm 1.07 \times 10^{-4}$)	6.88×10^{-3} ($\pm 2.13 \times 10^{-4}$)	6.69×10^{-3} ($\pm 2.39 \times 10^{-4}$)
LPGDSW-MA	6.96×10^{-3} ($\pm 5.04 \times 10^{-4}$)	5.46×10^{-3} ($\pm 2.01 \times 10^{-4}$)	5.53×10^{-3} ($\pm 2.26 \times 10^{-4}$)
LPGDSW-RMA	6.82×10^{-3} ($\pm 3.42 \times 10^{-4}$)	5.16×10^{-3} ($\pm 2.26 \times 10^{-4}$)	5.02×10^{-3} ($\pm 2.15 \times 10^{-4}$)
LPD	6.66×10^{-3} ($\pm 2.72 \times 10^{-4}$)	5.46×10^{-3} ($\pm 1.90 \times 10^{-4}$)	5.10×10^{-3} ($\pm 1.71 \times 10^{-4}$)
LPD-MA	6.32×10^{-3} ($\pm 1.95 \times 10^{-4}$)	5.49×10^{-3} ($\pm 1.63 \times 10^{-4}$)	5.18×10^{-3} ($\pm 1.86 \times 10^{-4}$)
LPD-RMA	6.01×10^{-3} ($\pm 1.60 \times 10^{-4}$)	5.13×10^{-3} ($\pm 1.91 \times 10^{-4}$)	4.93×10^{-3} ($\pm 1.49 \times 10^{-4}$)

- Case 1: the anomalies consist of two random circles with radii generated from the uniform distribution $U(-0.6, 0.6)$ and the conductivity values are 0.5 and 2 respectively in each circle;
- Case 2: the anomalies consist of four random circles with radii generated according to $U(-0.55, 0.55)$ and the conductivity values are 0.3, 0.5, 1.5, and 2.0.

We perform the DuNet methods with three different training sample sizes 50, 200 and 400, and 20 testing samples to evaluate the performance of the methods. We report that the testing time is approximately 19 seconds per sample for all DuNets.

4.3.3. Results and discussion. In the numerical experiments, we use the same set of unrolling schemes as in section 4.2, and in addition we also implement the regularized Gauss–Newton (GN) method and the primal-dual interior point method with total variation regularizer (PDIPM-TV) [40]. The parameters in both GN and PDIPM-TV are optimally tuned. We calculate the mean and standard deviation of MSE over ten independent runs with different training data sizes, and provide the results for Case 1 in table 3 and those for Case 2 in table 4. In what follows our discussion is focused on three aspects of the experimental results, (i) benefit

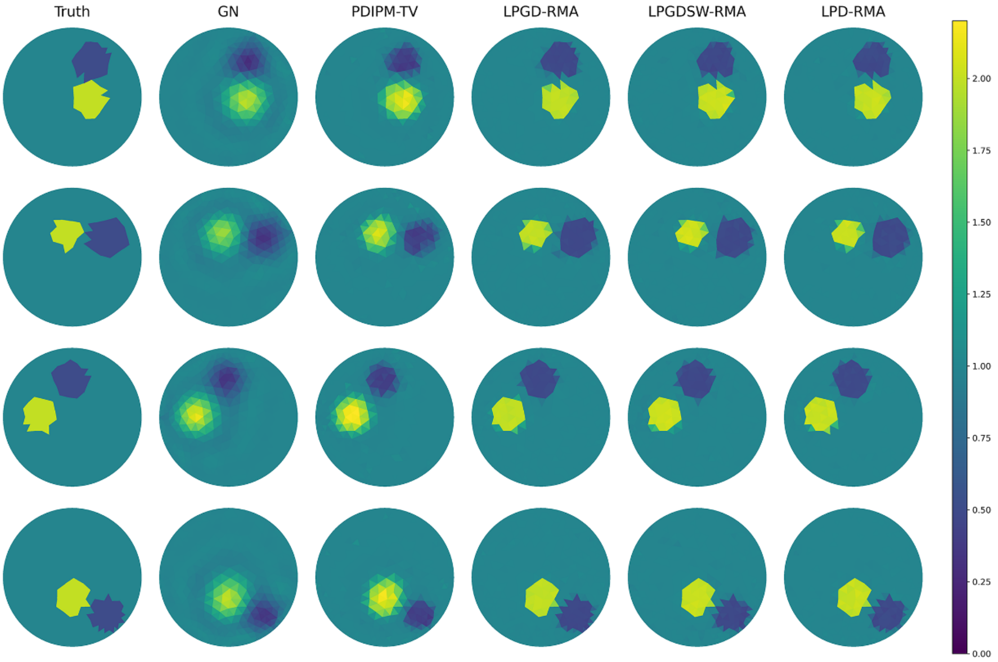


Figure 4. EIT reconstruction results of four testing samples in Case 1 (2 inclusions). From left to right: ground truth, GN, PDIPM-TV, LPGD-RMA, LPGDSW-RMA, and LPD-RMA.

of the RMA module, (ii) behaviors in low-data regimes, (iii) robustness against the number of inclusions in the conductivity area, and (iv) performance against the number of unrolling iterations.

4.3.3.1. Benefit of RMA scheme. First we note that five of the ten runs of the baseline LPGD (with no momentum acceleration) fail to converge within 20 epochs in both cases, and as such, we omit the results of the method in all the tables and figures. All the other algorithms can reasonably capture the inclusions' shape and position in all the ten runs. We highlight that, according to tables 3 and 4 the methods with the RMA module achieve the best performance in all but one test (LPGDSW method with 50 samples for Case 1) where the standard MA has the best results. In contrast, the effect of the standard MA module is not consistent: for example it results in worse performance than the baseline approach (without momentum acceleration) in the LPD method for case 1. The learning-based RMA module provides a more effective implicit regularizer than standard DuNets by utilizing previous gradient information. As such, we can see that DuNets-RMA achieves improved and more stable performance relative to standard DuNets. Moreover, we want to compare the proposed methods with the conventional optimization based approach. To do so we provide the reconstruction results of four testing samples in figure 4 (for Case 1) and figure 5 (for Case 2). For simplicity we only provide the results of the DuNets with RMA, which is compared with those obtained by GN and PDIPM-TV methods. It can be seen from both figures that all DuNets approaches with the RMA module can yield rather accurate reconstruction for all the inclusions, and the quality of the images is clearly better than those of GN and PDIPM-TV (note that the parameters in GN and PDIPM-TV

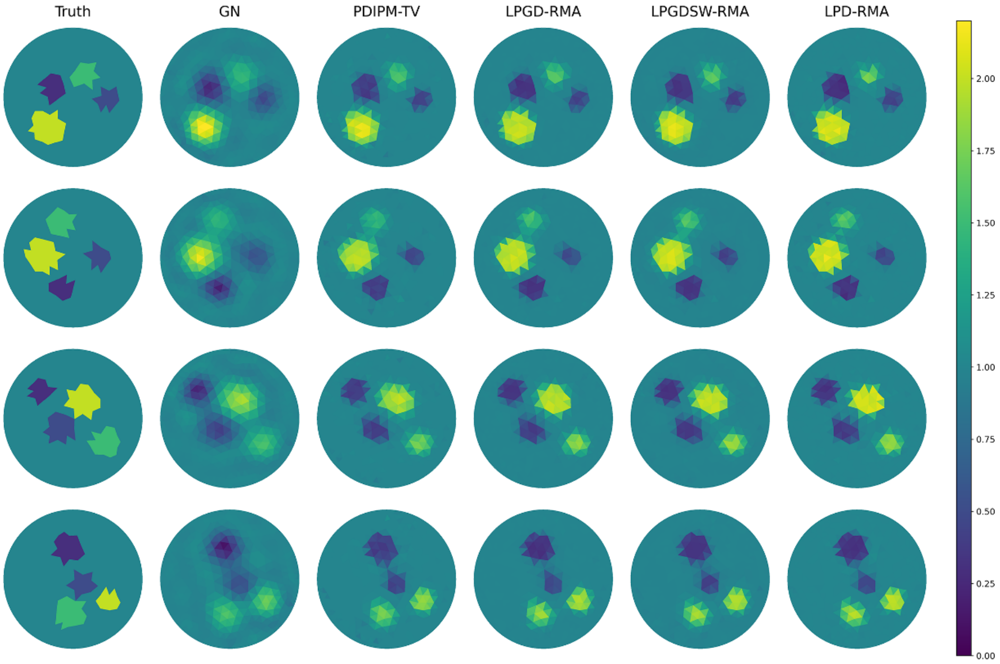
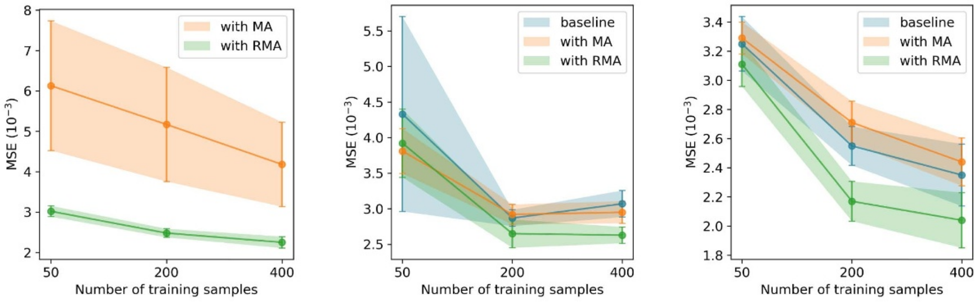


Figure 5. EIT reconstruction results of four testing samples in Case 2 (4 inclusions). From left to right: ground truth, GN, PDIPM-TV, LPGD-RMA, LPGDSW-RMA, and LPD-RMA.

are manually tuned for the best MSE results). Furthermore, the inclusions near the boundary are better recovered than those near the center, which confirms that inclusions far away from the boundary are more difficult to reconstruct since the boundary data are not sensitive to them [14, 35].

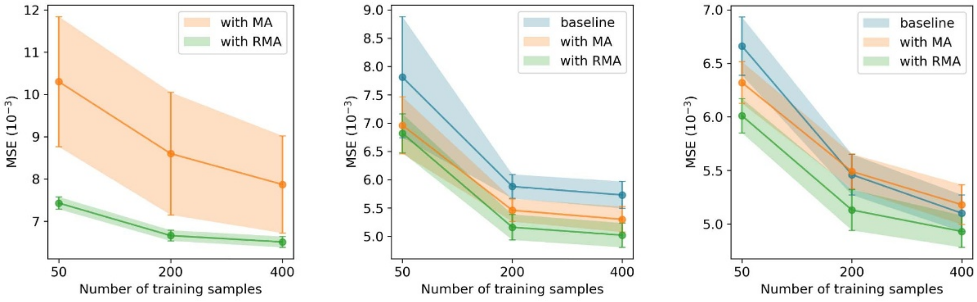
4.3.3.2. Data size. Next we examine the performance of the DuNets methods with respect to the data size. For this purpose we visualize the results in tables 3 and 4 with figures 6 and 7. One observes that MSE is decreasing with more training data used in all the methods with RMA, which is not the case for the baseline methods and those with MA. For example, in Case 1, the MSE results of LPGDSW become evidently higher when the data size changes from 200 to 400. These results demonstrate that the RMA module can increase the stability of the DuNet methods with respect the data size.

4.3.3.3. Number of inclusions. We now study how the methods perform with different numbers of inclusions by comparing the results in Case 1 (2 inclusions) and Case 2 (4 inclusions). We can see that the MSE results in Case 2 are generally higher than those in Case 1, indicating that more inclusions make the problems more challenging for all methods. Nevertheless, the RMA module considerably improves the performance of DuNets in both cases, an evidence that RMA is rather robust against the number of inclusions. In Case 1, the PDIPM-TV method yields a notably lower average MSE than that of LPGM-MA with 50 training datasets. Case 2 results show PDIPM-TV outperforming LPGM-MA, LPGDSW with 50 datasets, and



(a) LPGD (b) LPGDSW (c) LPD

Figure 6. The MSE results plotted against the number of training data for Case 1. The solid line represents the average of 10 tests, and the shade around the solid line depicts the one standard deviation.



(a) LPGD (b) LPGDSW (c) LPD

Figure 7. The MSE results plotted against the number of training data for Case 2. The solid line represents the average of 10 tests, and the shade around the solid line depicts the one standard deviation.

LPGD-RMA with 200 datasets, thus affirming the benefits of sparsity regularization on model performance.

4.3.3.4. Number of unrolled iterations. Finally, we evaluate the impact of the unrolled iteration number T on the performance of the LPD-RMA model. We train a set of LPD-RMA models with varying T values ($T = 6, 8, \ldots, 16$) using 200 training datasets in Case 1. Then we compute the average MSE value from ten independent runs over the 20 test datasets and present the results in figure 8. We can see from the figure that, when T is less than 10, an increase in T significantly enhances model performance in terms of the MSE value; however, when T is greater than 10, this enhancement diminishes and the MSE value slightly rises. One possible reason for this phenomenon is that an increase in T leads to a higher number of training parameters, making the model more prone to overfitting. Therefore, we should select an appropriate T that represents a good balance between model performance and complexity.

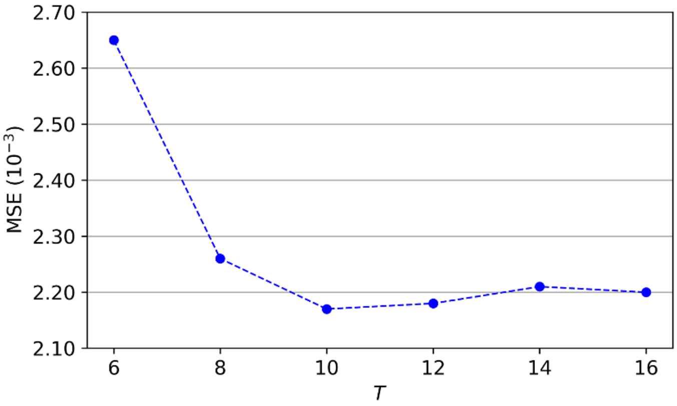


Figure 8. The MSE results plotted against the number of unrolled iterations.

5. Conclusion

In this work, we propose a method for improving the performance of DuNets in nonlinear inverse problems. In particular, we apply RMA to two popular DuNets: LPGD and LPD respectively. We provide numerical experiments that can demonstrate the performance of the proposed method. We expect that the proposed method can be extended to other unrolling algorithms and applied to a wide range of nonlinear inverse problems.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/zhouqp631/DuNets-RMA.

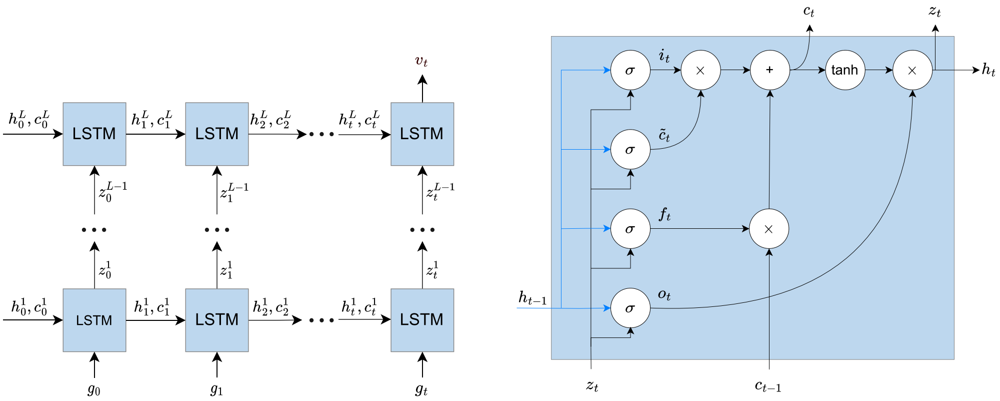
Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant 12101614, and the Natural Science Foundation of Hunan Province, China, under Grant 2021JJ40715. We are grateful to the High Performance Computing Center of Central South University for assistance with the computations.

Appendix. The LSTM network

Here we will provide a detailed description of the LSTM network used in RMA. Recall that in RMA, at each time t, a network model $(v_t, h_t, c_t) = \Xi_\vartheta\left(g_{t-1}, h_{t-1}, c_{t-1}\right)$ is used, and the structure of this network is specified as follows:

- The model $\Xi_\vartheta(\cdot)$ is a L-layer network with a LSTM-cell (denoted as $\mathrm{LSTM}^l(\cdot)$ for $l = 1, \ldots, L$) at each layer.
- Both the cell state c_t and the hidden state h_t have L components: $c_t = (c_t^1, \ldots, c_t^L)$ and $h_t = (h_t^1, \ldots, h_t^L)$ with each component being a vector of a prescribed dimension, and the initial states h_0 and c_0 are set to be zero.



(a) RMA structure (b) LSTM cell

Figure A1. (a) RMA structure uses a deep LSTM-RNN consisting of L hidden layers. (b) LSTM structure: cell gates are the input gate i$_t$, forget gate f_t, output gate o_t, and a candidate cell state $\tilde{c}_t$. In practice, the current output z_t is considered equal to the current hidden state h_t.

- For the lth layer, the inputs of the LSTM cell are g_t, c_{t-1}^1 and h_{t-1}^1, and the output of it are h_t^1 and an intermediate state z_t^1 that will be inputted into the next layer:

$$\left(z_t^l, h_t^l, c_t^l\right) = \mathrm{LSTM}^l\left(z_t^{l-1}, h_{t-1}^l, c_{t-1}^l\right),$$

where z_t^0 is initialized as g_t for the first layer $l = 1$. In the final layer $l = L$, the output h_t^L is set as the velocity v_t.

The structure of the LSTM network is summarized in figure A1(a).

We now discuss the details of the LSTM cell that combines the input features g_t at each time step and the inherited information from previous time steps. In what follows we often omit the layer index l when not causing ambiguity. At each layer, the LSTM cell proceeds as follows. First LSTM computes a candidate cell state $\tilde{c}_t$ by combining h_{t-1}^l and z_t^{l-1} (with $z_t^1 = g_{t-1}$), as:

$$\tilde{c}_t = \tanh\left(W_{hc}h_{t-1}^l + W_{gc}z_t^{l-1} + b_c\right),$$

and it then generates a forget gate x_t, an input gate i_t, and an output gate o_t via the sigmoid function $\sigma(\cdot)$:

$$\begin{aligned} f_t &= \sigma\left(W_{hx}h_{t-1}^l + W_{gx}z_t^{l-1} + b_x\right), \\ i_t &= \sigma\left(W_{hi}h_{t-1}^l + W_{gi}z_t^{l-1} + b_i\right), \\ o_t &= \sigma\left(W_{ho}h_{t-1}^l + W_{go}z_t^{l-1} + b_o\right). \end{aligned}$$

The forget gate is used to filter the information inherited from c_{t-1}, and the input gate is used to filter the candidate cell state at t. Then we compute the cell state c_t via,

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tilde{c}_t,$$

which serves as a memory reserving information from the previous iterations, and the hidden representation h_t^l as,

$$h_t = o_t \otimes \tanh\left(c_t\right),$$

where $\otimes$ denotes the element-wise product. Finally the output of the LSTM model v_t at time t is calculated as:

$$v_t = W_{hg}h_t + b_g,$$

which is used to replace the gradient in the standard DuNets methods. This is a brief introduction to LSTM tailored for our own purposes, and readers who are interested in more details of the method may consult [12, 16].

ORCID iD

Qingping Zhou https://orcid.org/0000-0003-1530-3525

References

[1] Adler J and Öktem O 2017 Solving ill-posed inverse problems using iterative deep neural networks *Inverse Problems* **33** 124007

[2] Adler J and Öktem O 2018 Learned primal-dual reconstruction *IEEE Trans. Med. Imaging* **37** 1322–32

[3] Baguer D O, Leuschner J and Schmidt M 2020 Computed tomography reconstruction using deep image prior and learned reconstruction methods *Inverse Problems* **36** 094004

[4] Benning M and Burger M 2018 Modern regularization methods for inverse problems *Acta Numer.* **27** 1–111

[5] Boyd S *et al* 2011 Distributed optimization and statistical learning via the alternating direction method of multipliers *Found. Trends Mach. Learn.* **3** 1–122

[6] Chambolle A and Pock T 2011 A first-order primal-dual algorithm for convex problems with applications to imaging *J. Math. Imaging Vis.* **40** 120–45

[7] Cherkaoui H, Sulam J and Moreau T 2020 Learning to solve TV regularised problems with unrolled algorithms *Advances in Neural Information Processing Systems* vol 33 pp 11513–24

[8] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H and Bengio Y 2014 Learning phrase representations using RNN encoder-decoder for statistical machine translation (arXiv:1406.1078)

[9] Chung J, Gulcehre C, Cho K and Bengio Y 2014 Empirical evaluation of gated recurrent neural networks on sequence modeling *NIPS 2014 Workshop on Deep Learning*

[10] Colibazzi F, Lazzaro D, Morigi S and Samoré A 2022 Learning nonlinear electrical impedance tomography *J. Sci. Comput.* **90** 1–23

[11] Combettes P L and Pesquet J-C 2011 Proximal splitting methods in signal processing *Fixed-Point Algorithms for Inverse Problems in Science and Engineering* (Springer) pp 185–212

[12] Greff K, Srivastava R K, Koutník J, Steunebrink B R and Schmidhuber J 2016 LSTM: a search space odyssey *IEEE Trans. Neural Netw. Learn. Syst.* **28** 2222–32

[13] Gregor K and LeCun Y 2010 Learning fast approximations of sparse coding *Proc. 27th Int. Conf. on Machine Learning* pp 399–406

[14] Guo R and Jiang J 2021 Construct deep neural networks based on direct sampling methods for solving electrical impedance tomography *SIAM J. Sci. Comput.* **43** B678–711

[15] Gupta H, Hwan Jin K, Nguyen H Q, McCann M T and Unser M 2018 CNN-based projected gradient descent for consistent CT image reconstruction *IEEE Trans. Med. Imaging* **37** 1440–53

[16] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80

[17] Hosseini S A H, Yaman B, Moeller S, Hong M and Akçakaya M 2020 Dense recurrent neural networks for accelerated MRI: history-cognizant unrolling of optimization algorithms *IEEE J. Sel. Top. Signal Process.* **14** 1280–91

[18] Houdard A, Bouveyron C and Delon J 2018 High-dimensional mixture models for unsupervised image denoising (HDMI) *SIAM J. Imaging Sci.* **11** 2815–46

[19] Kaipio J and Somersalo E 2006 *Statistical and Computational Inverse Problems* vol 160 (Springer)

[20] Kingma D P and Ba J 2015 Adam: a method for stochastic optimization *Int. Conf. on Learning Representations*, ed Y Bengio and Y LeCun

[21] Kumar R, Banerjee A and Vemuri B C 2009 Volterrafaces: discriminant analysis using volterra kernels *2009 IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 150–5

[22] Liu B, Yang B, Xu C, Xia J, Dai M, Ji Z, You F, Dong X, Shi X and Fu F 2018 pyEIT: a Python based framework for electrical impedance tomography *SoftwareX* **7** 304–8

[23] Liu Y, Gao Y and Yin W 2020 An improved analysis of stochastic gradient descent with momentum *Advances in Neural Information Processing Systems* vol 33 pp 18261–71

[24] Lohit S, Liu D, Mansour H and Boufounos P T 2019 Unrolled projected gradient descent for multi-spectral image fusion *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE) pp 7725–9

[25] Mardani M, Sun Q, Donoho D, Papyan V, Monajemi H, Vasanawala S and Pauly J 2018 Neural proximal gradient descent for compressive imaging *Advances in Neural Information Processing Systems* vol 31

[26] Monga V, Yuelong Li and Eldar Y C 2021 Algorithm unrolling: interpretable, efficient deep learning for signal and image processing *IEEE Signal Process. Mag.* **38** 18–44

[27] Nesterov Y E 1983 A method for solving the convex programming problem with convergence rate $o(1/k^2)$ *Dokl. Akad. Nauk SSSR* **269** 543–7

[28] Seo J K, Cheol Kim K, Jargal A, Lee K and Harrach B 2019 A learning-based method for solving ill-posed nonlinear inverse problems: a simulation study of lung EIT *SIAM J. Imaging Sci.* **12** 1275–95

[29] Shechtman Y, Eldar Y C, Cohen O, Nicholas Chapman H, Miao J and Segev M 2015 Phase retrieval with application to optical imaging: a contemporary overview *IEEE Signal Process. Mag.* **32** 87–109

[30] Sung Y, Choi W, Fang-Yen C, Badizadegan K, Dasari R R and Feld M S 2009 Optical diffraction tomography for high resolution live cell imaging *Opt. Express* **17** 266–77

[31] Sutskever I, Martens J, Dahl G and Hinton G 2013 On the importance of initialization and momentum in deep learning *Int. Conf. on Machine Learning* (PMLR) pp 1139–47

[32] Tang J Mukherjee S and Schönlieb C-B 2022 Accelerating deep unrolling networks via dimensionality reduction (arXiv:2208.14784)

[33] Wang H, Guixian X and Zhou Q 2024 A comparative study of variational autoencoders, normalizing flows and score-based diffusion models for electrical impedance tomography *J. Inverse Ill-Posed Problems* (https://doi.org/10.1515/jiip-2023-0037)

[34] Xu G, Wang H and Zhou Q 2024 Enhancing electrical impedance tomography reconstruction using learned half-quadratic splitting networks with Anderson acceleration *J. Sci. Comput.* **98** 49

[35] Yang Q, Sadeghi A, Wang G, Giannakis G B and Sun J 2020 Gauss-Newton unrolled neural networks and data-driven priors for regularized PSSE with robustness (arXiv:2003.01667)

[36] Yang Y, Tao R, Wei K and Ying F 2022 Dynamic proximal unrolling network for compressive imaging *Neurocomputing* **510** 203–17

[37] Zhang J and Ghanem B 2018 ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 1828–37

[38] Zhang J, He T, Sra S and Jadbabaie A 2020 Why gradient clipping accelerates training: a theoretical justification for adaptivity *Int. Conf. on Learning Representations*

[39] Zhang X, Liu J, Lu Y and Dong B 2019 Dynamically unfolding recurrent restorer: a moving endpoint control method for image restoration *7th Int. Conf. on Learning Representations, ICLR 2019*

[40] Zhou Z, dos Santos G S, Dowrick T, Avery J, Sun Z, Hui X and Holder D S 2015 Comparison of total variation algorithms for electrical impedance tomography *Physiol. Meas.* **36** 1193

[41] Zou Y and Guo Z 2003 A review of electrical impedance techniques for breast cancer detection *Med. Eng. Phys.* **25** 79–90

[42] Zoumpourlis G, Doumanoglou A, Vretos N and Daras P 2017 Non-linear convolution filters for CNN-based learning *Proc. IEEE Int. Conf. on Computer Vision* pp 4761–9