Metalearning-Based Alternating Minimization Algorithm for Nonconvex Optimization

Jing-Yuan Xia, Shengxi Li, *Member, IEEE*, Jun-Jie Huang, *Member, IEEE*, Zhixiong Yang, Imad M. Jaimoukha, and Deniz Gündüz, *Fellow, IEEE*

***Abstract*— In this article, we propose a novel solution for nonconvex problems of multiple variables, especially for those typically solved by an alternating minimization (AM) strategy that splits the original optimization problem into a set of subproblems corresponding to each variable and then iteratively optimizes each subproblem using a fixed updating rule. However, due to the intrinsic nonconvexity of the original optimization problem, the optimization can be trapped into a spurious local minimum even when each subproblem can be optimally solved at each iteration. Meanwhile, learning-based approaches, such as deep unfolding algorithms, have gained popularity for nonconvex optimization; however, they are highly limited by the availability of labeled data and insufficient explainability. To tackle these issues, we propose a meta-learning based alternating minimization (MLAM) method that aims to minimize a part of the global losses over iterations instead of carrying minimization on each subproblem, and it tends to learn an adaptive strategy to replace the handcrafted counterpart resulting in advance on superior performance. The proposed MLAM maintains the original algorithmic principle, providing certain interpretability. We evaluate the proposed method on two representative problems, namely, bilinear inverse problem: matrix completion and nonlinear problem: Gaussian mixture models. The experimental results validate the proposed approach outperforms AM-based methods.**

***Index Terms*— Alternating minimization (AM), deep unfolding, gaussian mixture model (GMM), matrix completion, meta-learning (ML).**

I. Introduction

ITERATIVE minimization is one of the most widely used approaches in signal processing, machine learning, and computer science. Typically, when dealing with multiple variables, these methods follow an alternating minimization (AM)-based strategy that converts the original problem of multiple variables into an iterative minimization of a sequence of subproblems corresponding to each variable, while the rest of the variables is held fixed. However, due to the nonconvexity of the problem, the obtained solutions do not necessarily converge to a global optimum, even when all the subproblems are solved optimally at each iteration (see [1] for example). The major issue underlying the failure of AM when facing nonconvexity is that a greedy and nonadaptive optimization rule is carried out when solving each subproblem throughout the iterations. Therefore, it lacks sufficient adaptiveness and effectiveness in terms of handling local optimums.

Recent advances in deep learning have highlighted its success in obtaining promising results for nonconvex optimization problems [2]–[5]. However, generic deep learning methods have limited generalization ability especially when test data are significantly different from the training data. This problem becomes more crucial in the ill-posed nonconvex optimization tasks. The weak explainability of the deep neural network behavior also questions its applicability in certain scenarios.

Deep unfolding [6] as an alternative learning-based approach has achieved significant success and popularity in solving various optimization problems. It improves model explainability by mapping a model-based iterative algorithm to a specific neural network architecture with learnable parameters. In this way, the mathematical principles of the original algorithm are maintained, thus leading to better generalization behavior [2], [7]–[12]. We note that most of the deep unfolding algorithms are designed for solving linear inverse problems and require supervised learning. There are also deep unfolding algorithms for solving nonconvex optimization problems. Luo *et al.* [9] propose to unfold alternating optimization for blind image super-resolution, which is typically ill-posed and nonconvex. In [10], an unfolded weighted minimum mean square error (WMMSE) algorithm is proposed to estimate the parameter of the gradient descent step size for solving the multiple input single output (MISO) beamforming problem, which is highly nonlinear and nonconvex. We can see from [9], [10], when solving nonconvex problems, deep unfolding algorithms either only retain the iterative framework and replace all components with deep networks [9] or learn a minimum number of parameters but result in less effective performance [10]. There exists a tradeoff between achieving better performance with highly overparameterized deep networks and retaining model explainability and generalization ability

Manuscript received October 20, 2021; revised February 12, 2022; accepted April 1, 2022. This work was supported by the National Natural Science Foundation of China under Project 61921001 and Project 62022091. *(Corresponding author: Jun-Jie Huang.)*

Jing-Yuan Xia, Jun-Jie Huang, and Zhixiong Yang are with the College of Electronic Engineering, College of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: j.xia16@imperial.ac.uk; j.huang15@imperial.ac.uk).

Shengxi Li is with the College of Electronic Engineering, Beihang University, Beijing 100191, China (e-mail: shengxi.li17@imperial.ac.uk).

Imad M. Jaimoukha is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: i.jaimouka@imperial.ac.uk).

Deniz Gündüz is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K., and also with the Department of Engineering Enzo Ferrari, University of Modena and Reggio Emilia (UNIMORE), 41121 Modena, Italy (e-mail: d.gunduz@imperial.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3165627.

Digital Object Identifier 10.1109/TNNLS.2022.3165627

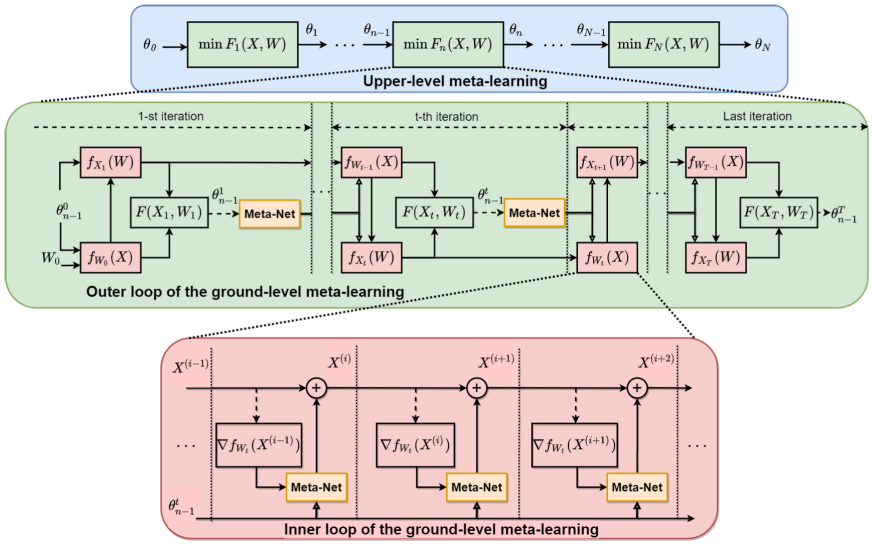


Fig. 1. Overall structure of the MLAM algorithm. The parameters of the applied metanetwork are denoted by θ. The upper level metalearning is implemented on a set of nonconvex problems $\{F_n(\boldsymbol{X}, \boldsymbol{W})\}_{n=1}^{N}$, as shown in the upper row. θ is continuously updated across different problems. The ground-level metalearning is applied to solve each nonconvex problem $F_n(\boldsymbol{X}, \boldsymbol{W})$ through an alternative outer loop between two inner loops corresponding to two variables, respectively, presented in the middle row. $\boldsymbol{\theta}$ is dynamically updated with respect to the global loss $F(\boldsymbol{X}, \boldsymbol{W})$. The inner loop is depicted in the bottom row, where the variable is iteratively updated by the metanetwork. $\boldsymbol{\theta}$ is frozen at inner loop iterations. The dashed lines denote gradient operator occurs, and the solid line represents information flow along the edge.

with minimum learnable parameters. This tradeoff is also highly related to the amount of required labeled data and the request for prior knowledge. Therefore, it is essential to design a new approach that enables us to carry a learning-based neural network model with interpretable optimization-inspired behavior in an unsupervised learning way.

Metalearning has witnessed increasing importance in terms of strong adaptation ability in solving new tasks [13]–[19]. Unlike the standard supervised learning solutions, metalearning does not focus on solving a specific task at hand but aims to learn domain-general knowledge in order to generate an adaptive solution for a series of new tasks. Typically, a metanetwork collects domain-specific knowledge when solving each specific task and then extracts domain-general knowledge across solving different tasks. Most of the popular metalearning algorithms, such as model-agnostic metalearning (MAML) [20], metric-based metalearning [21], and learning-to-learn [22], share a hierarchical optimization structure that is composed of *inner* and *outer* procedures: the metanetwork performs as an optimizer to solve specific tasks at inner procedure, and the parameters of metanetwork are then updated through the outer procedure. We note that the AM-based iterative method can also be regarded as sharing this bilevel optimization framework. However, the optimization strategy in AM-based algorithms is commonly frozen, and the inner optimization behavior is independent of the outer alternating procedure. Therefore, we are inspired to propose a metalearning based alternating minimization (MLAM) algorithm for solving nonconvex problems, which will enable inner optimization to be continuously updated with respect to the mutual knowledge extracted over outer steps.

The proposed MLAM algorithm is composed of two-level metalearning, namely, the *upper level* and *ground-level* metalearning. The upper level metalearning learns on a set of nonconvex problems and aims to enhance the adaptability toward new problems. In contrast, the ground-level metalearning learns on a sequence of subproblems within each nonconvex problem from the upper level and, therefore, aims to find an adaptive and versatile algorithm for the sequential subproblems. The overview structure of the proposed MLAM method is shown in Fig. 1.

Specifically, the upper level metalearning learns to leverage the optimization experiences on a series of problems, while the learned algorithm in the ground-level metalearning maintains the original inner-and-outer iterative structure and the algorithmic principles but replaces the frozen and handcrafted algorithmic rule with a dynamic and adaptive metalearned rule. In other words, it aims to learn an optimization strategy that is able to provide a "bird's-eye view" of the mutual knowledge extracted across outer loops for those subproblems being optimized in the inner loops. Therefore, the learned strategy does not optimize each subproblem locally and exhaustively through minimization; instead, it optimizes them by incorporating the global loss information with superior adaptability. Moreover, the proposed MLAM algorithm is able to solve optimization problems in an unsupervised manner. As a result, the proposed MLAM algorithm achieves better performance in nonconvex problems while requiring less (and even no) labeled data for training.

The main contributions of this article are mainly threefold.

1) The core contribution is the proposed MLAM approach for nonconvex optimization problems. In an unsupervised manner, MLAM achieves a less-greedy and adaptive optimization strategy to learn a nonmonotonic algorithm for solving nonconvex optimization problems.
2) The proposed MLAM takes a step further toward enhanced interpretability. The algorithmic principles of the original model-based iterative algorithm are

fully maintained without the need to replace iterative operations with black-box deep neural networks.

3) With extensive simulations, we have validated that the proposed MLAM algorithm achieves promising performances on the challenging problems of matrix completion and the Gaussian mixture model (GMM). It is able to effectively solve these extremely difficult nonconvex problems even when the traditional approaches fail.

The rest of this article is organized as follows. Section II gives the background and a brief review of previous approaches. Section III introduces our proposed MLAM approach and presents an long short-term memory (LSTM)-based MLAM method. Section IV illustrates two representative applications of our MLAM method. Section V provides simulation results. Section VI concludes this article.

II. Relevant Prior Work

In this section, we will first briefly introduce the general problem formulation for multivariable nonconvex optimizations and the solution approaches. Then, we will demonstrate our motivation for proposing MLAM and review the relevant metalearning approaches.

Nonconvex optimization problems that involve more than one variable are of great practical importance but are often difficult to be well accommodated. The underlying relationship between variables can be linear (e.g., product and convolution) or nonlinear (e.g., logarithmic operation and exponential kernel). For illustration convenience, we consider a general optimization formulation over an intersection of two variables in the matrix form, which can be expressed in the form of

$$\left(\hat{\boldsymbol{W}}, \hat{\boldsymbol{X}}\right) = \underset{(\boldsymbol{W},\boldsymbol{X})\in\mathcal{W}\times\mathcal{X}}{\arg\min} \; F(\boldsymbol{W}, \boldsymbol{X}) \tag{1}$$

where $F : \mathcal{W}\times\mathcal{X} \to \mathbb{R}$ is a nonconvex function that describes the mapping between the observations $\boldsymbol{Y} = F(\boldsymbol{W}, \boldsymbol{X})$ and two variables $\boldsymbol{W} \in \mathcal{W}$ and $\boldsymbol{X} \in \mathcal{X}$.

A. Model-Based Solutions

The model-based iterative algorithms [23]–[31] typically solve (1) by adopting an AM-based strategy. The basic idea is to sequentially optimize a subproblem corresponding to each variable while keeping the other variable fixed. That is, starting from an arbitrary initialization $\boldsymbol{W}_0 \in \mathcal{W}$, the AM-based algorithm sequentially solves two subproblems at the tth iteration via

$$\begin{aligned} \boldsymbol{X}_t &= \arg\min_{\boldsymbol{X}\in\mathcal{X}} f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}) \\ \boldsymbol{W}_t &= \arg\min_{\boldsymbol{W}\in\mathcal{W}} f_{\boldsymbol{X}_t}(\boldsymbol{W}) \end{aligned} \tag{2}$$

where $f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_t}(\boldsymbol{W})$ are the functions corresponding to $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively, while fixing the other one to the value obtained in the previous iterations, i.e., $f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}) = f(\boldsymbol{W}_{t-1}, \boldsymbol{X})$ and $f_{\boldsymbol{X}_t}(\boldsymbol{W}) = f(\boldsymbol{W}, \boldsymbol{X}_t)$. The solution of each subproblem in (2) could be attained by a gradient-descent-based iterative process

$$\boldsymbol{X}_i = \boldsymbol{X}_{i-1} + \phi\left(\{\boldsymbol{X}_k\}_{k=0}^{i-1}, \nabla f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}_{i-1})\right) \tag{3}$$

where $\{\boldsymbol{X}_k\}_{k=0}^{i-1}$ represent the historical values of the parameters for i optimization steps, $\nabla f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X}_{i-1})$ is the gradient of objective function on $\boldsymbol{X}_{i-1}$, and $\phi(\cdot)$ defines the variable updating rule of different algorithms. Algorithms such as iterative shrinkage-thresholding algorithm (ISTA) [7] and WMMSE [10] formulate a closed-form solution when iteratively solving each subproblem in (2), which is essentially equal to a first order stationary point obtained by gradient descent-based methods as well.

Before proceeding further, we first introduce some concepts that will be used throughout this article. We define the *overall problem* as the optimization problem with objective function $F(\boldsymbol{W}, \boldsymbol{X})$, which will be called the global loss function. We refer to the optimization problems over each of the variables as the *subproblem* and their objective functions $f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_t}(\boldsymbol{W})$ for $t \geq 1$ as local loss functions. We define the alternative iterations over the subproblems as the outer loop and the iterative iterations for solving each subproblem as the inner loop.

The AM-based algorithm attempts to solve the overall problem by sequentially minimizing the two subproblems $f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}_t}(\boldsymbol{W})$. However, the AM-based algorithm does not necessarily converge to a global optimal solution. This could be due to two main reasons: 1) AM-based methods optimize the local loss functions without fully utilizing the information from the global loss function and 2) AM-based methods usually solve the local loss function greedily using the first-order information, which may not necessarily lead to the best solution, that is, the global optima, in terms of the global loss function.

Addressing these two issues of the AM is nontrivial. The key difficulty is that the variable optimized rules of the model-based solutions are frozen during the iteration with respect to a certain update function, i.e., ϕ in (3). Typically, ϕ is designed for optimizing each subproblem greedily; as a result, it is not expected to reach the global optimal solutions for the overall problem.

B. Learning-Based Solutions

Different from the model-based approaches, the recent deep learning-based methods typically require training an overparameterized deep neural networks in an end-to-end learning fashion with a large labeled dataset [7], [32]–[36]. During testing, the trained deep neural network is fed by the observations and directly outputs the estimated variables. The performance of these methods is highly bounded by the training datasets; however, the ground-truth data are neither sufficient nor even exist in realistic nonconvex tasks, such as the GMM problems. Another shortcoming that is common to these learning-based methods is the weak explainability of the end-to-end deep neural network behavior.

Different from the generic deep learning approaches, deep unfolding algorithms try to combine model- and learning-based approaches. They have three major features: 1) they map the iterative optimization algorithm into a specific unfolded network architecture with trainable parameters; 2) each layer in the deep unfolding network corresponds to one iteration of the original iterative algorithm, while the number of layers,

especially the iterations, is frozen; and 3) similar to the other deep learning approaches, deep unfolding also requires pairwise labeled data for training.

Mathematically, deep unfolding approaches follow the iterative framework as in (2) but replace the analytical minimization algorithm (or specific operators, such as soft-thresholding or singular value thresholding) by neural networks in the form of

$$\begin{aligned} \boldsymbol{X}_t &= \mathsf{Layer}_t^X(\boldsymbol{W}_{t-1}) \\ \boldsymbol{W}_t &= \mathsf{Layer}_t^W(\boldsymbol{X}_t) \end{aligned} \tag{4}$$

where the number of iterations is fixed with $t = T$. Hence, the whole deep unfolded network is composed by T layers, where each layer is composed of several operators that reflect the mathematical behavior of the original iterative algorithm.

In [36], Zhang *et al.* propose a deep unfolded network for image super-resolution in which the solver for one variable is a generic deep network, while the other one keeps consistent with the model-based solver. In deep alternating network [9], two networks, referring to an estimator and a restorer, work as two solvers for the splitted subproblems. Therefore, the whole unfolding algorithm alternates between two network operators. In a different way, the unfolded network in [10] almost retains the original model-based iterative algorithm to solve a nonlinear problem but takes a network-based generator to learn the hyperparameters for gradient descent step size by unsupervised learning. However, its performance does not surpass the counterpart model-based algorithm.

In summary, deep unfolding has made a step further toward better explainability; however, these approaches still perform as an end-to-end network behavior and mostly only enable interpretable alternating structure while replacing the original optimization-based algorithm with deep neural networks. Consequently, its interpretability is limited when applied to ill-posed nonconvex problems, and it is based on a data-driven optimization strategy. Besides, most deep unfolding algorithms require a large number of labeled data for supervised learning. Hence, the performance of learning-based approaches in solving ill-posed nonconvex problems, especially those without sufficient labeled data, is still limited.

III. MLAM Approach

In this section, we introduce the proposed MLMA approach for solving optimization problems with multiple variables that are highly nonconvex and ill-posed. As aforementioned, in solving these problems, the existing model-based methods typically struggle, while, on the other hand, the lack of sufficient labeled training samples also restricts the performance of learning-based methods. Therefore, the key idea of our MLAM approach is to design a novel way that is not only capable of benefitting from both the optimization-based algorithmic principle and the superior performances by learning but also surpasses the AM-based strategy through metalearning.

A. Overall Structure of the MLAM Approach

The proposed MLAM approach consists of two levels of metalearning. The upper level metalearning operates on a set of overall problems, and the ground-level metalearning optimizes the sequential subproblems within an overall problem. Both upper level and ground-level operations contain a bilevel optimization structure that is composed of outer and inner procedures. The outer loop of the upper level metalearning continuously updates the parameters of the MetaNet $\boldsymbol{\theta}$ across different overall problems, and each inner loop equals one ground-level metalearning on solving an overall problem. For both the ground-level and the upper level metalearning processes, we denote the inner loop index at superscripts and the outer loops' index at subscripts. We will then introduce the details of MLAM in a bottom-up manner.

1) Upper Level Metalearning: The upper level metalearning is depicted in Fig. 1 (the boxes in blue). It aims to extract a general knowledge of updating rules across different overall problems.

Each overall problem is considered as a single task, and the whole learning process is accommodated on a set of tasks within the same dimension. Therefore, the learned algorithm is no longer designed for solving a single task but a set of tasks $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^N$. Although the proposed MLAM model follows the AM structure, it establishes a new bridge between the upper level and ground-level metalearning by replacing the frozen update functions with metanetworks. This allows for variables updated at the inner loops to be guided by global loss information from the outer loops, thus achieving a global scope optimization.

2) Ground-Level Metalearning: The general structure of the ground-level metalearning is shown in Fig. 1 (the boxes in green and red). A ground-level metalearning is performed by solving a specific overall problem $F_n(\boldsymbol{W}, \boldsymbol{X})$. This process contains an alternating optimization process on a sequence of subproblems (boxes in green), which is named the outer loop, and each subproblem is solved by an iterative gradient descent-based operation, which is defined as the inner loop (boxes in red).

In the inner loop, the variable (e.g., $\boldsymbol{X}$ and $\boldsymbol{W}$) to be optimized is updated based on the iterative gradient descent process using a metanetwork, and the parameter of the MetaNet is frozen. Taking the optimization on $f_{\boldsymbol{W}_{t-1}}(\boldsymbol{X})$ as an example, the ith step update on the variable $\boldsymbol{X}$ at the inner loop can be expressed in the following form:

$$\boldsymbol{X}^{(i)} = \boldsymbol{X}^{(i-1)} + \mathsf{MetaNet}\left(\nabla f_{\boldsymbol{W}_{t-1}}\left(\boldsymbol{X}^{(i-1)}\right)\right) \tag{5}$$

where $\mathsf{MetaNet}(\cdot)$ is a neural network with learnable parameter $\boldsymbol{\theta}$ and performs as an optimizer for variable update.

MetaNet replaces the handcrafted update function $\phi(\cdot)$ in (3) as a learnable and adaptive update function. Its parameter $\boldsymbol{\theta}$ is frozen at the inner loop and will be updated at the outer loop with respect to the global loss function. Specifically, different from the traditional AM-based algorithms, MLAM establishes an extra update cue for the parameters of MetaNet at the outer loops by minimizing the global loss $F(\boldsymbol{X}, \boldsymbol{W})$. This enables that the gradients of $f_{\boldsymbol{W}}$ and $f_{\boldsymbol{X}}$ on variables $\boldsymbol{X}$ and $\boldsymbol{W}$ and the gradient of $F(\boldsymbol{X}, \boldsymbol{W})$ on parameter $\boldsymbol{\theta}$ are integrated into one circulating system. In this way, the optimization behavior on each subproblem is no longer independent to the others, all of which interact through the MetaNet. In Fig. 1,

the dashed arrows at the outer loop of the ground-level metalearning indicate the backpropagation of global loss to update the parameters $\boldsymbol{\theta}$ based on gradient descent. Taking the tth iteration as an instance, the parameter update is expressed as

$$\boldsymbol{\theta}_{n-1}^{t} = \boldsymbol{\theta}_{n-1}^{t-1} + \alpha \cdot \Phi\left(\nabla_{\boldsymbol{\theta}_{n-1}^{t-1}} F(\boldsymbol{X}_1, \boldsymbol{W}_1)\right) \tag{6}$$

where α is the learning rate and $\Phi(\cdot)$ denotes the learning algorithms for network update (e.g., the Adam method [37]).

Because the optimization landscapes for different subproblems can be significantly different, MetaNet tends to extract a general knowledge on generating descent steps for variable updates across different subproblems. Consequently, MetaNet learns to provide a superior and adaptive estimation for a sequence of subproblems through this metalearning process.

In Fig. 1, the outer loop shows how the parameters at each iteration are updated. We define the update interval as the number of iterations for each parameter update, and in Fig. 1, the update interval is set to 1. Generally speaking, the smaller the update interval the higher chances are updated, and the larger the update interval the more experience could be learned. The update interval of the parameters could be tuned. We will show more details in Sections III-B and V-A.

Remark 1: The implementation of the two levels of metalearning is indispensable. As the proposed MLAM learns in an unsupervised way, the training process of ground-level metalearning could be also regarded as solving when training on one problem. Therefore, MLAM can be directly applied to a specific problem by only carrying on the ground-level metalearning [19]. In this article, the proposed approach works with both the two levels of the metalearning.

The metalearning implemented on the proposed MLAM approach mainly contributes to two aspects.

1) Recalling to the upper level metalearning, the training strategy of the MetaNet follows the metalearning strategy, thus improving the adaptation of the learned parameters on solving different nonconvex optimization problems $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^{N}$. Specifically, the metalearning behavior significantly enhances the capacity of solving different tasks by leveraging the experience of solving a series of $\{F_i(\boldsymbol{W}, \boldsymbol{X})\}_{i=1}^{N}$.
2) The optimization performance of solving each specific task is dramatically improved by achieving a dynamic and adaptive gradient-based updating rule through the implemented ground-level metalearning. The solution strategy of solving each task $F_i(\boldsymbol{W}, \boldsymbol{X})$ is metalearned over the alternating iterations. In this way, the solution strategy is leveraged by the metalearning behavior to allow the subproblems $\{f_{\boldsymbol{W}_t}(\boldsymbol{X}), f_{\boldsymbol{W}_t}(\boldsymbol{X})\}_{t=1}^{T}$ to be solved in a less greedy but more effective way. Essentially, as a benefit of the metalearning, the MLAM has the capacity of leveraging the experience of solving a series of subproblems $\{f_{\boldsymbol{W}_t}(\boldsymbol{X}), f_{\boldsymbol{W}_t}(\boldsymbol{X})\}_{t=1}^{T}$ to learn a gradient-based strategy that provides better convergence on solving the current task $F(\boldsymbol{W}, \boldsymbol{X})$.

B. MLAM With LSTM-Based MetaNet

In this section, we introduce the implementation details of using recurrent neural networks (RNNs) as the MetaNet in the proposed MLAM algorithm. Specifically, the LSTM network [38] is adopted as the variable update function at the ground level in the MLAM model. The proposed LSTM-MLAM updates the parameters of the MetaNet with respect to the accumulated global losses.

RNN has a sequentially processing chain structure to achieve the capacity of "memory" on sequential data. LSTM [38] is one of the most well-known RNNs and is able to memorize and forget different sequential information. Memory is the most important feature of LSTM (RNN). It stores the status information of previous iterations and allows for the information to flow along the entire chain process. In this way, LSTM can integrate previous information with the current step input. Mathematically, the output of LSTM at the ith iteration $\boldsymbol{H}^{(i)}$ is determined by the current gradient $\nabla f(\boldsymbol{x}_i)$ and the last cell state $\boldsymbol{C}^{(i)}$ in the following forms:

$$\begin{aligned} \boldsymbol{H}^{(i)} &= \mathsf{LSTM}\left(\nabla f(\boldsymbol{x}_i), \boldsymbol{C}^{(i)}, \boldsymbol{\theta}^{(i)}\right) \\ \boldsymbol{C}^{(i+1)} &= \boldsymbol{z}_f \odot \boldsymbol{C}^{(i)} + \boldsymbol{z}_i \odot \tilde{\boldsymbol{C}}^{(i-1)} \end{aligned} \tag{7}$$

where $\mathsf{LSTM}(\cdot, \cdot, \boldsymbol{\theta})$ denotes the LSTM network with parameters $\boldsymbol{\theta}$, $\odot$ denotes Hadamard Product, $\boldsymbol{z}_f$ and $\boldsymbol{z}_i$ are the vectors of intermediate conditions inside LSTM, and $\tilde{\boldsymbol{C}}^{(i-1)} = \nabla_{\boldsymbol{\theta}^{(i-1)}} \mathcal{L}^{(i)}$ is the candidate cell state, referring to the gradient of current loss $\mathcal{L}^{(i)}$ over the last parameters $\boldsymbol{\theta}^{(i-1)}$ in our problem.

We adopt two LSTM networks $\mathsf{LSTM}_{\boldsymbol{X}}$ and $\mathsf{LSTM}_{\boldsymbol{W}}$ as MetaNet to generate variable update functions, recalling $\phi(\cdot)$ in (3), for solving subproblems corresponding to variables $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively. We also denote $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ as the parameters of $\mathsf{LSTM}_{\boldsymbol{X}}$ and $\mathsf{LSTM}_{\boldsymbol{W}}$, and denote $\boldsymbol{C}_{\boldsymbol{X}}$ and $\boldsymbol{C}_{\boldsymbol{W}}$ as their cell states. The inputs of the LSTM are the gradient of local loss function and the sequential knowledge of variables, represented by cell state $\boldsymbol{C}$. Then, the LSTM outputs the variable update term that integrates step size and direction together. Denoting the inner loop update steps $i-1$ and $j-1$ at superscripts and outer loops steps $t-1$ and t at subscripts for each subproblem, the variables are updated in the following forms:

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{X}}^{(i-1)} &= \mathsf{LSTM}_{\boldsymbol{X}}\left(\nabla f_{\boldsymbol{W}_{t-1}}\left(\boldsymbol{X}^{(i-1)}\right),\ \boldsymbol{C}_{\boldsymbol{X}}^{(i-1)},\ \theta_{\boldsymbol{X}}\right) \\ \boldsymbol{X}^{(i)} &= \boldsymbol{X}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{X}}^{(i-1)} \end{aligned} \tag{8}$$

and

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{W}}^{(j-1)} &= \mathsf{LSTM}_{\boldsymbol{W}}(\nabla f_{\boldsymbol{X}_t}(\boldsymbol{W}^{(j-1)}),\ \boldsymbol{C}_{\boldsymbol{W}}^{(j-1)},\ \theta_{\boldsymbol{W}}) \\ \boldsymbol{W}^{(j)} &= \boldsymbol{W}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{W}}^{(j-1)}. \end{aligned} \tag{9}$$

As mentioned in Section III-A, at the inner loops, the parameters $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$ are frozen and are used to generate the update steps $\boldsymbol{H}_{\boldsymbol{X}}$ and $\boldsymbol{H}_{\boldsymbol{W}}$ for variables with frozen iteration numbers; therefore, the update strategy is essentially determined by the parameters $\boldsymbol{\theta}_{\boldsymbol{X}}$ and $\boldsymbol{\theta}_{\boldsymbol{W}}$. At the outer loops, we leverage the accumulated global losses to guide the parameter update for better optimization strategy through backpropagation. Let t_{out} denote the update interval; the

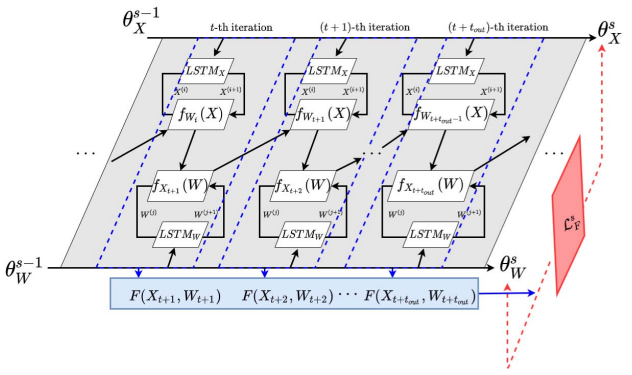


Fig. 2. Outer loop iterations on subproblems are placed on the gray plane. Each iteration contains two subproblems (dashed blue rhombus), and each subproblem optimizes the corresponding variable at inner loop along with an LSTM MetaNet. The global loss $F(\boldsymbol{X}, \boldsymbol{W})$ is computed after each iteration. For every t_{out} number of iterations, the accumulated global losses $\mathcal{L}_F^s$ (rhombus in red) is calculated and backpropagated to update the parameters $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_W$ based on gradient descent manners (referred by the red dashed arrows).

accumulated global loss is given by

$$\mathcal{L}_F^s = \frac{1}{t_{\text{out}}} \sum_{t_s=(s-1)t_{\text{out}}+1}^{\text{st}_{\text{out}}} \omega_{t_s} F\left(\boldsymbol{W}_{t_s}, \boldsymbol{X}_{t_s}\right) \tag{10}$$

where $\omega_t \in \mathbb{R}_{\geq 0}$ denotes the weight associated with each outer step, and $s = 1, 2, \ldots, S$, with $S = T/t_{\text{out}}$ being the maximum update number for LSTM networks and T being the maximum outer steps. For every t_{out} outer loop iteration, the accumulated global losses $\mathcal{L}_F^s$ are computed and are used to update $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_W$ as follows:

$$\begin{aligned} \boldsymbol{\theta}_X^{s+1} &= \boldsymbol{\theta}_X^s + \alpha_X \cdot \text{Adam}\left(\boldsymbol{\theta}_X^s, \nabla_{\boldsymbol{\theta}_X^s} \mathcal{L}_F^s\right) \\ \boldsymbol{\theta}_W^{s+1} &= \boldsymbol{\theta}_W^s + \alpha_W \cdot \text{Adam}\left(\boldsymbol{\theta}_W^s, \nabla_{\boldsymbol{\theta}_W^s} \mathcal{L}_F^s\right) \end{aligned} \tag{11}$$

where α_X and α_W denote the learning rates for the metanetworks LSTM_X and LSTM_W, respectively. The parameters of LSTMs are updated by the Adam method [37].

Remark 2: The formulation (10) can be extended to accommodate prior information as follows:

$$\mathcal{L}' = \mathcal{L}_F^s + \omega_w \left\| \boldsymbol{W}_{t_s} - \tilde{\boldsymbol{W}} \right\|_F^2 + \omega_x \left\| \boldsymbol{X}_{t_s} - \tilde{\boldsymbol{X}} \right\|_F^2 \tag{12}$$

where ω_w and ω_x denote the weights of the prior knowledge, and $\{\tilde{\boldsymbol{W}}\}$ and $\{\tilde{\boldsymbol{X}}\}$ are the available paired training samples from historical data.

Hence, $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_W$ successfully build a connection between the variable update functions and the global losses. At inner loops, $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_W$ convey an extra global loss information from the outer loops for the update functions. The accumulated global losses allow LSTMs to be updated with respect to the mutual knowledge of dealing with different subproblems. Therefore, in the ground-level metalearning, the learned algorithm has better adaptability to the new subproblem in the sequence. The accumulated global loss-based ground-level metalearning is depicted in Fig. 2.

There are two merits of adopting $\mathcal{L}_F^s$ to update the parameters. First, the update with a small update interval, e.g., $t_{\text{out}} = 1$, will lead to severe fluctuation. An appropriate update interval with accumulated global losses is able to effectively relieve the factor of the outliers in the training process. We shall present the related simulation results in Section V-A. Second, the leveraged global losses can provide more mutual knowledge than one global loss value. The parameters are updated with the objective to minimize a partial trajectory of the global losses. Therefore, it allows the MetaNet to learn a nonmonotonic solution, where the global loss could increase at the beginning iterations but quickly decrease to better optimums on the global scope. We will discuss in detail with a practical example in Section IV-B.

Algorithm 1 General Structure of MLAM Algorithm for Solving One Problem

Input: global loss function $F(\boldsymbol{W}, \boldsymbol{X})$, local loss functions $f_{\boldsymbol{W}}(\boldsymbol{X})$ and $f_{\boldsymbol{X}}(\boldsymbol{W})$, random initialization $\boldsymbol{W}_0$, number of outer loops T, and number of inner loops I and J.
Output: Estimated variables $\boldsymbol{W}_T$, $\boldsymbol{X}_T$.
for $t \leftarrow 1, \ldots, T$ **do**
  for $i \leftarrow 0, \ldots, I-1$ **do**
    $\Delta \boldsymbol{X} = \text{LSTM}_X(\nabla f_{\boldsymbol{W}_t}(\boldsymbol{X}^{(i)}), \boldsymbol{C}_X^{(i)}, \boldsymbol{\theta}_X^s)$
    $\boldsymbol{X}^{(i+1)} \leftarrow \boldsymbol{X}^{(i)} + \Delta \boldsymbol{X}$;
  end
  $\boldsymbol{X}_t \leftarrow \boldsymbol{X}^{(i)}$;
  Update local loss function $f_{\boldsymbol{X}_t}(\boldsymbol{W})$;
  for $j \leftarrow 0, \ldots, J-1$ **do**
    $\Delta \boldsymbol{W} = \text{LSTM}_W(\nabla f_{\boldsymbol{X}_t}(\boldsymbol{W}^{(j)}), \boldsymbol{C}_W^{(j)}, \boldsymbol{\theta}_W^s)$
    $\boldsymbol{W}^{(j+1)} \leftarrow \boldsymbol{W}^{(j)} + \Delta \boldsymbol{W}$;
  end
  $\boldsymbol{W}_t \leftarrow \boldsymbol{W}^{(j)}$;
  Update local loss function $f_{\boldsymbol{W}_t}(\boldsymbol{X})$;
  Update global loss function $F(\boldsymbol{W}_t, \boldsymbol{X}_t)$;
  for $s \leftarrow 1, \ldots t/t_{out}$ **do**
    $\mathcal{L}_F^s = \frac{1}{t_{out}} \sum_{t_s=(s-1)t_{out}+1}^{st_{out}} \omega_{t_s} F(\boldsymbol{W}_{t_s}, \boldsymbol{X}_{t_s})$
    $\boldsymbol{\theta}_X^{s+1} = \boldsymbol{\theta}_X^s - \alpha_X \nabla_{\boldsymbol{\theta}_X^s} \mathcal{L}_F^s$
    $\boldsymbol{\theta}_W^{s+1} = \boldsymbol{\theta}_W^s - \alpha_W \nabla_{\boldsymbol{\theta}_W^s} \mathcal{L}_F^s$
  end
end

At this stage, the proposed MLAM algorithm is summarized in Algorithm 1. Specifically, the algorithm starts from a random initialization $\boldsymbol{W}_0$ at the beginning of an outer loop. At the tth outer loop, it contains two inner loops for updating $\boldsymbol{X}$ and $\boldsymbol{W}$, respectively. Each inner loop starts from a random initialization $\boldsymbol{X}^{(0)}$ and $\boldsymbol{W}^{(0)}$ and updates variables based on (8) and (9) and then repeats for I and J times, respectively. In this article, we set $I = J = t_{\text{in}}$ in which t_{in} indicates the maximum iteration number at the inner loops. The choices for t_{out} and t_{in} will be discussed in Section V-A. In practice, it is reasonable to set different values for I and J according to the demand of the objective in different problems. At the end of each inner loop, the output of this inner loop is regarded as $\boldsymbol{X}_t$ or $\boldsymbol{W}_t$ at the tth outer loop and is then assigned to generate subproblem $f_{\boldsymbol{X}_t}$ and $f_{\boldsymbol{W}_t}$, respectively. For every t_{out} step at outer loops, the parameters $\boldsymbol{\theta}_X$ and $\boldsymbol{\theta}_W$ are updated following equations (11).

Finally, our MLAM algorithm will stop when t reaches the maximum outer loop number T.

Compared to the deep unfolding algorithms, the fundamental differences of the MLAM method are mainly threefold: 1) replacing the variable update function within each iteration with a metanetwork instead of mapping the whole procedure at each iteration by a black-box-based network operator; thus, the inner loop at each iteration is interpretable; 2) focusing on learning a new strategy for the whole iterations instead of following the basics of the original strategy with end-to-end network behavior on trainable parameters, leading to a better explainability on the optimization manner; and 3) it is feasible for unsupervised learning, while the deep unfolding methods are mostly supervised learning. Therefore, the proposed MLAM method highlights advances in model explainability and the improvements in combining the advantages of learning- and model-based approaches. We believe that this is a further step ahead that makes a higher level of learning than the deep unfolding, where the learning objective is no longer the trainable parameters but the whole optimization strategy.

In a summary, a hierarchical LSTM-based MLAM model is proposed in this section. It contains two (or more for multiple variables if needed) LSTM networks that perform optimization at inner loops with frozen parameters; their parameters are updated during outer loops with respect to minimizing accumulated global losses. Therefore, the original structure of algorithms well-established in the field is maintained, while the performance is improved by the metalearned algorithmic rule.

IV. Applications in Typical Problems

In this section, we shall apply the proposed MLAM approach to a bilinear inverse problem and a nonlinear problem. For those problems, the model-based methods perform less effectively, and learning-based methods typically do not work due to the lack of sufficient labeled data. Specifically, matrix completion and GMM problems are analyzed in this article as two representatives.

A. Bilinear Inverse Problem: Matrix Completion

The bilinear inverse problem is a typical optimization problem whose variables are within an intersection of two sets. Many nonconvex optimization problems can be cast as bilinear inverse problems, including low-rank matrix recovery [39]–[43], dictionary learning [44]–[46], and blind deconvolution [47]–[49]. For example, in bind deconvolution and the matrix completion, $F(\cdot,\cdot)$ in (1) represents circular convolution and matrix product, respectively.

Next, we focus on the matrix completion [41]–[43], [50], [51] as a representative bilinear inverse problem to demonstrate how to apply MLAM to solve this type of problems. Matrix completion is a class of tasks that aims to recover missing entries in a data matrix [50]–[52] and has been widely applied in practice, including recommending system [43] and collaborative filtering [53]. Typically, it is formulated as a low-rank matrix recovery problem in which the matrix to be completed is assumed to be low-rank with given rank information.

As for optimization, the low-rank matrix completion problem is usually formulated as the multiplication of two matrices and then converted into two corresponding subproblems that are generally strongly convex [39], [40], [54]. Then, gradient descent-based methods, such as alternating least square (ALS) [55] and stochastic gradient descent (SGD) [56], are often applied on solving each subproblem and can achieve good performance under certain conditions. However, satisfactory results are not universally guaranteed. On the one hand, the performance depends on a set of factors, including initialization strategies, the parameter setting of gradient descent algorithms, the sparsity level of the low-rank matrix, and so on. On the other hand, the assumptions behind these constraints are stringent, such as the feature bias on variables for sparse subspace clustering mechanism [52], [57], which are not common in practice.

Therefore, the bilinear inverse problem is basically ill-posed and, thus, has always been a challenging task. Consequently, the performance of the traditional model-based methods is highly limited by the prior knowledge of models and structural constraints. Meanwhile, the strong nonconvexity and constraints also limit the application of the deep learning and unfolding techniques in this type of problem.

In this section, we further consider several more realistic and, therefore, challenging scenarios, including high-rank matrix completion, matrix completion without given rank knowledge, and mixed rank matrix completion problems, where the existing model-based and even learning-based approaches fail.

The matrix completion problem is then formulated as [58]

$$\min_{\boldsymbol{U},\boldsymbol{V}} F(\boldsymbol{U},\boldsymbol{V}) := \frac{1}{2}\left\|\mathcal{P}_{\Omega}\left(\boldsymbol{R}-\boldsymbol{U}\boldsymbol{V}^{T}\right)\right\|_{F}^{2}+\frac{\lambda}{2}\left(\|\boldsymbol{U}\|_{F}^{2}+\|\boldsymbol{V}\|_{F}^{2}\right) \tag{13}$$

where the projection $\mathcal{P}_{\Omega}(\cdot)$ preserves the observed elements defined by $\boldsymbol{\Omega}$ and replaces the missing entries with 0, and λ is the weight parameter of the regularizers. The matrix completion problem is typically formulated as a low-rank matrix recovery problem, which parameterizes a low-rank matrix $\boldsymbol{R}\in\mathbb{R}^{z\times q}$ as a multiplication of two matrices $\boldsymbol{U}\boldsymbol{V}^{T}$ with $\boldsymbol{U}\in\mathbb{R}^{z\times p}$, $\boldsymbol{V}\in\mathbb{R}^{q\times p}$ and $p\leq\min(z,q)$.

Remark 3: When taking the matrix multiplication $\boldsymbol{R}=\boldsymbol{U}\boldsymbol{V}^{T}$, the parameter p is set to the rank of $\boldsymbol{R}$, which is typically known. If the rank is not provided, the problem will be much more difficult, and the existing methods would be unworkable. In Section V-A, we will verify that the proposed MLAM still works properly when p is unknown.

Here, we define (13) as the overall problem and $F(\boldsymbol{U},\boldsymbol{V})$ as the global loss function. It is obvious that the overall problem is not convex in terms of $\boldsymbol{U}$ and $\boldsymbol{V}$, but the subproblems are convex when fixing one variable and updating the other. Therefore, we split (13) into two subproblems in quadratic form, fixing one variable in (13) and updating the other one, referring to $f_{\boldsymbol{U}}(\boldsymbol{V})$ and $f_{\boldsymbol{V}}(\boldsymbol{U})$. The ALS and SGD methods can be applied by iteratively minimizing the two subproblems.

Meanwhile, according to Algorithm 1, our MLAM method can be directly applied to solve the problem (13) using two LSTM networks $\mathsf{LSTM}_{\boldsymbol{U}}$ and $\mathsf{LSTM}_{\boldsymbol{V}}$ with parameters $\boldsymbol{\theta}_{\boldsymbol{U}}$ and $\boldsymbol{\theta}_{\boldsymbol{V}}$ to optimize matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. The variable update equations are given by

$$\begin{aligned} \boldsymbol{U}^{(i)} &= \boldsymbol{U}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{U}}^{(i-1)} \\ \boldsymbol{V}^{(j)} &= \boldsymbol{V}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{V}}^{(j-1)} \end{aligned} \tag{14}$$

where $\boldsymbol{H}_{\boldsymbol{U}}$ and $\boldsymbol{H}_{\boldsymbol{V}}$ are the outputs of $\mathsf{LSTM}_{\boldsymbol{U}}$ and $\mathsf{LSTM}_{\boldsymbol{V}}$, respectively. Two LSTM networks are updated for every t_{out} outer loop steps via backpropagating accumulated global losses $\mathcal{L}_F^s$ according to (10). In this way, the updating rule of the learned algorithm could be adjusted to find gradient descent steps $\boldsymbol{H}_{\boldsymbol{U}}$ and $\boldsymbol{H}_{\boldsymbol{V}}$ for minimizing $F(\boldsymbol{U}, \boldsymbol{V})$ adaptively.

The advantages of our MLAM model for the matrix completion problem may be explained by the replacement of updating step function ϕ by the neural networks. In the AM methods, when a local optimum $(\boldsymbol{U}', \boldsymbol{V}')$ is reached, the gradient of the variables equals to zero, e.g., $(\partial f_{\boldsymbol{V}'}(\boldsymbol{U})/\partial \boldsymbol{U})|_{\boldsymbol{U}=\boldsymbol{U}'} = \boldsymbol{0}$. At this stage, the update function, which is determined by the gradients of variables, will be stuck at the local optimum, while, in our MLAM model, our update functions are determined by their parameters, $\boldsymbol{\theta}_{\boldsymbol{U}}$ and $\boldsymbol{\theta}_{\boldsymbol{V}}$, which are further determined by leveraging on partial global loss trajectories across the outer loop steps. This major difference mainly brings two benefits to our MLAM method. One advantage is that, even at local optimum points, our MLAM can still provide a certain step update on variables. This can be understood that, even when one of the input $(\partial f_{\boldsymbol{V}'}(\boldsymbol{U})/\partial \boldsymbol{U})|_{\boldsymbol{U}=\boldsymbol{U}'} = \boldsymbol{0}$, $\mathsf{LSTM}_{\boldsymbol{U}}$ still obtains some nonzero outputs given a nonlinear function of zero input, cell state, and parameters. Another advantage is that the leveraged global loss leads to a smooth optimization on a global loss landscape, which allows the learned algorithm to be essentially guided by the inductive bias from a smoother transform of the global loss landscape. In Section V, besides the traditional low-rank matrix completion problem, we further evaluate MLAM in matrix completion problems in the case of high rank, unknown rank, and mixed rank (a set of matrices with different ranks, from low rank to full rank).

B. Nonlinear Problem: Gaussian Mixture Model

Different from the bilinear inverse problems, optimization over the intersection of two variables that have a nonlinear mapping between variables and observed samples also plays a significant role in statistical machine learning, including the Bayesian model [59], the graphic model [60], [61], and the finite mixture model [62]. The subproblem in nonlinear problem typically has no closed-form solution, and the overall problem possesses many local optimums. Therefore, it is difficult to guarantee the convergence of the model-based approaches, especially for the global optima, as well as intractable to obtain labeled data for the existing learning-based methods.

GMM [63]–[65] is one of the most important probabilistic models in machine learning. GMM problem, consisting of a set of Gaussian distributions in the form of weighted Gaussian density components, is usually estimated by the maximum log-likelihood method [65]. The variables in the GMM problem possess a nonlinear mapping to the observations. Many methods have been proposed to solve the GMM problem, i.e., maximizing the likelihood of GMM problems, such as conjugate gradients, quasi-Newton and Newton [66]. However, these methods typically perform inferior to the one called expectation–maximization (EM) algorithm [62], [63]. One possible reason is due to the nonconvexity and nonlinearity of the GMM problem that requires a sophisticated step descent strategy to find a good stationary point. On the other hand, the EM algorithm omits the hyperparameter related to the step size by converting the origin estimation problem of maximizing likelihood into a relaxed problem where a lower bound is maximized monotonically and analytically. Even though the EM algorithm has been widely applied to the GMM problem, it also suffers from the aforementioned nonconvexity and nonlinearity. When the nonconvexity is high (referring to some real-world scenarios: the number of observations is not sufficient, dealing with high-dimensional data [67] and a large number of clusters), the convergence is not guaranteed and the performance significantly degrades. Many works attempt to replace the EM algorithm by reformulating GMM by adopting matrix manifold optimization [64], [68] and also a learning-based method in high dimensions [67].

Detailed descriptions of GMM problems can be found in [69]. Given a set of G i.i.d. samples $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^G$, each entry $\boldsymbol{x}_g$ is a D-dimensional data vector. Then, a typical optimization when using the GMM to model the samples is to maximize the log-likelihood (MLL) [65], which is equivalent to minimize the Kullback–Leibler divergence from the empirical distribution. The parameters of the GMM can then be optimized as follows:

$$\max_{\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K} \log p(\boldsymbol{X}) = \sum_{g=1}^{G} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{x}_g \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \tag{15}$$

where

$$\mathcal{N}\left(\boldsymbol{x}_g \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) = \frac{\exp\left\{-\frac{1}{2}\left(\boldsymbol{x}_g - \boldsymbol{\mu}_k\right)^T \boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{x}_g - \boldsymbol{\mu}_k\right)\right\}}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_k|^{1/2}}. \tag{16}$$

In (15), for the kth Gaussian component, $\boldsymbol{\mu}_k$ is the mean vector and defines the cluster center, covariance $\boldsymbol{\Sigma}_k$ denotes the cluster scatter, π_k represents mixing proportion with $\sum_{k=1}^K \pi_k = 1$, and $|\boldsymbol{\Sigma}_k|$ represents the determi $>$ 1t of $\boldsymbol{\Sigma}_k$.

However, it is intractable to directly obtain a closed-form solution that maximizes $\log p(\boldsymbol{X})$ in (15). The key difficulty is that, by differentiating $\log p(\boldsymbol{X})$ (summation of logarithmic summation) and equalizing it to 0, each parameter is intertwined with each other. Gradient descent methods in an AM manner can alternatively solve (15), but they typically perform inferior to the EM algorithm [62], [63].

We should also point out that the EM algorithm still updates the GMM parameters in an AM manner, i.e., iterating to optimize over each parameter at which its gradient equals 0 with the other parameters being frozen. Furthermore, the EM algorithm, together with other first-order methods, has been proven to converge to arbitrary bad local optimum

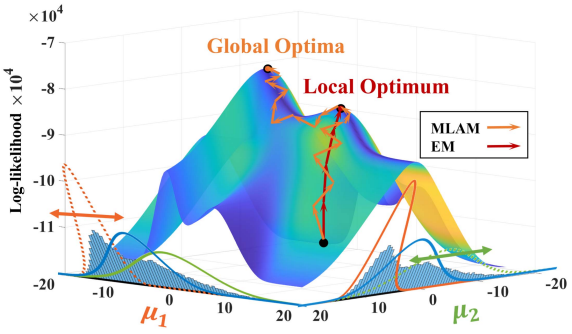


Fig. 3. Illustration of converge trajectories achieved by the proposed MLAM and EM methods in a simple GMM problem. In this GMM, the samples were obtained by three Gaussian distributions of parameters $\{\mu_1 = -3, \mu_2 = 3, \mu_3 = 0\}$, $\{\sigma_1 = 1, \sigma_2 = 10, \sigma_3 = 5\}$, and $\{\pi_1 = \pi_2 = \pi_3 = (1/3)\}$. For the convenience of visualization, we here show the surface of log-likelihood and converge trajectories with regard to μ_1 and μ_2, whereas setting the other parameters to the ground-truth when optimizing GMMs; this means that the global optima exist at $\mu_1 = -3$ and $\mu_2 = 3$.

almost surely [70]. As we have discussed before, this AM strategy can be improved by replacing the frozen updating rule that searches for the optimum in a local landscape with a less-greedy rule that updates variables up to global scope knowledge on the loss landscape of the global objective function.

Therefore, we propose to solve GMMs by adopting our MLAM method, which directly applies a learning-based gradient descent algorithm to the original meta-learning (ML) problem (15) without any extra constraints. In this scenario, we consider the GMM problem with covariance $\boldsymbol{\Sigma}$ being given; hence, the MLL estimation of GMM is presented in terms of negative log-likelihood as follows:

$$\min_{\{\pi_k,\boldsymbol{\mu}_k\}_{k=1}^K} F_{\text{ne}}(\boldsymbol{\pi},\boldsymbol{\mu}) = -\sum_{g=1}^{G}\log\sum_{k=1}^{K}\pi_k\mathcal{N}\left(\boldsymbol{x}_g \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \quad (17)$$

In this case, we treat the (17) as our overall problem and split it into two subproblems $f_{\boldsymbol{\pi}}(\boldsymbol{\mu})$ and $f_{\boldsymbol{\mu}}(\boldsymbol{\pi})$. Define vector $\boldsymbol{\pi} \in \mathbb{R}^K$ and matrix $\boldsymbol{\mu} \in \mathbb{R}^{K\times D}$, where K is the maximum number of clusters and D is the dimensionality of samples. In our MLAM framework, we build two MetaNets $\text{LSTM}_{\boldsymbol{\pi}}$ and $\text{LSTM}_{\boldsymbol{\mu}}$ with parameters $\boldsymbol{\theta}_{\boldsymbol{\pi}}$ and $\boldsymbol{\theta}_{\boldsymbol{\mu}}$ to update $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ as follows:

$$\begin{aligned}\boldsymbol{\pi}^{(i)} &= \boldsymbol{\pi}^{(i-1)} + \boldsymbol{H}_{\boldsymbol{\pi}}^{(i-1)}\\ \boldsymbol{\mu}^{(j)} &= \boldsymbol{\mu}^{(j-1)} + \boldsymbol{H}_{\boldsymbol{\mu}}^{(j-1)}\end{aligned} \quad (18)$$

where $\boldsymbol{H}_{\boldsymbol{\pi}}$ and $\boldsymbol{H}_{\boldsymbol{\mu}}$ are the outputs of the two LSTM neural networks, respectively.

Similar to applying MLAM in matrix completion, we start from a random initialization $\boldsymbol{\mu}^0$ and $\boldsymbol{\pi}^0$. During our MLAM procedure given in Algorithm 1, $\boldsymbol{\pi}_k$ and $\boldsymbol{\mu}_k$ are updated based on (18) for t_{in} steps in inner loops, respectively. Meanwhile, we backforward the accumulated global losses with respect to the negative log-likelihood of GMM, referring to $\mathcal{L}_F^s$, for every t_{out} steps on the outer loops. In this way, the algorithm is updated based on the global scope knowledge about the global losses across outer loop iterations.

Considering that the existing numerical gradient-based solutions typically perform less effectively and accurately than the EM algorithm [63], we focus on comparing our MLAM method and the EM algorithm. The EM algorithm converts maximization of the log-likelihood into maximization on its lower bound; hence, it has closed-form formulations to implement an AM strategy on its maximization step for variable updating. Nevertheless, our MLAM method directly optimizes the cost function, i.e., the log-likelihood, and the variables are updated constantly up to the global scope knowledge extracted by LSTMs. In Fig. 3, we show the convergence trajectory of our MLAM and EM algorithm on the geometry of a GMM problem with three clusters. As mentioned in Section III-B, the proposed MLAM with accumulated global losses allows the learned algorithm converges in a nonmonotonic way. It can be seen that the EM algorithm (red arrows) quickly converges to the local optimum and stops at it. In contrast, the MLAM approach (orange arrows), first going to the local optimum yet, is able to escape from the local optimum and converges to the global optima. Specifically, when encountering the local optimum, the proposed MLAM first goes down on the geometry and then moves toward the global optima, thus escaping from the local optimum. This reveals that the learned algorithm does not request each step moving toward the most ascending direction, but the global losses on a partial trajectory should be minimized, recalling the accumulated global losses minimization $\mathcal{L}_F^s$ in (10). Therefore, the MLAM enjoys significant freedom to learn a nonmonotonic algorithm for convergence in terms of the update strategy on $\mathcal{L}_F^s$. We also shall point out that this also results in fluctuations on the trajectory, which could increase the needed iterations when the geometry is smooth and benign.

V. Experiments

In this section, we will present simulation results on the aforementioned matrix completion and GMM problems to validate the effectiveness and efficiency of our MLAM method.

For the experimental settings, our MetaNets employ two-layer LSTM networks with 500 hidden units in each layer. Each network is trained by minimizing the accumulated loss functions according to (10) via truncating backpropagation through time (BPTT) [71], which is a typical training algorithm to update weights in RNNs, including LSTMs. The weights of LSTM are updated by Adam [37], and the learning rate is set to 10^{-3}. In all simulations, we set $\omega_{t_s} = 1$ for simplicity. The parameters of LSTM networks are randomly initialized and continuously updated through the whole training process. For evaluation, we fix the parameters of our MLAM model and evaluate the performance of the testing datasets.

A. Numerical Results on Matrix Completion

In this section, we consider learning to optimize synthetic D-dimensional matrix completion problems. We take $D = 10$ and $D = 100$ to evaluate algorithms for both small and large-scale cases. For each matrix completion problem,

the ground-truth matrix $\boldsymbol{R} \in \mathbb{R}^{D\times D}$ is randomly and synthetically generated with rank p. Meanwhile, the observation $\boldsymbol{R}_S = \mathcal{P}_\Omega(\boldsymbol{R})$ is generated by randomly setting a certain percentage of entries in $\boldsymbol{R}$ to be zeros, and the nonzero fraction of entries is the observation rate. Matrix completion for $\boldsymbol{R}$ is then achieved by solving the low-rank matrix recovery on $\boldsymbol{R}_S$ in the form of (13) in Section IV-A. The two factorized low-dimensional matrices $\boldsymbol{U} \in \mathbb{R}^{D\times p}$ and $\boldsymbol{V}^{D\times p}$ are then used to generate reconstruction of the ground-truth matrix, denoted by $\hat{\boldsymbol{R}} = \boldsymbol{U}\boldsymbol{V}^T$. The evaluation criterion is given by the relative mean square error (RMSE) of

$$\text{RMSE} = \frac{\left\|\boldsymbol{R} - \hat{\boldsymbol{R}}\right\|_F}{\|\boldsymbol{R}\|_F}.$$

As aforementioned, the classical AM-based ALS [55] and SGD [56] approaches, as well as the learning-based deep matrix factorization (DMF) [5] and unfolding matrix factorization (UMF) methods [11], have been adopted for comparisons. The parameters of all compared methods are carefully adjusted to present their best performances.

Different simulation scenarios on matrix completion are evaluated comprehensively. The detailed simulation settings include: 1) each simulation contains a set of 200 matrix completion problems, half of which are employed as training samples for parameter update on LSTM, while the remaining 100 matrix completion problems are used to evaluate the performance as testing samples; 2) we set $T = 100$ as total alternating steps for each problem; 3) the averaged RMSE over 100 testing samples is used for evaluation; and 4) in all the training and testing processes, the ground-truth matrix $\boldsymbol{R}$ is not given and is only used to evaluate performance after the optimizing processes.

1) Parameter Setting: The numbers of variable update steps on inner loops t_{in} and update interval t_{out} are the two most important hyperparameters. Different settings on t_{in} and t_{out} are, thus, tested at first to provide a brief guidance on the choices of t_{in} and t_{out}.

Empirically, there is a tradeoff between performance and efficiency. Here, we set t_{in} and t_{out} to both vary from 1 to 20 with 20% observation rate, and there are, therefore, $20 \times 20 = 400$ different parameter combinations to be evaluated for rank-5 matrix completion problems. The performance of these 400 simulations are shown in Fig. 4. We can see that, for all choices of t_{out}, the increase on t_{in} leads to significant improvements on performance when $t_{\text{in}} \leq 10$; however, further increasing on t_{in} witnesses little gain on performances. At the same time, the variations of t_{out} have less impact on RMSE results, while larger choices ($t_{\text{out}} \geq 10$) bring improved stability (there are less fluctuations when $t_{\text{in}} \geq 10$ and $t_{\text{out}} \geq 10$). Fig. 4 indicates that a sufficient number of update steps for inner and outer loops play a significant role in these optimization processes. We can see that, when either t_{in} or t_{out} is small (less than 5), the optimization does not perform well enough. A larger choice of t_{in}, however, typically brings higher computational cost. Thus, in the rest of this article, we set $t_{\text{in}} = 10$ and $t_{\text{out}} = 10$ as the default parameter setting.

It is understandable that t_{in} directly determines the number of variable update steps within each inner loop. When

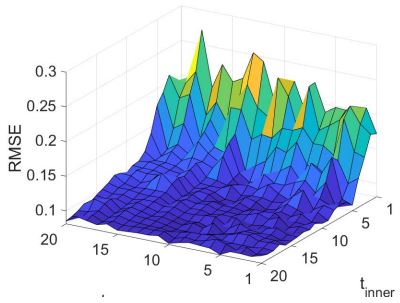


Fig. 4. Performance of the proposed MLAM method on the rank-5 matrix completion problem with different parameter combinations of t_{in} and t_{out} that range from 1 to 20 with a step size of 1.

t_{in} is small, the learned updating rule needs to optimize variables in a few steps; however, this could be intractable in general. Meanwhile, the accumulated global losses depend on t_{out} at outer loops, which indicates the length of the trajectory of global losses at outer loops. Therefore, a large enough t_{out} could provide sufficient global losses trajectory for parameter update. According to the two-level metalearning in our MLAM model, sufficient update steps at inner and outer loops can ensure that each level of metalearning works well. We infer that small t_{in} may limit the ground-level metalearning corresponding to inner loops, making it unable to extract effective subproblems across knowledge with merely few update steps, and small t_{out} could cause that the upper level metalearning corresponding to outer loops becomes less stable due to the lack of updates on parameters.

2) Standard Matrix Completion: In this part, we will first evaluate all methods on low-rank matrix completion tasks and then apply them to high-rank matrix completion tasks. In these simulations, the rank of the matrices is given. In this section, the main goal is the proof of the concept of the proposed MLAM. We evaluate four existing approaches, including conventional model-based methods and the state-of-the-art learning-based methods for comparison.

In Table I, the average RMSE of the five evaluated methods on four sets of 100-D rank-5 matrix completion problems has been reported. Different sets have different observation rates, including 20%, 40%, 60%, and 80%. From Table I, it is clear that our MLAM algorithm significantly outperforms all the existing methods, especially in high observation rate scenarios. It is also noticeable that, when the observation rate is 20%, both the model-based methods (ALS and SGD) and learning-based methods (DMF and UMF) work not well, while our MLAM method achieves good reconstruction with RMSE < 0.1.

In Fig. 5, we present the RMSE variance of the five tested methods on 100 trails of rank-5 matrix completion tasks with 20% observation rate. It can be observed that the MLAM obtains the best performance while keeping the variance relatively small. It is noticeable that the ALS approach fluctuates severely, while the SGD and DMF approaches present relatively large variance compared to the MLAM. Though the UMF method gains robust results with small variance, the accuracy is significantly larger than the MLAM.

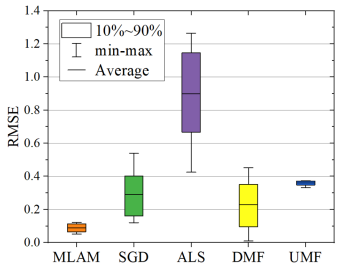


Fig. 5. RMSE variance of the evaluated methods on rank-5 matrix completion tasks with 20% observation rate.

TABLE I

RMSE OF RANK-5 MATRIX COMPLETION WITH DIFFERENT OBSERVATION RATES Θ

Methods \ Θ	0.2	0.4	0.6	0.8
MLAM	**0.089**	**0.057**	**0.045**	**0.003**
SGD	0.292	0.213	0.136	0.072
ALS	0.864	0.717	0.423	0.228
DMF	0.228	0.091	0.057	0.025
UMF	0.363	0.359	0.335	0.312

From these simulation results, we can conclude that, in the classic low-rank matrix completion problem, our MLAM method has shown better performances than the comparison methods, especially when the observation rate is low.

We then consider more challenging matrix completion problems. Generally, the matrix completion problem is assumed to be solved as a low-rank matrix completion problem. Here, we are aiming to solve high-rank or even full-rank matrix completion problems without adding any further assumptions. In this case, we test all the five approaches on six sets of 100-D matrix completion problems with 20% observation rate, whose ranks are ranged from 5 to 100.

The averaged RMSE results of matrix completion problems with variant rank are listed in Table II. There is a clear trend of decreasing performances of the comparison methods when the rank of matrices increases. Noticeably, the ALS method is no longer workable when the rank is larger than 10. SGD, DMF, and UMF methods fail after the rank reaches 40. This can be understood that these approaches are typically assumed specific penalty terms for recalling the low-rank property. However, the proposed MLAM method shows different results: the higher the rank, the smaller the RMSEs. We can conclude that our algorithm is capable of solving the matrix completion problem in high-rank, even full-rank scenarios, without any extra constraints, while standard methods typically fail.

The major computational cost in each iteration of MLAM is from the network-based computation of the gradient update term at inner loops. In this instance, due to the implementation of the networks at each inner loop iteration, the computational complexity of the MLAM approach is significantly regarded to the number of inner and outer loops. The execution time of the evaluated methods is compared in Table IV. It can be seen that the proposed MLAM approach has a larger time cost than the compared methods, but the difference is within

TABLE II

RMSE OF MATRIX COMPLETION WITH DIFFERENT RANKS

Methods \ Rank	5	10	20	40	80	100
MLAM	**0.089**	**0.085**	**0.078**	**0.072**	**0.053**	**0.052**
SGD	0.292	0.405	0.689	>1	>1	>1
ALS	0.864	0.942	>1	>1	>1	>1
DMF	0.228	0.263	0.392	0.643	>1	>1
UMF	0.363	0.376	0.417	0.582	>1	>1

TABLE III

RMSE OF RANK-10 MATRIX COMPLETION WITH VARIANT FACTORIZED MATRIX DIMENSION p

Methods \ p	10	20	40	80	100
MLAM	**0.09**	**0.12**	**0.19**	**0.30**	**0.36**
SGD	0.31	0.78	>1	>1	>1
ALS	0.89	>1	>1	>1	>1
DMF	0.26	0.53	0.81	>1	>1
UMF	0.37	0.68	0.87	>1	>1

TABLE IV

TIME COMPLEXITY OF DIFFERENT METHODS

Methods	MLAM	DMF	UMF	ALS	SGD
Time(ms)	4618	3704	2243	**1272**	1518

the second level. This is caused by the alternatively utilizing network at each iteration. There is a tradeoff between time consumption and performance: the higher the accuracy, the larger the time consumption. In this article, the main goal is the proof of the concept that the MLAM approach could provide better performance in solving the multivariable nonconvex optimization problem. In future work, we will further improve the computational speed of the MLAM to reduce the time cost for better practical applications.

3) Blind Matrix Completion: In this part, we consider more challenging cases where the rank information is not known. We consider that these are blind matrix completion problems that are typically intractable using previous methods. We first test our MLAM method and standard methods in the case where a set of matrices completion problems has the same but unknown rank. Then, we will further test our MLAM method with a more difficult case, which has a set of matrices completion problems with different unknown ranks.

In the first case, we test our MLAM method and two standard methods on a set of rank-10 matrix completion problems with variant factorized matrix dimension p. In Table III, we report the results of applying different p for reconstructing a rank-10 matrix through the five tested approaches. We can see that the proposed MLAM method is more robust when there is a mismatch between p and the true rank compared to the other methods.

When we take a large gap, such as $p = 80$ or $p = 100$, to reconstruct a rank-10 matrix, the MLAM method still achieves acceptable performances. Meanwhile, the rest of the tested methods quickly degrades with the increase in the mismatch between p and the rank values. The RMSEs of the existing methods quickly arise to more than 100%

TABLE V
RMSEs OF MIXED MATRIX COMPLETION WITH DIFFERENT CHOICES OF p

p \ Rank	10	20	30	40	50	60	70	80	90	100
10	0.120	0.081	0.065	0.055	0.049	0.046	0.043	0.042	0.042	**0.039**
50	0.250	0.130	0.095	0.075	0.067	0.055	0.052	0.048	0.046	**0.045**
100	0.450	0.210	0.140	0.100	0.081	0.068	0.061	0.056	0.052	**0.048**

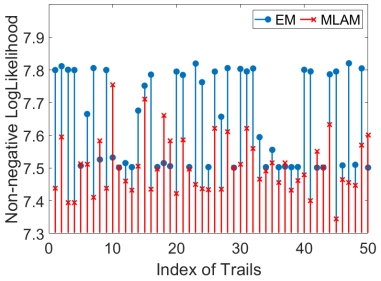


Fig. 6. Performance of EM and MLAM over 50 testing samples. Averaged nonnegative log-likelihood of 50 trails through EM algorithm is 7.77, while the result through MLAM model is 7.56.

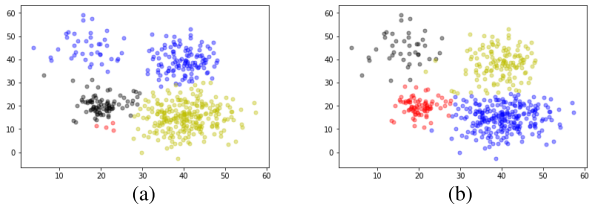


Fig. 7. Clustering results of one GMM problem with random initialization. (a) EM clustering result. (b) MLAM clustering result.

when $p \geq 40$. In contrast, our MLAM method shows a good tolerance for the increase in the difference between the real rank and p. Thus, the MLAM method does not require accurate rank information to achieve a successful matrix completion.

Then, we test our MLAM method on matrix completion problems with different ranks for the matrix $\boldsymbol{R}$. This means that, in the training and testing sample sets, there are mixed matrices with different ranks. Thus, the learned algorithm is required to adapt to matrix completion problems across different ranks.

In Table V, RMSE results of three sets of matrix completion problems are listed. For each set, 100 training samples and 100 testing samples are generated by setting their ground-truth rank uniformly distributed between 10 and 100 with a step size of 10. We take three different choices of p for each set and calculate the mean RMSE on the samples of each rank, respectively.

The proposed MLAM method in the three cases performs generally well on samples with different ranks. The comparison of the three cases further reveals that our method of choosing $p = 10$ achieves the best performance in all samples, especially in low-rank scenarios where it provides significantly better results than the others. For the other two choices of p, our method also has comparable performances on high-rank samples and achieves sufficiently good accuracy on the majority of the samples.

In summary, we have been shown that our MLAM method is capable of solving the matrix completion problem without any prior information. Typically, these problems have underlying complicated landscapes geometries. Therefore, it is hard for standard gradient descent-based methods to achieve satisfactory results. The proposed MLAM model makes it possible to find good solutions through learning an appropriate updating rule that is dynamic and adaptive.

B. Numerical Results on Gaussian Mixture Model

In this section, we apply the MLAM method to GMM problems. Given the dataset $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^{G}$ (G denotes the number of observation samples), we optimize the mean cluster center $\boldsymbol{\mu}_k$ and mixing proportion π_k while keeping the covariance Σ_k frozen, which is similar to the optimization of the *k-means* problem. We consider one GMM problem as one sample in the training and testing sets. For solving each GMM problem, we set the maximum alternating steps as 100 and record the negative log-likelihood F_{ne} trajectory according to (17). Similar to the settings in the matrix completion simulation, we also choose $t_{\text{in}} = 10$ and $t_{\text{out}} = 10$ for all the simulation scenarios in the GMM problems.

The EM algorithm has been adopted for comparison. The stopping criterion for the EM algorithm is $|F_{\text{ne}}(t-1) - F_{\text{ne}}(t)| < 10^{-4}$. The EM algorithm and our MLAM method all start from the same random initialization for $\boldsymbol{\mu}_k$ and π_k.

We first start from the 2-D GMM problems, whose data vector $\boldsymbol{x}_g \in \mathbb{R}^2$. Given four clusters ($K = 4$), $G = 500$ data points in $\boldsymbol{X} = \{\boldsymbol{x}_g\}_{g=1}^{G}$, and we show 50 tested results with random initialization in Fig. 6. We only provide 500 data points here to increase the difficulty of optimizing these GMM problems due to the limited number of observations. From Fig. 6, it can be seen that, on these 50 tests, the MLAM method outperforms the EM algorithm in most cases. The mean negative log-likelihood F_{ne} of the MLAM method on these 50 samples is 7.52, while that of the EM algorithm is 7.75. More specifically, we randomly select one clustering result among these 50 trails, which is shown in Fig. 7, which clearly shows that the MLAM method obtains much better clustering results than the EM algorithm.

Furthermore, we conduct simulations on a flower-shaped synthetic data ($G = 10\,000$) with random initializations. Each cluster in the flower-shaped data is composed of Gaussian-distributed samples. This typically is a hard problem as the anisotropic clusters lead to an extensive local optimum. Ten randomly selected optimization can be found in Fig. 8, in which our algorithm consistently achieves nearly optimal

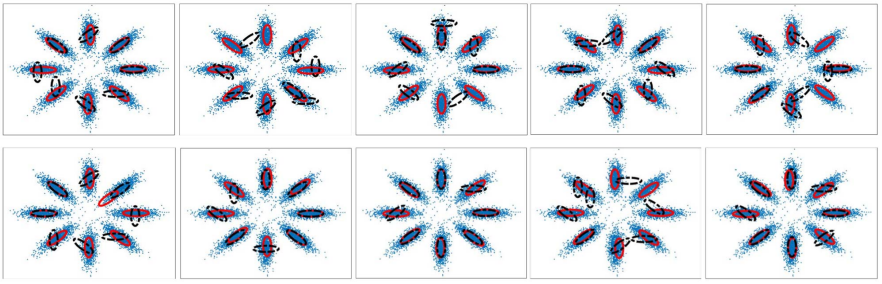


Fig. 8. EM and MLAM performance on ten flower-shaped data. Black and red dash lines denote EM and MLAM clustering results, respectively.

TABLE VI
HIGH-DIMENSIONAL GMM SIMULATION RESULTS

Methods \ Dimension	4	8	16	32	64
EM	6.52	12.64	23.83	47.10	96.88
MLAM	**6.44**	**12.42**	**23.05**	**46.93**	**95.72**

clustering, but the results from EM are highly biased. Although being illustrative in two dimensions, the data in Fig. 8 consist of eight anisotropic clusters, which might be the main reason that the EM method converges to a bad local optimum. As has been proven in [70], when solving GMMs with more than 2 clusters ($K > 2$), the EM method is highly likely to converge to arbitrarily bad local optimum under random initialization. More specifically, the EM method is a special case of the Bayesian variational inference and provides a tractable solver to maximize the lower bound of the log-likelihood of GMM problems. More importantly, maximizing the log-likelihood is equivalent to using the Kullback–Leibler (KL) divergence in estimating two distributions [63]. Thus, the performance of the EM method is basically limited to the intrinsic nature of the KL divergence, which can output infinitely large values with gradient vanishing issues when two distributions are well-separated [70], [72]. In other words, the EM method is highly sensitive to initialization and may converge to a bad local optimum from random initialization, the phenomenon that has been studied in [68]. In contrast, the proposed MLAM method consistently achieves global estimation, verifying the global optimization nature of our method. Therefore, our MLAM method obtains significant improvements in accuracy, even for some challenging scenarios, such as insufficient observations and anisotropic clusters in these illustrative evaluations.

We further evaluate our method for estimating high-dimensional GMM problems. Several sets of high-dimensional synthetic data ($G = 500$ and $K = 4$) are also randomly generated for the evaluation, with dimensions 4, 8, 16, 32, and 64. The averaged negative log-likelihoods of 100 tests for each testing dimension are reported in Table VI. It is clear that, from Table VI, our MLAM method outperforms the EM algorithm on all the high-dimensional sample sets, and the variation in dimensions does not affect the performance of our method, while this typically decreases the performance of EM algorithm in general.

In this stage, it has been verified that our proposed method is able to solve the nonconvex multivariable problem with nonlinear relationships between variables and observation data. Even without closed-form landscapes in these problems, our MLAM method still successfully finds good solutions, while the EM algorithm fails.

VI. CONCLUSION

We have proposed an MLAM method for solving nonconvex optimization problems. The learned algorithm has been verified to have a faster convergence speed and better performances than existing AM-based methods. To achieve that, our MLAM method has employed LSTM-based metalearners to build interaction between variable updates and the global loss. In this way, the variables are updated by the LSTM networks with frozen parameters at inner loops, which are then updated by minimizing accumulated global losses at outer loops. This article is just proof of the concept that a less greedy and learning-based solution for nonconvex problems could surpass both the traditional model- and learning-based methods. More importantly, these concepts optimize each independent solution step in-exhaustively by globally learning the optimization strategy, integrate these independent steps by the bilevel metalearning optimization model, and spark a new direction of improving the optimization-inspired solutions for advanced performance. It reveals that applying neural network-based behavior to assist the well-established algorithmic principle, instead of replacing it with black-box network behavior, could bring significant advances.

For future work, we plan to apply the proposed MLAM method to more challenging and practical nonconvex problems, such as dictionary learning in compress sensing, blind image super-resolution, and multiple input multiple output (MIMO) beamforming. We would also like to extend some theoretical analysis on our MLAM model. Moreover, the MLAM can be directly applied to solve nonconvex problems online without pretraining. It is deserved to discover the application of applying MLAM as an algorithm without training for nonconvex problems.

REFERENCES

[1] M. J. D. Powell, "On search directions for minimization algorithms," *Math. Program.*, vol. 4, no. 1, pp. 193–201, Dec. 1973.
[2] Q. Hu, Y. Cai, Q. Shi, K. Xu, G. Yu, and Z. Ding, "Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1394–1410, Feb. 2021.
[3] D. Ren, K. Zhang, Q. Wang, Q. Hu, and W. Zuo, "Neural blind deconvolution using deep priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3341–3350.

[4] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.

[5] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," *Neural Netw.*, vol. 98, pp. 34–41, Feb. 2018.

[6] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[7] D. You, J. Xie, and J. Zhang, "ISTA-NET++: Fexible deep unfolding network for compressive sensing," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2021, pp. 1–6.

[8] R. Fu, V. Monardo, T. Huang, and Y. Liu, "Deep unfolding network for block-sparse signal recovery," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 2880–2884.

[9] Z. Luo, Y. Huang, S. Li, L. Wang, and T. Tan, "Unfolding the alternating optimization for blind super resolution," 2020, *arXiv:2010.02631*.

[10] L. Pellaco, M. Bengtsson, and J. Jaldén, "Deep unfolding of the weighted MMSE beamforming algorithm," 2020, *arXiv:2006.08448*.

[11] T. T. N. Mai, E. Y. Lam, and C. Lee, "Ghost-free HDR imaging via unrolling low-rank matrix completion," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 2928–2932.

[12] R. Li, S. Zhang, C. Zhang, Y. Liu, and X. Li, "Deep learning approach for sparse aperture ISAR imaging and autofocusing based on complex-valued ADMM-Net," *IEEE Sensors J.*, vol. 21, no. 3, pp. 3437–3451, Feb. 2021.

[13] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, 2017.

[14] J. Li and M. Hu, "Continuous model adaptation using online meta-learning for smart grid application," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3633–3642, Aug. 2021.

[15] L. Liu *et al.*, "GenDet: Meta learning to generate detectors from few shots," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 1, 2021, doi: 10.1109/TNNLS.2021.3053005.

[16] L. Chen, B. Hu, Z.-H. Guan, L. Zhao, and X. Shen, "Multiagent meta-reinforcement learning for adaptive multipath routing optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 21, 2021, doi: 10.1109/TNNLS.2021.3070584.

[17] C. Finn and S. Levine, "Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm," 2017, *arXiv:1710.11622*.

[18] K. Li and J. Malik, "Learning to optimize," 2016, *arXiv:1606.01885*.

[19] J. Xia and G. Deniz, "Meta-learning based beamforming design for MISO downlink," 2021, *arXiv:2103.11978*.

[20] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1126–1135.

[21] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang, "Boosting few-shot learning with adaptive margin loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12576–12584.

[22] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[23] C. L. Byrne, "Alternating minimization and alternating projection algorithms: A tutorial," *Sci. New York*, pp. 1–41, 2011.

[24] W. Ha and R. F. Barber, "Alternating minimization and alternating descent over nonconvex sets," 2017, *arXiv:1709.04451*.

[25] U. Niesen, D. Shah, and G. Wornell, "Adaptive alternating minimization algorithms," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 1641–1645.

[26] T. Sun, D. Li, H. Jiang, and Z. Quan, "Iteratively reweighted penalty alternating minimization methods with continuation for image deblurring," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 3757–3761.

[27] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Math. Program.*, vol. 141, nos. 1–2, pp. 349–382, Oct. 2013.

[28] S. Guminov, P. Dvurechensky, N. Tupitsa, and A. Gasnikov, "On a combination of alternating minimization and Nesterov's momentum," 2019, *arXiv:1906.03622*.

[29] X. Zhang, W. Jiang, K. Huo, Y. Liu, and X. Li, "Robust adaptive beamforming based on linearly modified atomic-norm minimization with target contaminated data," *IEEE Trans. Signal Process.*, vol. 68, pp. 5138–5151, 2020.

[30] J. Sui, Z. Liu, L. Liu, B. Peng, T. Liu, and X. Li, "Online non-cooperative radar emitter classification from evolving and imbalanced pulse streams," *IEEE Sensors J.*, vol. 20, no. 14, pp. 7721–7730, Jul. 2020.

[31] J. Sui, Z. Liu, L. Liu, A. Jung, and X. Li, "Dynamic sparse subspace clustering for evolving high-dimensional data streams," *IEEE Trans. Cybern.*, early access, Nov. 24, 2020, doi: 10.1109/TCYB.2020.3023973.

[32] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 184–199.

[33] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 391–407.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[35] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," 2017, *arXiv:1708.08296*.

[36] K. Zhang, L. Van Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3217–3226.

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

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proc. 45th Annu. ACM Symp. Symp. Theory Comput.*, 2013, pp. 665–674.

[40] D. Gross, "Recovering low-rank matrices from few coefficients in any basis," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1548–1566, Mar. 2011.

[41] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Math. Program. Comput.*, vol. 4, no. 4, pp. 333–361, 2012.

[42] M. Hardt, "Understanding alternating minimization for matrix completion," in *Proc. IEEE 55th Annu. Symp. Found. Comput. Sci.*, Oct. 2014, pp. 651–660.

[43] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[44] I. Tosic and P. Frossard, "Dictionary learning: What is the right representation for my signal?" *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[45] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[46] Q. Yu, W. Dai, Z. Cvetkovic, and J. Zhu, "Bilinear dictionary update via linear least squares," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 7923–7927.

[47] Y. Zhang, Y. Lau, H.-W. Kuo, S. Cheung, A. Pasupathy, and J. Wright, "On the global geometry of sphere-constrained sparse blind deconvolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4894–4902.

[48] T. G. Stockham, T. M. Cannon, and R. B. Ingebretsen, "Blind deconvolution through digital signal processing," *Proc. IEEE*, vol. 63, no. 4, pp. 678–692, Apr. 1975.

[49] J. R. Hopgood and P. J. Rayner, "Blind single channel deconvolution using nonstationary signal processing," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 476–488, Sep. 2003.

[50] X. Lu, T. Gong, P. Yan, Y. Yuan, and X. Li, "Robust alternative minimization for matrix completion," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 3, pp. 939–949, Jun. 2012.

[51] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, p. 717, Dec. 2009.

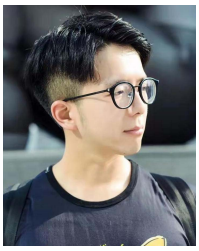
[52] J. Fan and T. W. S. Chow, "Sparse subspace clustering for data with missing entries and high-rank matrix completion," *Neural Netw.*, vol. 93, pp. 36–44, Sep. 2017.

[53] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the Netflix prize," in *Proc. Int. Conf. Algorithmic Appl. Manage.* Berlin, Germany: Springer, 2008, pp. 337–348.

[54] S. Choudhary and U. Mitra, "On identifiability in bilinear inverse problems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 4325–4329.

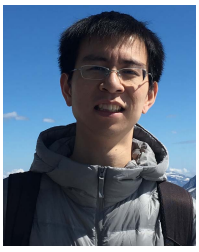
[55] P. M. Kroonenberg and J. de Leeuw, "Principal component analysis of three-mode data by means of alternating least squares algorithms," *Psychometrika*, vol. 45, no. 1, pp. 69–97, Mar. 1980.

[56] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 69–77.
[57] C. Yang, D. Robinson, and R. Vidal, "Sparse subspace clustering with missing entries," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2463–2472.
[58] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, Bonn, Germany, Aug. 2005, pp. 713–719.
[59] S. Li, W. Ying, L. Jie, and X. Gao, "Multispectral image classification using a new Bayesian approach with weighted Markov random fields," in *Proc. CCF Chin. Conf. Comput. Vis.*, 2015, pp. 168–178.
[60] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Rejoinder: Latent variable graphical model selection via convex optimization," in *Proc. Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*. IEEE, 2010, pp. 1610–1613.
[61] L. Stankovic *et al.*, "Graph signal processing—Part III: Machine learning on graphs, from graph topology to applications," 2020, *arXiv:2001.00426*.
[62] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. B, Methodol.*, vol. 39, no. 1, pp. 1–22, 1977.
[63] L. Xu and M. I. Jordan, "On convergence properties of the EM algorithm for Gaussian mixtures," *Neural Comput.*, vol. 8, no. 1, pp. 129–151, 1996.
[64] R. Hosseini and S. Sra, "Matrix manifold optimization for Gaussian mixtures," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 910–918.
[65] D. A. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, vol. 741. Boston, MA, USA: Springer, 2009.
[66] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Rev.*, vol. 26, no. 2, pp. 195–239, 1984.
[67] R. Ge, Q. Huang, and S. M. Kakade, "Learning mixtures of Gaussians in high dimensions," in *Proc. 47nth Annu. ACM Symp. Theory Comput.*, 2015, pp. 761–770.
[68] S. Li, Z. Yu, M. Xiang, and D. Mandic, "Solving general elliptical mixture models through an approximate Wasserstein manifold," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 4658–4666.
[69] G. J. McLachlan and D. Peel, *Finite Mixture Models*. Hoboken, NJ, USA: Wiley, 2004.
[70] C. Jin, Y. Zhang, S. Balakrishnan, M. J. Wainwright, and M. I. Jordan, "Local maxima in the likelihood of Gaussian mixture models: Structural results and algorithmic consequences," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4116–4124.
[71] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[72] J. Xu, D. J. Hsu, and A. Maleki, "Global analysis of expectation maximization for mixtures of two Gaussians," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.



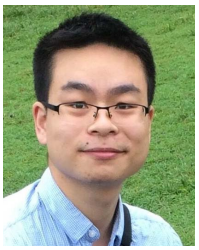
Jing-Yuan Xia received the B.Sc. and M.Sc. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2014 and 2016, respectively, and the Ph.D. degree from Imperial College London (ICL), London, U.K., in 2020.

He has been a Lecturer with the College of the Electronic Science, NUDT since 2020. His current research interests include machine learning, nonconvex optimization, and representation learning.



Shengxi Li (Member, IEEE) received the bachelor's and master's degrees from Beihang University, Beijing, China, in July 2014 and March 2016, respectively, and the Ph.D. degree from the Department of EEE, Imperial College London, London, U.K., under the supervision by Prof. Danilo Mandic, in August 2021.

He is an Associate Professor with Beihang University. His research interests include generative models, statistical signal processing, rate-distortion theory, and perceptual video coding. He is the recipient of the Imperial Lee Family Scholarship and the Chinese Government Award for Outstanding Self-financed Students Abroad.



Jun-Jie Huang (Member, IEEE) received the B.Eng. (Hons.) degree in electronic engineering and the M.Phil. degree in electronic and information engineering from The Hong Kong Polytechnic University, Hong Kong, in 2013 and 2015, respectively, and the Ph.D. degree from Imperial College London (ICL), London, U.K., in 2019.

From 2019 to 2021, he was a Post-Doctoral Researcher with the Communications and Signal Processing (CSP) Group, Electrical and Electronic Engineering Department, ICL. He is a Lecturer with the College of Computer Science, National University of Defense Technology (NUDT), Changsha, China. His research interests include the areas of computer vision, signal processing, and deep learning.

Zhixiong Yang received the B.Sc. degree from Northeastern University, Shenyang, China, in 2021. He is currently pursuing the M.Sc. degree with the College of Electronic Science, National University of Defense Technology, Changsha, China.

His research interests include deep learning in signal processing and image processing.

Imad M. Jaimoukha received the B.Sc. degree in electrical engineering from the University of Southampton, Southampton, U.K., in 1983, and the M.Sc. and Ph.D. degrees in control systems from Imperial College London, London, U.K., in 1986 and 1990, respectively.

He was a Research Fellow with the Centre for Process Systems Engineering at ICL from 1990 to 1994. Since 1994, he has been with the Department of Electrical and Electronic Engineering, ICL. His research interests include robust and fault-tolerant control, system approximation, and global optimization.

Deniz Gündüz (Fellow, IEEE) received the M.S. and Ph.D. degrees from the NYU Tandon School of Engineering (formerly Polytechnic University), Brooklyn, NY, USA, in 2004 and 2007, respectively.

After his Ph.D., he served as a Post-Doctoral Research Associate with Princeton University, Princeton, NJ, USA, as a Consulting Assistant Professor with Stanford University, Stanford, CA, USA, and as a Research Associate with CTTC in Barcelona, Spain. In September 2012, he joined the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., where he is currently a Professor of information processing and serves as the Deputy Head of the Intelligent Systems and Networks Group. He is also a part-time Faculty Member with the University of Modena and Reggio Emilia, Modena, Italy, and has held visiting positions with the University of Padua, Padua, Italy, from 2018 to 2020, and Princeton University from 2009 to 2012.

Dr. Gündüz is a Distinguished Lecturer for the IEEE Information Theory Society from 2020 to 2022. He was a recipient of a Consolidator Grant of the European Research Council (ERC) in 2022, the IEEE Communications Society-Communication Theory Technical Committee (CTTC) Early Achievement Award in 2017, a Starting Grant of the ERC in 2016, and several best paper awards.