# Investigating Bi-Level Optimization for Learning and Vision from a Unified Perspective: A Survey and Beyond

Risheng Liu, *Member, IEEE,* Jiaxin Gao, Jin Zhang, Deyu Meng, *Member, IEEE,* and Zhouchen Lin, *Fellow, IEEE*

**Abstract**—Bi-Level Optimization (BLO) is originated from the area of economic game theory and then introduced into the optimization community. BLO is able to handle problems with a hierarchical structure, involving two levels of optimization tasks, where one task is nested inside the other. In machine learning and computer vision fields, despite the different motivations and mechanisms, a lot of complex problems, such as hyper-parameter optimization, multi-task and meta learning, neural architecture search, adversarial learning and deep reinforcement learning, actually all contain a series of closely related subproblms. In this paper, we first uniformly express these complex learning and vision problems from the perspective of BLO. Then we construct a best-response-based single-level reformulation and establish a unified algorithmic framework to understand and formulate mainstream gradient-based BLO methodologies, covering aspects ranging from fundamental automatic differentiation schemes to various accelerations, simplifications, extensions and their convergence and complexity properties. Last but not least, we discuss the potentials of our unified BLO framework for designing new algorithms and point out some promising directions for future research. A list of important papers discussed in this survey, corresponding codes, and additional resources on BLOs are publicly available at: https://github.com/vis-opt-group/BLO.

**Index Terms**—Bi-level optimization, Learning and vision applications, Value-function-based reformulation, Best-response mapping, Explicit and implicit gradients.

## 1 Introduction

Bi-Level Optimization (BLO) is the hierarchical mathematical program where the feasible region of one optimization task is restricted by the solution set mapping of another optimization task (i.e., the second task is embedded within the first one) [1]. The outer optimization task is commonly referred to as the Upper-Level (UL) problem, and the inner optimization task is commonly referred to as the Lower-Level (LL) problem. BLOs involve two kinds of variables, referred to as the UL and LL variables, accordingly.

The origin of BLOs can be traced to the domain of game theory and is known as Stackelberg competition [2]. Subsequently, it has been investigated in view of many important applications in various fields of science and engineering, particularly in economics, management, chemistry, optimal control, and resource allocation problems [3], [4], [5], [6]. Especially, in recent years, a great amount of modern applications in the fields of machine learning and computer vision (e.g., hyper-parameter optimization [7], [8], [9], [10], multi-task and meta learning [11], [12], [13], neural architecture search [14], [15], [16], generative adversarial learning [17], [18], [19], deep reinforcement learning [20], [21], [22] and image processing and analysis [23], [24], [25], just name a few) have arisen that fit the BLO framework.

In general, most of the earlier BLOs are highly complicated and computationally challenging to solve due to their nonconvexity and non-differentiability [26], [27]. Despite their apparent simplicity, BLOs are nonconvex problems with an implicitly determined feasible region even if the UL and LL subproblems are convex [28], [29]. Indeed, it has been proved that even strictly checking the local optimality of the simplest BLO model (e.g., linear BLO) is still a NP-hard problem [30], [31]. In addition, the existence of multiple optima for the LL subproblem can result in an inadequate formulation of BLOs, which could aggravate the difficulty of theoretical analysis [32]. Despite the challenges, a lot of research topics consisting of methods and applications of BLOs have followed in this field, see [27], [33]. Early studies focused on numerical methods, including extreme-point methods [34], branch-and-bound methods [3], [35], descent methods [36], [37], penalty function methods [38], [39], trust-region methods [40], [41], and so on. The most often used procedure is to replace the LL subproblem with its Karush–Kuhn–Tucker (KKT) conditions, and if assumptions are made (such as smoothness, convexity, among others) the BLOs can be transformed into single-level optimization

- *R. Liu and J. Gao are with the DUT-RU International School of Information Science & Engineering, Dalian University of Technology, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China. E-mail: rsliu@dlut.edu.cn, jiaxinn.gao@outlook.com. R. Liu is the corresponding author.*
- *J. Zhang is with the Department of Mathematics, Southern University of Science and Technology, and National Center for Applied Mathematics Shenzhen, China, E-mail: zhangj9@sustech.edu.cn.*
- *D. Meng is with School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi, China. E-mail: dymeng@mail.xjtu.edu.cn.*
- *Z. Lin is with the Key Laboratory of Machine Perception (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China, and also with the Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: zlin@pku.edu.cn.*
- *R. Liu, D. Meng and Z. Lin are also with Peng Cheng Laboratory, Shenzhen, Guangdong, China.*

*Manuscript received April 19, 2005; revised August 26, 2015.*

problems [42], [43], [44]. However, due to the high complexity of bi-level models, solving BLOs for large-scale and high-dimensional practical applications in learning and vision fields is still challenging [45].

The classical idea (e.g., the first-order approach in economics literature) to reformulate BLO is to replace the LL subproblem Eq. (1) by its KKT conditions and minimize over the original variables $\mathbf{x}$ and $\mathbf{y}$ as well as the multipliers. The resulting problem is a so-called Mathematical Program with Equilibrium Constraints (MPEC) [46], [47]. Unfortunately, MPECs are still a challenging class of problems because of the presence of the complementarity constraint [48]. Solution methods for MPECs can be categorized into two types of approaches. The first one, namely, the nonlinear programming approach rewrites the complementarity constraint into nonlinear inequalities, and then allows to leverage powerful numerical nonlinear programming solvers. The other one, namely, the combinatorial approach tackles the combinatorial nature of the disjunctive constraint. Despite the difficulties, MPEC has been studied intensively in the last three decades [49]. Recently, some progress on the MPEC approach in dealing with BLOs have been witnessed by the community of mathematical programming, in the context of selecting optimal hyper-parameters in regression and classification problems. There are two issues caused by the multipliers in the MPEC approach. First, in theory, if there exist more than one multipliers for the LL subproblem, MPEC will not be equivalent to the original BLO (in the local optimality scenario) [50]. Second, the introduced auxiliary multiplier variables can limit the numerical efficiency when solving the BLO problem.

In recent years, a variety of machine learning and computer vision tasks, including but not limited to, hyper-parameter optimization [13], [51], [52], [53]), multi-task and meta learning [54], [55], [56], [57], neural architecture search [14], [16], [58], [59], adversarial learning [17], [18], [21], [60], and deep reinforcement learning [20], [21], [61], [62], have been investigated in application scenarios. Despite the different motivations and mechanisms, all these problems contain a series of closely related subproblems and have a natural hierarchical optimization structure. However, although received increasing attentions in both academic and industrial communities, there still lack a unified perspective to understand and formulate these different categories of hierarchical learning and vision problems.

We notice that most previous surveys on BLOs (e.g., [1], [63], [64], [65], [66], [67], [68]) are purely from the viewpoint of mathematical programming and mainly focus on the formulations, properties, optimality conditions and these classical solution algorithms, such as evolutionary methods [5]. In contrast, the aim of this paper is to utilize BLO to express a variety of complex learning and vision problems, which explicitly or implicitly contain closely related subproblems. Furthermore, we present a unified perspective to comprehensively survey different categories of gradient-based BLO methodologies in specific learning and vision applications. In particular, we first provide a literature review on various complex learning and vision problems, including hyper-parameter optimization, multi-task and meta learning, neural architecture search, adversarial learning, deep reinforcement learning and so on. We demonstrate that all these tasks can be modeled as a general BLO formulation. Following this perspective, we then establish a best-response-based single-level reformulation to express these existing BLO models. By further introducing a unified algorithmic framework on the single-level reformulation, we can uniformly understand and formulate these existing gradient-based BLOs and analyze their accelerations, simplifications, extensions, and convergence and complexity proprieties. Finally, we demonstrate the potentials of our framework for designing new algorithms and point out some promising research directions for BLO in learning and vision fields.

Compared with existing surveys on BLOs, our major contributions can be summarized as follows:

1) To the best of our knowledge, this is the first survey paper to focus on uniformly understanding and (re)formulating different categories of complex machine learning and computer vision tasks and their solution methods (especially in the context of deep learning) from the perspective of BLO.
2) By introducing a best-response-based single-level reformulation and constructing a best-response-based algorithmic framework, we obtain a general and flexible platform that can successfully unify different existing gradient-based BLO methodologies and uniformly analyze these accelerations, simplifications, and extensions in literature.
3) The convergence behaviors of gradient-based BLOs are comprehensively analyzed. Especially, we establish a general convergence analysis template to investigate the iteration behaviors of a series of gradient-based BLOs from a unified perspective. The time and space complexity of various mainstream schemes are also systematically analyzed.
4) Our gradient-based BLO platform not only comprehensively covers mainstream gradient-based BLO methods, but also has the potentials for designing new BLO algorithms to deal with more challenging tasks. We also point out some promising directions for future research.

We summarize our mathematical notations in Table 1. The remainder of this paper is organized as follows. We first introduce some necessary fundamentals of BLOs in Section 2. Then, Section 3 provides a comprehensive survey of various learning and vision applications that all can be modeled as BLOs. In Section 4, we establish an algorithmic framework in a unified manner for existing gradient-based BLO schemes. Within this framework, we further understand and formulate two different categories of BLOs (i.e., explicit and implicit gradients for best-response) in Section 5 and Section 6, respectively. We also discuss the so-called lower-level singleton issue of BLOs in Section 7. The convergence and complexity properties of these gradient-based BLOs are discussed in Section 8. Section 9 puts forward potentials of our framework for designing new algorithms to deal with more challenging pessimistic BLOs. Finally, Section 10 points out some promising directions for future research.

## 2 Fundamentals of Bi-Level Optimization

Bi-Level Optimization (BLO) contains two levels of optimization tasks, where one is nested within the other as a

TABLE 1
Summary of mathematical notations.

| Notation | Description | Notation | Description |
|---|---|---|---|
| $\mathcal{D}_{\mathtt{tr}}/\mathcal{D}_{\mathtt{val}}$ | Training/Validation data | $o^{ij}/\mathbf{x}_o^{ij}$ | Operations/Operation weights |
| $\pi/r$ | Policy/Reward | $s/a$ | State/Action |
| $Q^\pi$ | Q-function under $\pi$ | $Q^\pi(s,a)$ | State-action value-function |
| $G/D$ | Generator/Discriminator | $\mathbf{u}/\mathbf{v}$ | Real-world image/Random noise |
| $\rho_t$ | Aggregation parameters | $\mathbf{x}\in\mathbb{R}^m/\mathbf{y}\in\mathbb{R}^n$ | UL/LL variable |
| $F/f$ | UL/LL objective | $\mathcal{S}(\mathbf{x})$ | Solution set of the LL subproblem (given $\mathbf{x}$) |
| $\mathbf{y}^*(\mathbf{x})$ | BR mapping | $\widetilde{\mathcal{S}}(\mathbf{x})$ | Solution set of the ISB subproblem (given $\mathbf{x}$) |
| $\mathtt{dist}(\cdot)$ | Point-to-set distance | $\circ$ | Compound operation |
| $\Psi(\mathbf{x})$ | $\Psi=\Psi_T\circ\cdots\circ\Psi_1\circ\Psi_0$ | $\Psi_{\boldsymbol{\theta}}(\mathbf{x})$ | Hyper-network with parameters $\boldsymbol{\theta}$ |
| $\Psi_t$ | Dynamic system at $t$-th stage | $\frac{\partial\varphi(\mathbf{x})}{\partial\mathbf{x}}$ | Gradient of $\mathbf{x}$ |
| $\frac{\partial F(\mathbf{x},\mathbf{y}^*(\mathbf{x}))}{\partial\mathbf{x}}$ | Direct gradient of $\mathbf{x}$ | $\mathbf{G}(\mathbf{x})$ | Indirect gradient of $\mathbf{x}$ |
| $\frac{\partial\mathbf{y}^*(\mathbf{x})}{\partial\mathbf{x}}$ | BR Jacobian | $\frac{\partial\varphi(\mathbf{x}^k)}{\partial\mathbf{x}^k}$ | Numerical BR Jacobian w.r.t. $\mathbf{x}^k$ |
| $\varphi(\mathbf{x})$ | UL value-function | $\psi(\mathbf{x})$ | LL value-function |
| $\mathbf{d}(\mathbf{y}_{t-1};\mathbf{x})$ | Optimistic aggregated gradient | $\widetilde{\mathbf{d}}(\mathbf{y}_{t-1};\mathbf{x})$ | Pessimistic aggregated gradient |
| $\left(\frac{\partial^2 f}{\partial\mathbf{y}\partial\mathbf{y}'}\right)^{-1}$ | Inverse Hessian matrix | $\left(\frac{\partial^2 f}{\partial\mathbf{y}\partial\mathbf{y}'}\right)^{-1}\frac{\partial F}{\partial\mathbf{y}}$ | Inverse Hessian-vector product |
| $t/k$ | Index of the LL/UL iteration | $T/K$ | Maximum LL/UL iteration number |
| $(\cdot)_t$ | $t$-th LL iteration | $(\cdot)^k$ | $k$-th UL iteration |
| $(\cdot)'$ | Transposition operation | $(\mathbf{x}^*,\mathbf{y}^*)$ | The optimal UL and LL solutions |
| $\mathbf{Z}_T$ | $\sum_{t=1}^{T}\left(\prod_{i=t+1}^{T}\mathbf{A}_i\right)\mathbf{B}_t$ | $\mathbf{Z}_{T-M}$ | $\sum_{t=T-M+1}^{T}\left(\prod_{i=t+1}^{T}\mathbf{A}_i\right)\mathbf{B}_t$ |
| $\mathbf{P}(\mathbf{y}_{t-1},\boldsymbol{\omega})$ | Layer-wise transformation | $\sum_{j=0}^{\infty}\left(\mathbf{I}-\frac{\partial^2 f}{\partial\mathbf{y}\partial\mathbf{y}'}\right)^j$ | Neumann series |
| $\mathbf{A}_t$ | $\frac{\partial\Psi_t(\mathbf{y}_{t-1};\mathbf{x})}{\partial\mathbf{y}_{t-1}}$ | $\mathbf{B}_t$ | $\frac{\partial\Psi_t(\mathbf{y}_{t-1};\mathbf{x})}{\partial\mathbf{x}}$ |
| $\inf_{\mathbf{y}\in\mathcal{S}(\mathbf{x})}F(\mathbf{x},\mathbf{y})$ | Optimistic objective | $\psi_\mu(\mathbf{x})$ | Parameterized LL value-function (with $\mu$) |
| $\sup_{\mathbf{y}\in\mathcal{S}(\mathbf{x})}F(\mathbf{x},\mathbf{y})$ | Pessimistic objective | $\varphi_{\mu,\theta,\tau}(\mathbf{x})$ | Parameterized UL value-function (with $\mu$, $\theta$ and $\tau$) |

constraint. The inner (or nested) and outer optimization tasks are often respectively referred to as the Lower-Level (LL) and Upper-Level (UL) subproblems [1]. Correspondingly, there are two types of variables, namely, the LL ($\mathbf{y}\in\mathbb{R}^n$) and UL ($\mathbf{x}\in\mathbb{R}^m$) variables. Specifically, the LL subproblem can be formulated as the following parametric optimization task

$$\min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}),\ \text{(parameterized by } \mathbf{x}), \tag{1}$$

where we consider a continuous function $f:\mathbb{R}^m\times\mathbb{R}^n\to\mathbb{R}$ as the LL objective and $\mathcal{Y}\subseteq\mathbb{R}^n$ is a nonempty set. By denoting the value-function as $\psi(\mathbf{x}):=\min_{\mathbf{y}\in\mathcal{Y}}f(\mathbf{x},\mathbf{y})$, we can define the solution set of the LL subproblem with given $\mathbf{x}$ as $\mathcal{S}(\mathbf{x}):=\{\mathbf{y}\in\mathcal{Y}\mid f(\mathbf{x},\mathbf{y})\leq\psi(\mathbf{x})\}$. Then the standard BLO problem can be formally expressed as

$$\min_{\mathbf{x}\in\mathcal{X}} F(\mathbf{x},\mathbf{y}),\ s.t.\ \mathbf{y}\in\mathcal{S}(\mathbf{x}), \tag{2}$$

where the UL objective $F:\mathbb{R}^m\times\mathbb{R}^n\to\mathbb{R}$ is also a continuous function and the feasible set $\mathcal{X}\subseteq\mathbb{R}^m$. In fact, a feasible solution to BLO in Eq. (2) should be a vector of UL and LL variables, such that it satisfies all the constraints in Eq. (2), and the LL variables are optimal to the LL subproblem in Eq. (1) for the given UL variables as parameters. In Fig. 1, we provide a simple visual illustration for BLOs stated in Eq. (2).

The above BLO problem has a natural interpretation as a non-cooperative game between two players (i.e., Stackelberg game [1]). Correspondingly, we may also call the UL and LL subproblems as the leader and follower, respectively. Then the "leader" chooses the decision $\mathbf{x}$ first, and afterwards the "follower" observes $\mathbf{x}$ so as to respond with a decision $\mathbf{y}$.

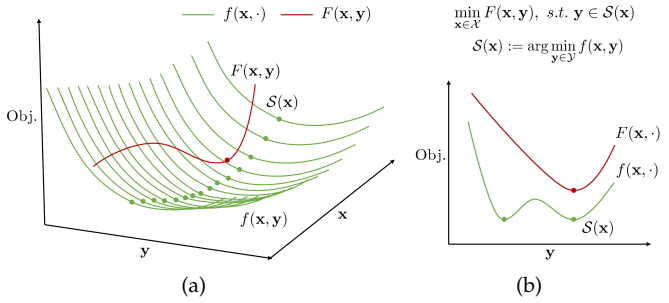



Fig. 1. Illustrating the problem of BLO. (a) first shows a standard BLO problem with the situation of multiple solutions of $f$. Green curves denote LL objectives denoted by $f$, and their corresponding minimizers given by $\mathcal{S}(\mathbf{x})$ were shown as green dots. The red curve represents the UL objective $F$, whose minimizer is shown as the red dot. (b) further illustrates that, in general, not all points (green dots) in $\mathcal{S}(\mathbf{x})$ could minimize the UL objective denoted by $F$.

Therefore, the follower may depend on the leader's decision. Likewise, the leader has to satisfy a constraint that depends on the follower's decision.

It is worthwhile noting that the LL subproblem may have multiple solutions for every (or some) fixed value of the UL decision making variable $\mathbf{x}$. When the solution of the LL subproblem is not unique, it is difficult for the leader to predict which point in $\mathcal{S}(\mathbf{x})$ the follower will choose (see Fig. 1 (b) for example).

## 3 Understanding and Modeling Practical Problems by BLOs

In this section, we demonstrate that even with different motivations and mechanisms, a variety of modern complex learning and vision tasks (e.g, hyper-parameter optimization, multi-task and meta learning, neural architecture search, adversarial learning, deep reinforcement learning and so on) actually share close relationships from the BLO perspective. Moreover, we provide a uniform BLO expression to (re)formulate all these problems. Table 2 provides a summary of learning and vision applications, which can be understood and modeled by BLO.

### 3.1 Hyper-parameter Optimization

Hyper-parameter Optimization (HO) refers to the problem of identifying the optimal set of hyper-parameters that can't be learned using the training data alone. Early in learning and vision areas, designing regularized models or support vector machines are generally the recommended approaches of selecting hyper-parameters [156]. Based on the representation of the hierarchical structure, these approaches are first expressed as a BLO problem and then transformed into the single-level optimization problem by replacing the LL subproblem with its optimality condition [157]. Due to the high computational cost, especially in high-dimensional hyper-parameter space, these original methods even could not guarantee a local optimal solution [158].

In recent years, gradient-based HO methods with deep neural networks have received extensive attention, which are generally divided into two categories: iterative differentiation (i.e., [7], [10], [11], [12], [13], [23], [71], [73], [75], [159]) and implicit differentiation (i.e., [8], [9], [72], [74], [135], [160], [161], [162], [163]), depending on how the gradient (w.r.t. hyper-parameters) can be computed. The former approximates the best-response function by performing several steps of gradient descent on the loss function, while the latter derives the hyper-gradients through the implicit function theory. One particular type of gradient-based HO is the data hyper-cleaning problem [13], [51], which generally trains a linear classifier with a cross-entropy function (w.r.t. parameters $\mathbf{y}$) and learns to optimize the hyper-parameters $\mathbf{x}$ with a $\ell_2$ regularization function.

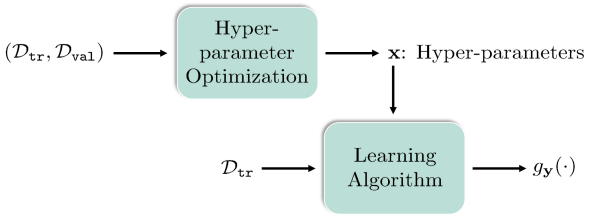



Fig. 2. Schematic diagram of HO. The UL subproblem involves optimization of hyper-parameters $\mathbf{x}$ based on $(\mathcal{D}_{\mathtt{tr}}, \mathcal{D}_{\mathtt{val}})$, while the LL subproblem involves optimization of weight parameters $\mathbf{y}$, aiming to find the learning algorithm $g_{\mathbf{y}}(\cdot)$ based on $\mathcal{D}_{\mathtt{tr}}$.

Indeed, HO can be understood as the most straightforward application of BLO in learning and vision fields [156]. Specifically, the UL objective $F(\mathbf{x}, \mathbf{y}; \mathcal{D}_{\mathtt{val}})$ aims to minimize the validation set loss with respect to the hyper-parameters (e.g. weight decay), and the LL objective $f(\mathbf{x}, \mathbf{y}; \mathcal{D}_{\mathtt{tr}})$ needs to output a learning algorithm by minimizing the training loss with respect to the model parameters (e.g. weights and biases). As illustrated in Fig. 2, the full dataset $\mathcal{D}$ is divided into the training and validation datasets (i.e., $\mathcal{D}_{\mathtt{tr}} \cup \mathcal{D}_{\mathtt{val}}$) and we instantiate how to model the HO task from the perspective of BLO. Inspired by this nested optimization, most HO applications can be characterized by the bi-level structure and formulated as the BLO problems. The UL subproblem involves the optimization of hyper-parameters $\mathbf{x}$ and the LL subproblem (w.r.t. weight parameters $\mathbf{y}$) aims to find the learning algorithm $g_{\mathbf{y}}(\cdot)$ by minimizing the training loss.

### 3.2 Multi-task and Meta Learning

The goal of meta learning (a.k.a., learning to learn) is to design models that can learn new skills or adapt to new environments rapidly with a few training examples (see Fig. 3 for a schematic diagram). As a variant of meta learning, multi-task learning just intends to jointly perform all the given tasks [164], [165]. One of the most well-known instances of meta learning is few-shot classification (i.e., $N$-way $M$-shot). Each task is a $N$-way classification designed to learn the meta-parameter with $M$ training samples selected from each of the class. Specially, the full meta training data set $\mathcal{D} = \{\mathcal{D}^j\}$ $(j = 1, \cdots, N)$ can be segmented into $\mathcal{D}^j = \mathcal{D}^j_{\mathtt{tr}} \cup \mathcal{D}^j_{\mathtt{val}}$, where $\mathcal{D}^j$ is linked to the $j$-th task.

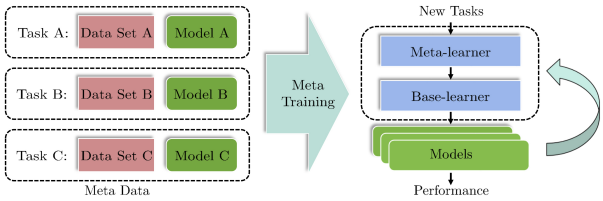



Fig. 3. Illustrating the training process of meta learning. The whole process is visualized to learn new tasks quickly by drawing upon related tasks on corresponding data sets. It can be decomposed into two parts: the "base-learner" trained for operating a given task and the "meta-learner" trained to learn how to optimize the base-learner.

According to the dependency between the meta-parameters and the network parameters, current meta learning based methods can be roughly categorized as two groups, i.e., meta-feature learning and meta-initialization learning, as can be seen in Fig. 4. Specifically, meta-initialization learning aims to investigate the meta information of multiple tasks by the network initialization, which can also be understood as the promotion of fine-tuning [84], [86]. From the BLO perspective, we actually formulate the network parameters and their initialization (based on multi-task information) by the LL and UL subproblems, respectively. In contrast, meta-feature learning methods first separate the network architecture as the meta feature extraction part and the task-specific part. Then they formulate a hierarchical learning process [11], [12], [56]. So in such tasks, we use the UL and LL subproblems to model the meta-feature part and the task-specific part, respectively.

#### 3.2.1 *Meta-feature Learning*

Meta-Feature Learning (MFL) aims to learn a sharing meta feature representation of all tasks. Recently, series of meta

TABLE 2
Summary of related learning and vision applications that can be (re)formulated as BLOs. The abbreviations are listed as follows: Hyper-parameter Optimization (HO), Meta-Feature Learning (MFL), Meta-Initialization Learning (MIL), Neural Architecture Search (NAS), Adversarial Learning (AL), and Deep Reinforcement Learning (DRL).

| Task | Important work | Other work |
|---|---|---|
| HO | [51] (ICML, 2017), [13] (AISTATS, 2019), [32] (ICML, 2020) | [69] (SIIMS, 2014), [70] (EURO, 2020), [71] (ICML, 2015), [72] (AISTATS, 2020), [73] (ICML, 2018), [74] (ICML, 2016), [75] (arXiv, 2019), [9] (ICLR, 2019), [53] (ICML, 2021), [76] (ICML, 2017), [77] (NIPS, 2020), [12] (ICML, 2018) |
| MFL | [12] (ICML, 2018), [56] (ICML, 2017) | [11] (arXiv, 2017), [78] (CVPR, 2018), [79] (CVPR, 2018), [80] (ICLR, 2018), [81] (ICLR, 2017), [82] (ICLR, 2018), [83] (ICML, 2019) |
| MIL | [84] (NIPS, 2019), [57] (ICLR, 2019), [85] (ICLR, 2019), [86] (NIPS, 2019) | [87] (ICLR, 2017), [55] (ICML, 2017), [88] (ICML, 2017), [89] (arXiv, 2018), [90] (arXiv, 2019), [91] (CVPR, 2020), [92] (AAAI, 2020), [93] (arXiv, 2018), [94] (NIPS, 2019), [95] (ICLR, 2020), [96] (ICML, 2019), [97] (ICML, 2018), [98] (CVPR, 2020), [99] (AAAI, 2020), [100] (ICASSP, 2020) |
| NAS | [14] (ICLR, 2019), [101] (NIPS, 2018), [58] (TGRS, 2020), [102] (CVPR, 2020), [59] (ICLR, 2019) | [103] (ICLR, 2019), [104] (CVPR, 2018), [16] (ICLR, 2019), [105] (ICCV, 2019), [106] (AISTATS, 2020), [107] (CVPR, 2020), [108] (AAAI, 2020), [109] (CVPR, 2020), [110] (CVPR, 2019), [111] (NIPS, 2019), [112] (ICCV, 2019), [113] (NIPS, 2019), [114] (CVPR, 2020), [115] (arXiv, 2019), [116] (SIGKDD, 2020), [117] (CVPR, 2020), [118] (CVPR, 2020), [119] (arXiv, 2020) |
| AL | [120] (arXiv, 2016), [21] (arXiv, 2016) | [121] (AAAI, 2020), [122] (CVPR, 2020), [18] (arXiv, 2018), [17] (PR, 2019), [123] (ICML, 2020), [60] (CVPR, 2020), [19] (CVPR, 2020), [124] (ICML, 2018) |
| DRL | [20] (AAAI, 2020), [21] (arXiv, 2016), [61] (ICML, 2020) | [125] (CIRED, 2019), [62] (arXiv, 2020), [126] (NIPS, 2019), [127] (ICML, 2020), [128] (AAMAS, 2020), [129] (arXiv, 2019), [130] (TSG, 2019), [131] (NeurIPS, 2016), [132] (NeurIPS, 2017), [133] (arXiv, 2018), [61] (ICML, 2020), [134] (ICML, 2019) |
| Others | [135] (SIIMS, 2013), [23] (SSVM, 2015), [136] (TIP, 2020), [137] (TNNLS, 2020), [138] (TIP, 2016) | [139] (ICML, 2016), [140] (ICLR, 2019), [25] (IJCAI, 2020), [141] (arXiv, 2019), [142] (UAI, 2020), [143] (arXiv, 2021), [144] (arXiv, 2020), [145] (TIP, 2020), [146] (arXiv, 2019), [147] (arXiv, 2020), [148] (T-RO, 2020), [149] (WACV, 2020), [150] (arXiv, 2020), [151] (NIPS, 2020), [152] (arXiv, 2020), [153] (arXiv, 2020), [154] (CVPR, 2020), [155] (ICLR, 2018) |

learning based approaches show that multi-task with hard parameter sharing and meta-feature representation are essentially similar [166], [167]. The optimization of meta-learner with respect to meta-parameters based on the UL subproblem is similar to HO [11], [12], [73]. The cross-entropy function $\ell(\mathbf{x}, \mathbf{y}^j; \mathcal{D}^j_{\mathtt{tr}})$ is actually considered as the task-specific loss for the $j$-th task on the meta training data set to define the LL objective.

As illustrated in the subfigure (a) of Fig. 4, following the bi-level framework, the network architecture in this category can be subdivided into two groups. The first is the cross-task intermediate representation layer parameterized by $\mathbf{x}$ (illustrated by the blue block), outputting the meta features. The second is the logistic regression layer parameterized by $\mathbf{y}^j$ (illustrated by the green block), as the ground classifier for the $j$-th task. As can be seen, the feature layers are shared across all episodes, while the softmax regression layer is episode (task) specific. We can also observe that the process of network forward propagation corresponds to the process of passing from the feature extraction part to the softmax part.

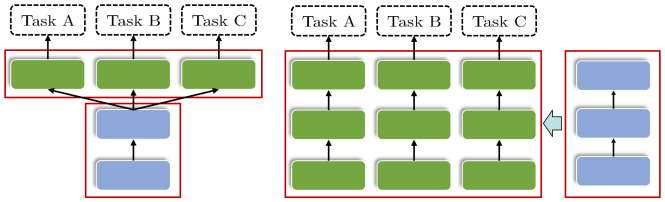
### 3.2.2 Meta-initialization Learning

Meta-Initialization Learning (MIL) aims to learn a meta initialization for all tasks. MAML [88], known for its simplicity, estimates initialization parameters with the cross-entropy and mean-squared error for supervised classification and regression tasks purely by the gradient-based search. Except for initial parameters, recent approaches have focused on learning other meta variables, such as updating strategies (e.g., descent direction and learning rate [87], [90], [168]) and an extra preconditioning matrix (i.e., [82], [97], [169]). Moreover, implicit gradient methods have a rapid development in the context of few-shot meta learning. There exist a large variety of algorithms replacing the gradient process of the optimization of base-learner through calculation of implicit meta gradient [84], [85], [94], [170]. Due to the large amount of computation required to calculate the Hessian vector product in the training process, various Hessian-free algorithms have been proposed to alleviate the costly computation of second-order derivatives, including but not limited to [54], [55], [56], [57], [93], [95]. In particular, various first-order approximation BLO algorithms have been proposed to avoid the time-consuming calculation of second-order derivatives in [89]. For instance, a modularized optimization library was proposed in [53] to unify several meta learning algorithms into a common BLO framework[1].

As can be shown in subfigure (b) of Fig. 4, $\mathbf{x}$ denoted by blue blocks corresponds to network initialization parameters, and $\mathbf{y}$ denoted by green blocks corresponds to model parameters and is treated as the updated variable satisfying the condition $\mathbf{y}^j_0 = \mathbf{x}$. Compared to MFL, there is

1. The code for this library is available at https://github.com/dut-media-lab/BOML.

no deeply intertwined and entangled relationship between two variables $(\mathbf{x}, \mathbf{y}^j)$, and $\mathbf{x}$ is only explicitly related to $\mathbf{y}$ in the initial state. As a bi-level coupled nested loop strategy, the LL subproblem based on base-learner is trained for operating a given task, and the UL subproblem based on meta-learner aims to learn how to optimize the base-learner. Among the well-known approaches in this direction, most recent approaches (i.e., [89], [171]) have claimed that the LL objective is denoted by the task-specific loss on the training data set, i.e., $f(\mathbf{x}, \{\mathbf{y}^j\}) = \ell(\mathbf{x}, \mathbf{y}^j; \mathcal{D}^j_{\mathtt{tr}})$. By utilizing cross-entropy function, the UL objective is given by $F(\mathbf{x}, \{\mathbf{y}^j\}) = \sum_j \ell(\mathbf{x}, \mathbf{y}^j; \mathcal{D}^j_{\mathtt{val}})$.




(a) MFL (b) MIL

Fig. 4. Illustration of two architectures that are generally applied to multi-task and meta learning: MFL and MIL. Both of them can be separated into two parts: meta-parameters denoted by $\mathbf{x}$ (blue blocks) and parameters denoted by $\mathbf{y}^j$ (green blocks). (a) shows meta-parameters for features shared across tasks and parameters of the logistic regression layer. (b) shows meta (initial) parameters shared across tasks and parameters of the task specific layer.

Both MFL and MIL are essential solution strategies of one optimizer based on another optimizer, thus conforming to the construction of the BLO scheme. As a task-specific loss associated with the $j$-th task, the LL objective can be defined as $\mathbf{y}^j \in \arg\min_{\mathbf{y}^j \in \mathcal{Y}} f\left(\mathbf{x}, \mathbf{y}^j; \mathcal{D}^j_{\mathtt{tr}}\right)$, $j = 1, \cdots, N$. Also, based on $\{\mathcal{D}^j_{\mathtt{val}}\}$, the UL objective can be given by $\min_{\mathbf{x} \in \mathcal{X}} F\left(\mathbf{x}, \{\mathbf{y}^j\}; \{\mathcal{D}^j_{\mathtt{val}}\}\right)$.

To summarize, the UL meta-learner performs gradient descent operations and updates the meta-parameter with feedback from base-learners to extract generalized meta knowledge. Subsequently, the better meta knowledge is fed into the base-learner (i.e., the LL subproblem) as part of its model for optimizing $\mathbf{y}$, thereby forming an optimization cycle.

### 3.3 Neural Architecture Search

Neural Architecture Search (NAS) seeks to automate the process of choosing the optimal neural network architecture [172]. Recently, there has aroused a great deal of interest in gradient-based differentiable NAS methods [14], [173], [174]. Specifically, these gradient-based differentiable NAS methods mainly contain three main concepts: search space, search strategy and performance estimation strategy. As shown in Fig. 5, by designing an architecture search space, they generally use a certain search strategy to find the optimal network architecture. Such a process can be regarded as the system of optimizing the operation and connection of each node.

DARTS [14], the most well-known instance, relaxed the search space to be continuous and conducted searching for architectures in a differentiable way to simultaneously

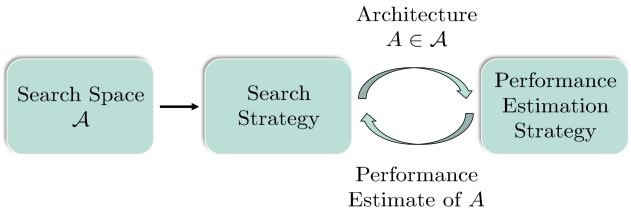



Fig. 5. Schematic diagram of NAS. Derived from a predefined search space $\mathcal{A}$, NAS first selects an architecture $A$ to transport into the performance estimation strategy, then returns the estimated performance of $A$ to the search strategy.

optimize the architectures and weights. Actually, each operation corresponds to a coefficient in DARTS. By denoting $\mathbf{x} = \{\mathbf{x}^{ij}\}$ as the architecture parameters and $\mathbf{x}^{ij}$ as the form of connection between two nodes, the expression formula of mixed operations $\bar{o}^{ij}(\cdot)$ based on the softmax function can be written as

$$\bar{o}^{ij}(\cdot) = \sum_{o \in \mathcal{O}} \frac{\exp(\mathbf{x}^{ij}_o)}{\sum\limits_{o' \in \mathcal{O}} \exp(\mathbf{x}^{ij}_{o'})} o(\cdot),$$

where $o$ and $o'$ are operations and $\mathcal{O}$ is the set of all candidate operations. Then, $o^{ij} = \arg\max_{o \in \mathcal{O}} \mathbf{x}^{ij}_o$ is further evaluated and performed in order to obtain the optimal architecture. However, due to the sharp deterioration in performance caused by the large number of skip connections, a great deal of improved approaches have emerged, such as ENAS [104], PC-DARTS [59], P-DARTS [105], just to name a few.

Currently, a series of gradient-based differentiable NAS methods combined with meta learning have been proposed, see [16], [107], [175], [176]. Based on the bi-level coupling mechanism, these gradient-based differentiable NAS methods have achieved promising results in the numerous visual and learning applications, such as image classification [58], semantic segmentation [110], [114], [177], object detection [102], [111], [112], [116], [117], medical image analysis [114], [177], video classification [139], recommendation system [119], graph network [115], [129] and representation learning [129], etc.

Given the proper search space, it is helpful for these gradient-based differentiable NAS methods to derive the optimal architecture for different vision and learning tasks. From the BLO's point of view, the UL objective w.r.t. the architecture weights (e.g. block/cell) can be parameterized by $\mathbf{x}$. And the LL objective w.r.t. the model weights can be parameterized by $\mathbf{y}$. Therefore, the full searching process can virtually be formulated as a BLO paradigm, where the UL objective is defined by $F(\mathbf{x}, \mathbf{y}; \mathcal{D}_{\mathtt{val}})$ based on the validation data set $\mathcal{D}_{\mathtt{val}}$, and the LL objective is given by $f(\mathbf{x}, \mathbf{y}; \mathcal{D}_{\mathtt{tr}})$ based on the training data set $\mathcal{D}_{\mathtt{tr}}$.

### 3.4 Adversarial Learning

Adversarial Learning (AL) is currently deemed as one of the most important learning tasks. It has been applied in a large variety of application areas, i.e., image generation [18], [60], [122], adversarial attacks [178] and face verification [17]. For example, the work proposed in [60] introduced an adaptive BLO model for image generation, which guided

the generator to reasonably modify the parameters in a complementary and promoting way. Moreover, a new adversarial training strategy has been proposed by learning a parametric optimizer with neural networks to study the adversarial attack [18]. As the current influential model, Generative Adversarial Network (GAN) can be deemed as deep generative models [179]. Recently, targeting at finding pure Nash equilibrium of generator and discriminator, the author proposed to exploit a fully differentiable search framework by formalizing as solving a bi-level mini-max optimization problem [19].

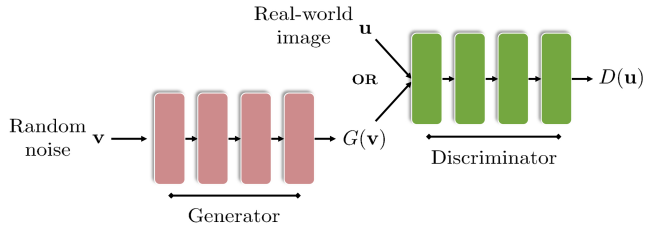



Fig. 6. Illustrating the architecture of GAN. The generator $G$ is represented as a deterministic feed forward neural network (red blocks), through which a fixed random noise $\mathbf{v}$ is passed to output $G(\mathbf{v})$. The discriminator $D$ is another neural network (green blocks) which maps the sampled real-world image $\mathbf{u} \sim p_{data}$ and $G(\mathbf{v})$ to a binary classification probability.

Most of the AL approaches can formulate the unsupervised learning problem as a bi-level game between two opponents: a generator which samples from a distribution, and a discriminator which classifies the samples as real or false, as shown in Fig. 6. The goal of GAN is to minimize the duality gap denoted by $\mathcal{V}(D, G)$:

$$\min_G \max_D \mathcal{V}(D,G) = \mathbb{E}_{\mathbf{u}\sim p_{data}(\mathbf{u})} \log D(\mathbf{u}) + \mathbb{E}_{\mathbf{v}\sim\mathcal{N}_{(0,1)}} \log(1 - D(G(\mathbf{v}))),$$

where the fixed random noise source $\mathbf{v}$ obtained from $\mathbf{v} \sim \mathcal{N}_{(0,1)}$ is input into the generator $G$, which, together with the sampled real-world image $\mathbf{u} \sim p_{data}$, is then authenticated by the discriminator $D$. Notice that $\mathbb{E}$ denotes the expectation which implies that the average value of some functions under a probability distribution.

Indeed, GAN problems generally correspond to the mini-max BLO problems, where the LL discriminator denoted by $f$ targets on learning a robust classifier, and the UL generator denoted by $F$ tries to generate the adversarial samples. Specifically, the UL and LL objectives can be respectively formulated as

$$F(\mathbf{x}, \mathbf{y}) = -\mathbb{E}_{\mathbf{v}\sim\mathcal{N}_{(0,1)}} \log(D(G(\mathbf{v}))),$$

$$f(\mathbf{x}, \mathbf{y}) = -\mathbb{E}_{\mathbf{u}\sim p_{data}(\mathbf{u})} \log D(\mathbf{u}) - \mathbb{E}_{\mathbf{v}\sim\mathcal{N}_{(0,1)}} \log(1 - D(G(\mathbf{v}))),$$

where $G$ and $D$ are parameterized with variables $\mathbf{x}$ and $\mathbf{y}$, respectively. In other words, the LL subproblem targets at maximizing the expected distance between characteristic functions of real and generated data distributions and the UL subproblem aims to maximizing the expected distance of the generated distribution.

### 3.5 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) aims to make optimal decisions by interacting with the environment and learning from the experiences. Indeed, a variety of DRL tasks, including Single-Agent Reinforcement Learning (SARL) [21], [22], [62], Multi-Agent Reinforcement Learning (MARL) [20], [125], [128], [180], Meta Reinforcement Learning (MRL) [61], [134], [181], [182], and Imitation Learning (IL) [131], [132], [133], which all can be modeled and tackled by BLO techniques.

As for SARL problems, Actor-Critic (AC) type methods have been widely studied and viewed as a bi-level or two-time-scale optimization problems [22], [62], as illustrated in Fig. 7. Indeed, AC type DRL methods often aim to simultaneously learn a state-action value-function $Q^\pi$ that predicts to expect the discounted cumulative reward and a policy which is optimal for that value function:

$$Q^\pi(s,a) = \mathbb{E}_{s_{i+j}\sim\mathcal{P}, r_{i+j}\sim\mathcal{R}, a_{i+j}\sim\pi}\left(\sum_{k=0}^{\infty} \gamma^j r_{i+j} | s_i = s, a_i = a\right),$$

where $\mathcal{P}$ and $\mathcal{R}$ denote dynamics of the environment and reward function, $s$ and $a$ are the state and action, $i$ and $j$ represent the i-th and j-th steps, and $\mathbb{E}$ is the expectation which implies that the average value of some function under a probability distribution. The policy maximizes the expected discounted cumulative reward for that state-action value-function, i.e., $\pi^* = \arg\max_\pi \mathbb{E}_{s_0\sim p_0, a_0\sim\pi}\left(Q^\pi(s_0, a_0)\right)$, where $s_0$, $a_0$ and $p_0$ correspond to the initial state, initial action and the initial state distribution, respectively. Under the BLO paradigm, the actor and critic correspond to the UL and LL variables, respectively. Let $\mathbf{x}$ denote the parameters of the state-action value-function and $\mathbf{y}$ denote the parameters of the policy $\pi$. The UL and LL objectives respectively take the form

$$F(\mathbf{x}, \mathbf{y}) = \mathbb{E} s_i, a_i \sim \pi(\mathtt{div}(\mathbb{E}_{s_{i+1}, a_{i+1}, r_{i+1}} (r_{i+1} + \gamma Q(s_{i+1}, a_{i+1})) \parallel Q(s_i, a_i))),$$

$$f(\mathbf{x}, \mathbf{y}) = -\mathbb{E}_{s_0\sim p_0, a_0\sim\pi} Q^\pi(s_0, a_0),$$

where $\mathtt{div}(\cdot||\cdot)$ represents any divergence.

MARL studies how multiple agents can collectively learn, collaborate, and interact with each other in an environment. In the classical MARL system, agents are treated equally and the goal is to solve the Markov game to an arbitrary Nash equilibrium when multiple equilibria exist, thus lacking a solution for selection. To address this issue, the work in [20] formulates MARL as the multi-state model-free Stackelberg equilibrium learning problem. Thus, under Markov games, they construct a BLO formulation to find Stackelberg equilibrium to address the MARL task. Similarly, a multi-agent bi-level cooperative reinforcement learning algorithm was proposed in [125] to solve the stochastic decision-making problem.

In recent years, MRL approaches (a.k.a., meta learning on reinforcement learning tasks), which aim to learn a policy that adapts fast to new tasks and/or environments, have achieved remarkable success [181], [183]. For example, the work in [182] learns a policy that can quickly adapt to other related models only with one policy gradient step. By adding control variables into gradient estimation, the work in [134] can obtain low variance estimates for policy gradients. While the work in [61] characterizes the optimality gap of the

stationary points attained by MAML for both reinforcement learning and supervised learning. Since all these works are based on the meta-initialization platform, it is also natural to formulate these meta reinforcement learning methods from the perspective of BLOs.

Generally, IL techniques are very useful when it is easier for an expert to demonstrate the desired behavior rather than to specify a reward function which would generate the same behavior or to directly learn the policy in DRL tasks [184]. In recent years, by connecting imitation learning with generative adversarial learning, a series of Generative Adversarial Imitation Learning (GAIL) techniques [131], [132], [133] have been investigated to imitate an expert in a model-free DRL scenario. Since GAIL type methods have a natural connection to the mechanism of GANs, we can definitely formulate these models using BLOs.

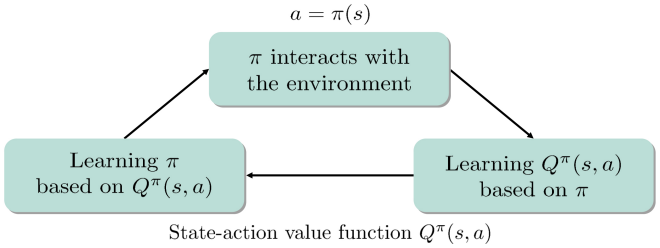



Fig. 7. Illustrating the schematic diagram of AC learning. First the actor $\pi$ interacts with the environment to learn the state-action value-function $Q^\pi(s,a)$, and then the actor $\pi$ is again obtained based on $Q^\pi(s,a)$.

### 3.6 Other Related Applications

The rapid development of deep learning has asserted its dominance in the field of image processing and analysis. In addition to the above mentioned tasks, there exist a significant amount of other related learning and vision tasks that can be re(formulated) as BLO problems, such as image enhancement [24], [135], [138], [141], [185], image registration [25], image-to-image translation [186], image recognition [187], image compression [188] and other related works [140], [142], [151]. For example, earlier work presented in [135] considered the parameter learning problem for image denoising models and incorporated a $p$-norm–based analytical prior. Under the BLO formulation, the LL subproblem is given by a variational model that consists of data fidelity and regularization terms, and the UL subproblem is expressed by the loss function. Furthermore, the work proposed in [138] formulated the discriminant dictionary learning method for image recognition tasks as a BLO. From this point of view, the UL subproblem can directly minimize the classification error, while the LL subproblem can use the sparsity term and the Laplacian term to characterize the intrinsic data structure.

By addressing a unified BLO problem, the LL subproblem is usually represented as a basic model that conforms to the laws or principles of physics, while the UL subproblem usually considers further constraints on the variables [23], [139].

## 4 Gradient-based BLOs

In past years, gradient-based techniques have become the most popular BLO solution strategies in learning and vision fields. In fact, one of the first gradient-based BLO methodology is [30]. Currently, a variety of explicit gradient-based methods have been investigated to solve BLOs [71], [73], [189]. Specifically, the works in [12], [51] first calculate gradient flow of the LL objective and then perform either reverse or forward gradient computations for the UL subproblem. Similar ideas have also been considered in [23], [75], [190], but with different specific implementations. On the other hand, there also exist some implicit gradient based methods [72], [84], [191] to use the implicit function theorem to obtain the gradient. In this section, we first review three categories of mainstream BLO formulations, which have been considered in various application scenarios. We then demonstrate how to uniformly reformulate these different BLOs from a single-level optimization perspective and investigate the intrinsic structures of existing gradient-based BLO algorithms within a unified algorithmic platform.

### 4.1 Different Formulations of BLO

It is worthwhile to notice that the original BLO model given in Eq. (2) is not clear in the case of the multiple LL optimal solutions for some of the selections of the UL decision maker [1]. Therefore, it is necessary to define, which solution out of the multiple LL solutions in $\mathcal{S}(\mathbf{x})$ should be considered. Here we actually consider three categories of viewpoints, i.e., singleton, optimistic and pessimistic BLOs.

The most straightforward idea in existing learning and vision literature is to assume that $\mathcal{S}(\mathbf{x})$ is a singleton. Formally, we call the BLO model is with the Lower-Level Singleton (LLS) condition if $\forall \mathbf{x} \in \mathcal{X}$, the solution set of the LL subproblem (i.e., $\mathcal{S}(\mathbf{x})$) is a singleton. In this case, we can simplify the original model as

$$\min_{\mathbf{x}\in\mathcal{X}} F(\mathbf{x},\mathbf{y}),\ s.t.\ \mathbf{y} = \arg\min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}). \tag{3}$$

Such singleton version of BLOs is well-defined and could cover a variety of learning and vision tasks (e.g., [7], [9], [71], [74], just name a few). Thus, in recent years, dozens of methods have been developed to address this nested optimization task in different application scenarios (see the following sections for more details).

Furthermore, the situation becomes more intricate if the LL subproblem is not uniquely solvable for each $\mathbf{x} \in \mathcal{X}$. Essentially, if the follower can be motivated to select an optimal solution in $\mathcal{S}(\mathbf{x})$ that is also best for the leader (i.e., with respect to $F$), it yields the so-called optimistic (strong) formulation of BLO

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ \min_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x},\mathbf{y}),\ s.t.\ \mathbf{y} \in \arg\min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}) \right\}. \tag{4}$$

The above stated optimistic viewpoint has drawn increasing attention in BLO literature [192], [193], [194] and recently also been investigated in learning and vision fields [32], [189], [195]. In Section 7, we will further explore how to solve such optimistic BLOs in detail.

If the leader does not have the information whether the follower returns the best response $\mathbf{y}$ from $\mathcal{S}(\mathbf{x})$ in terms of the UL objective $F$, then we have to assume that the follower is not cooperated with the leader. This is known as

the pessimistic (weak) formulation of BLO [196], [197] and can be given as:

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ \max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x},\mathbf{y}),\ s.t.\ \mathbf{y}\in \arg\min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}) \right\}. \tag{5}$$

It should be pointed out that till now we still lack efficient gradient-based algorithms to address the pessimistic BLO problems[2].

### 4.2 BR-based Single-Level Reformulation

In this work, we consider the optimal solution of the LL subproblem with a given UL variable $\mathbf{x}$ as the Best-Response (BR) of the follower (denoted as $\mathbf{y}^*(\mathbf{x})$). Then we can interpret BLO as a game process, in which the leader $\mathbf{x}$ considers what BR of the follower $\mathbf{y}$ is, i.e., how it will respond once it has observed the quantity of the leader [1], [198]. Based on the above understanding, we can reformulate the three different categories of BLOs as a unified single-level optimization problem.

Specifically, given the UL variable $\mathbf{x}$, we denote the corresponding BR mapping as $\mathbf{y}^*(\mathbf{x})$. In fact, if considering the singleton BLO, $\mathbf{y}^*(\mathbf{x})$ can be directly obtained by the unique LL solution. While for the optimistic and pessimistic BLOs, we actually first define their Inner Simple Bi-level (ISB) subproblems (w.r.t., $\mathbf{y}$)[3] as

$$\text{Optimistic ISB: } \min_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y}) \text{ and Pessimistic ISB: } \max_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y}). \tag{6}$$

Then by defining the solution set of ISB as $\widetilde{\mathcal{S}}(\mathbf{x})$, we could consider any $\mathbf{y}^*(\mathbf{x}) \in \widetilde{\mathcal{S}}(\mathbf{x})$ as the BR mapping, because points in $\widetilde{\mathcal{S}}(\mathbf{x})$ all obtain the minimum/maximum of $F(\mathbf{x},\mathbf{y})$ in $\mathcal{S}(\mathbf{x})$. Therefore, we can formulate the general BR mapping for different categories of BLOs as follows:

$$\begin{cases} \mathbf{y}^*(\mathbf{x}) := \arg\min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x},\mathbf{y}), & \text{Singleton}, \\ \mathbf{y}^*(\mathbf{x}) \in \widetilde{\mathcal{S}}(\mathbf{x}) := \begin{cases} \arg\min_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y}), & \text{Optimistic}, \\ \arg\max_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y}), & \text{Pessimistic}. \end{cases} \end{cases} \tag{7}$$

Based on Eq. (7), we actually obtain the following value-function-based reformulation (a single-level optimization model) for BLOs stated in Eq. (2), i.e.,

$$\min_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x}) := F(\mathbf{x},\mathbf{y}^*(\mathbf{x})), \tag{8}$$

in which $\varphi(\mathbf{x})$ actually can be used to uniformly represent the UL value-function of $F$ from the singleton, optimistic (i.e., $\inf_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y})$) and pessimistic (i.e., $\sup_{\mathbf{y}\in\mathcal{S}(\mathbf{x})} F(\mathbf{x},\mathbf{y})$) viewpoints.

2. In Section 9, we will demonstrate that we can also obtain some practical gradient-based iteration scheme within our general algorithmic platform for the pessimistic formulation of BLO.

3. It is known that the simple bi-level optimization is just a specific BLO problem with only one variable [32], [199].

### 4.3 A Unified Platform for Gradient-based BLOs

Moving one step forward, the gradient of $\varphi$ w.r.t. the UL variable $\mathbf{x}$ can be written as[4]

$$\underbrace{\frac{\partial \varphi(\mathbf{x})}{\partial \mathbf{x}}}_{\text{grad. of } \mathbf{x}} = \underbrace{\frac{\partial F(\mathbf{x},\mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{x}}}_{\text{direct grad. of } \mathbf{x}} + \underbrace{\mathbf{G}(\mathbf{x})}_{\text{indirect grad. of } \mathbf{x}}, \tag{9}$$

where the indirect gradient $G(\mathbf{x})$ can be further specified as the following two components:

$$\mathbf{G}(\mathbf{x}) = \underbrace{\left(\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}'}\right)'}_{\text{BR Jacobian}} \underbrace{\frac{\partial F(\mathbf{x},\mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}}}_{\text{direct grad. of } \mathbf{y}}. \tag{10}$$

Here we use "grad." as the abbreviation of gradient and denote the transpose operation as $(\cdot)'$. Note that, $\mathbf{y}^*(\mathbf{x})$ as a general mapping, can be given specific constraints and necessary assumptions to fit their particular requirements for these specific gradient-based BLO approaches in order to obtain different iteration formats and theoretical properties. For details, please refer to the following contents. In fact, by simple computation, the direct gradient is easy to obtain. However, the indirect gradient is intractable to obtain because we must compute the changing rate of the optimal LL solution with respect to the UL variable (i.e., the BR Jacobian $\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}}$). Please notice that we will also call $\frac{\partial \varphi(\mathbf{x}^k)}{\partial \mathbf{x}^k}$ as the practical BR Jacobian w.r.t. $\mathbf{x}^k$ in the following statements. The computation of the indirect gradient $\mathbf{G}(\mathbf{x})$ naturally motives the formulation of $\mathbf{y}^*(\mathbf{x})$ to obtain $\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}}$. For this purpose, a series of techniques have recently been developed from either explicit or implicit perspectives, which obtain their optimal solutions by recurrent differentiation through dynamic system and based on implicit differentiation theory, respectively.

Now we demonstrate how to formulate various existing gradient-based BLOs from a unified algorithmic platform. We first summarize a general BLO updating scheme in Alg. 1. It can be seen that the key component of this algorithm is to calculate the BR Jacobian. Then with $\frac{\partial \varphi(\mathbf{x}^k)}{\partial \mathbf{x}^k}$, we can just perform standard (stochastic) gradient descent schemes to update $\mathbf{x}^k$. Based upon our general algorithmic platform, we can observe that the main differences of these existing BLO approaches are just their specific strategies for calculating Jacobian of the BR mapping under different conditions (i.e., w/ LLS and w/o LLS).

In Fig. 8, we summarize mainstream gradient-based BLOs and illustrate their intrinsic relationships within our general algorithmic platform. It can be observed that in the LLS scenario, from the BR-based perspective, existing gradient methods can be categorized as two groups: Explicit Gradient for Best-Response (EGBR, stated in Section 5) and Implicit Gradient for Best-Response (IGBR, stated in Section 6). As for EGBR, there are mainly three types of methods, namely, recurrence-based EGBR (e.g., [12], [13], [14], [51], [71]), initialization-based EGBR (e.g., [89], [93] ) and proxy-based EGBR methods (e.g., [81], [86], [169], [171]), differing from each other in the way of formulating the BR mapping. For

4. Please notice that we actually do not distinguish between the operation of the derivatives and partial derivatives to simplify our presentation.

**Algorithm 1** A General Gradient-based BLO Scheme

**Input:** The UL and LL initialization.
**Output:** The optimal UL and LL solutions.
1: **for** $k = 1, \cdots, K$ **do**
2: Calculate the BR Jacobian $\frac{\partial \varphi(\mathbf{x}^k)}{\partial \mathbf{x}^k}$.
% (Mainstream calculation strategies are summarized in Figs. 8-9 and thoroughly surveyed in the following sections)
3: Perform (stochastic) gradient descent to update $\mathbf{x}^k$.
% (based on $\frac{\partial \varphi(\mathbf{x}^k)}{\partial \mathbf{x}^k}$)
4: **end for**

IGBR, existing works consider two groups of techniques (e.g., linear system [74], [84] and Neumann series [72]) to alleviate the computational complexity issue for the BR Jacobian. We emphasize that the validity of the above BLO methodologies must depend on the singleton of their LL solution set. When solving BLOs without the LLS assumption, recent works in [32], [189] have demonstrated that we need to first construct BR mapping based on both UL and LL subproblems, and then solve two optimization subproblems, namely, the single-level optimization subproblem (w.r.t. $\mathbf{x}$) and the ISB subproblem (w.r.t. $\mathbf{y}$). While the work in [195] has introduced a series of barrier functions and utilized interior point methods to obtain the BR mapping for each given $\mathbf{x}$.

To end up this section, we plot Fig. 9 to illustrate the optimization processes of existing mainstream gradient-based BLO methods from the BR mapping perspective and within our unified algorithmic platform. In the following (i.e., Sections 5-7), we will thoroughly survey these different categories of gradient-based BLO algorithms (including their acceleration, simplification and extension techniques) and their theoretical properties (convergence behaviors and computational complexity), accordingly.

## 5 Explicit Gradient for Best-Response

With the LLS condition, we delve deep into the EGBR category of methods, which aims to perform automatic differentiation through the LL dynamic system [200], [201] to solve the BLO problem. Specifically, given an initialization $\mathbf{y}_0 = \Psi_0(\mathbf{x})$ at $t = 0$, the iteration process of EGBRs can be generally written as

$$\mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1}; \mathbf{x}),\ t = 1, \cdots, T, \tag{11}$$

where $\Psi_t$ denotes some given updating scheme (based on the LL subproblem) at $t$-th stage and $T$ denotes the overall LL iterations number. For example, we can formulate $\Psi_t$ based on the gradient descent rule, i.e.,

$$\Psi_t(\mathbf{y}_{t-1}; \mathbf{x}) = \mathbf{y}_{t-1} - \eta_t \mathbf{d}_f(\mathbf{y}_{t-1}, \mathbf{x}), \tag{12}$$

where $\mathbf{d}_f(\mathbf{y}_{t-1}, \mathbf{x})$ is the descent mapping of $f$ at $t$-th stage (e.g., $\mathbf{d}_f(\mathbf{y}_{t-1}, \mathbf{x}) = \frac{\partial f(\mathbf{x}, \mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}$) and $\eta_t$ denotes the corresponding step size. Then we can calculate $\frac{\partial \varphi(\mathbf{x}^k)}{\partial \mathbf{x}^k}$ by substituting $\mathbf{y}_T := \Psi(\mathbf{x})$ approximately for $\mathbf{y}^*(\mathbf{x})$, and the full dynamic system can be defined as

$$\Psi(\mathbf{x}) := \Psi_T \circ \cdots \circ \Psi_1 \circ \Psi_0(\mathbf{x}). \tag{13}$$

Here the notation $\circ$ represents the compound dynamic operation of the entire iteration. That is, we actually consider the following optimization model

$$\min_{\mathbf{x} \in \mathcal{X}} \varphi_T(\mathbf{x}) := F(\mathbf{x}, \mathbf{y}_T(\mathbf{x})), \tag{14}$$

and need to calculate $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}}$ (instead of Eq. (9)) in the practical optimization scenario. Since it should be noted that $\Psi$ actually obtains an explicit gradient for best-response of the follower, we call this category of gradient-based BLOs as EGBR approaches hereafter. Starting from the Eq. (11), it is obvious to notice that $\mathbf{y}_t$ may be affected by the coupling of the variable $\mathbf{x}$ throughout the iteration. This coupling relationship will have a direct impact on the optimization process of the UL variable in Eq. (9). In fact, existing EGBR algorithms can be summarized from three perspectives. The first is that, if $\mathbf{x}$ closely acts on $\mathbf{y}_t$ during the whole iteration process, the subsequent optimization of variable $\mathbf{x}$ will be carried out recursively. The second is that when $\mathbf{x}$ only acts in the initial step, the subsequent optimization of variable $\mathbf{x}$ will be simplified. The third class is to replace the whole iterative process with a hyper-network, so as to efficiently approximate the BR mapping. Ultimately, in such cases, we divide them into three categories in terms of the coupling dependence of the two variables and the solution procedures, namely recurrence-based EGBR (stated in Section 5.1), initialization-based EGBR (stated in Section 5.2) and proxy-based EGBR (stated in Section 5.3).

### 5.1 Recurrence-based EGBR

It can be seen from Eq. (11) that all the LL iterative variables $\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_T$ depend on $\mathbf{x}$, and $\mathbf{x}$ serves as a recursive variable for the dynamic system. One of the most well-known approaches for calculating $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}}$ (with the above recurrent structure) is Automatic Differentiation (AD) [159], [202], which is also known as algorithmic differentiation or simply of "AutoDiff". There exist two diametrically opposed approaches for computing the gradient of recurrent neural networks, one of which corresponds to backward propagation through time in a reverse-mode way [203], [204], and the other corresponds to real-time recurrent learning in a forward-mode way [205], [206]. Since then quite a number of methods, closely related to this subject, have been proposed [12], [13], [51], [71]. Here we would like to review recurrence-based BR methods, covering forward-mode, reverse-mode AD, truncated and one-stage simplifications.

**Forward-mode AD (FAD):** To compute $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}}$, FAD appeals to the chain rule for the derivative of the dynamic system [51]. Specifically, recalling that $\mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1}, \mathbf{x})$, we have that the operation $\Psi_t$ indeed depends on $\mathbf{x}$ both directly by its expression and indirectly through $\mathbf{y}_{t-1}$. Hence, by drawing upon the chain rule, the formulation is given as[5]

$$\frac{\partial \mathbf{y}_t}{\partial \mathbf{x}} = \frac{\partial \Psi_t(\mathbf{y}_{t-1}; \mathbf{x})}{\partial \mathbf{y}_{t-1}} \frac{\partial \mathbf{y}_{t-1}}{\partial \mathbf{x}} + \frac{\partial \Psi_t(\mathbf{y}_{t-1}; \mathbf{x})}{\partial \mathbf{x}}. \tag{15}$$

To simplify the notation, we denote $\mathbf{Z}_t = \frac{\partial \mathbf{y}_t}{\partial \mathbf{x}}$, $\mathbf{A}_t = \frac{\partial \Psi_t(\mathbf{y}_{t-1}; \mathbf{x})}{\partial \mathbf{y}_{t-1}}$, $\mathbf{B}_t = \frac{\partial \Psi_t(\mathbf{y}_{t-1}; \mathbf{x})}{\partial \mathbf{x}}$ for $t > 0$ and $\mathbf{Z}_0 = \mathbf{B}_0 =$

5. Please notice that here we actually require $\mathbf{y}_t(\mathbf{x})$ to be a continuously differentiable function (w.r.t. $\mathbf{x}$) for all $t = 1, \cdots, T$. In existing EGBRs, they just introduce differentiable $\Psi_t$ to meet this requirement.

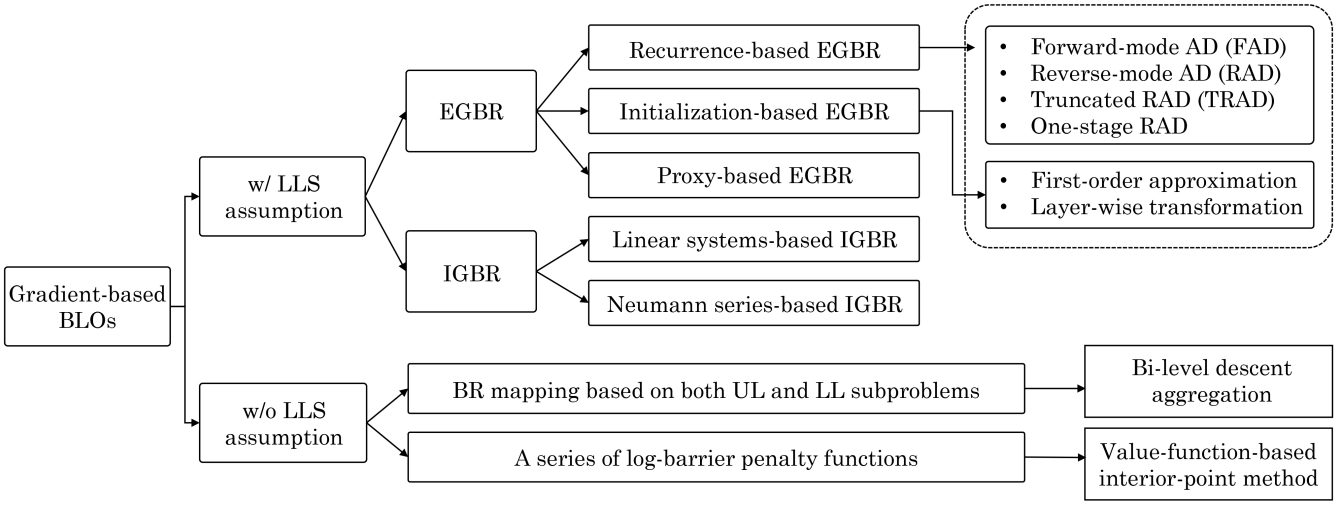



Fig. 8. Summary of the mainstream gradient-based BLOs. We categorize these existing approaches into two main groups, i.e., w/ and w/o LLS assumptions. When solving BLOs with LLS assumption, these methods can be further divided into two categories: EGBR and IGBR. As for EGBRs, they can be solved by different AD techniques (as denoted by the dashed rectangle). Very recently, two algorithms have also been proposed to address BLOs without the LLS assumption. In particular, they actually introduce a bi-level gradient aggregation or a value-function-based interior-point method to calculate the indirect gradient.

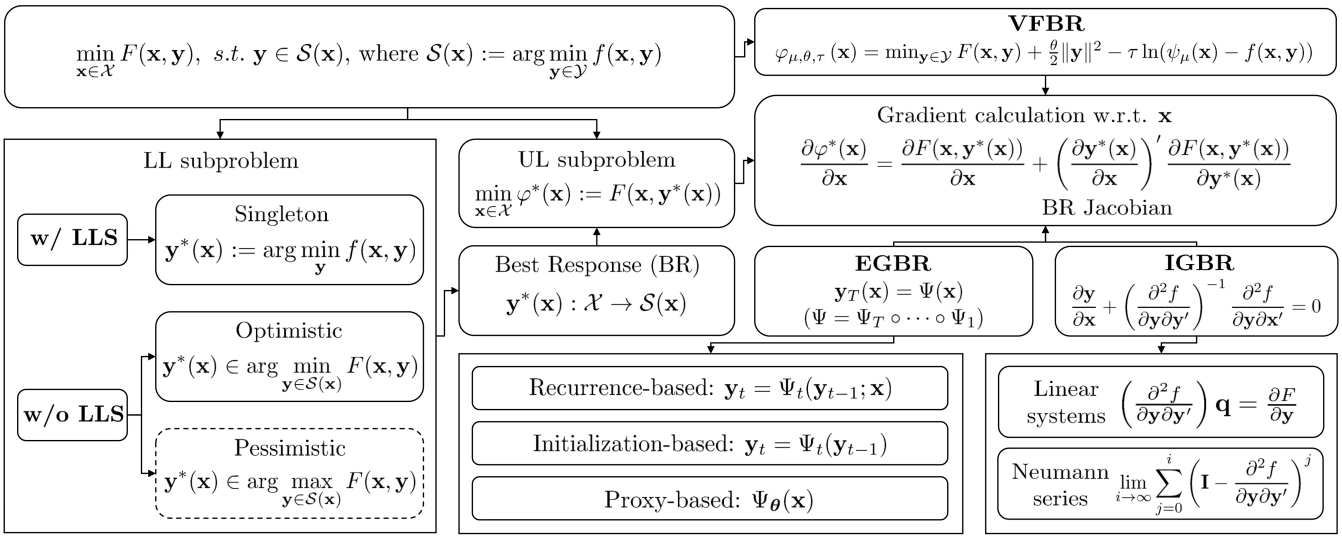



Fig. 9. Illustrating the roadmap of different categories of gradient-based BLOs. In the left bottom region, the formulations in the solid rectangles (i.e., singleton and optimistic) have been widely studied. In contrast, since gradient-based methods for pessimistic BLOs have not been properly investigated in existing literature, we denote this category of formulation by a dashed rectangle. In Section 9, we demonstrate that we can also obtain a practical pessimistic BLO scheme within our general algorithmic platform.

$\frac{\partial\Psi_0(\mathbf{x})}{\partial\mathbf{x}}$. Then we can rewrite Eq. (15) as $\mathbf{Z}_t = \mathbf{A}_t\mathbf{Z}_{t-1} + \mathbf{B}_t$ $(t = 1, \cdots, T)$. In this way, we have the following formulation to approximate the BR Jacobian

$$\frac{\partial\mathbf{y}_T(\mathbf{x})}{\partial\mathbf{x}} = \mathbf{Z}_T = \sum_{t=0}^{T}\left(\prod_{i=t+1}^{T}\mathbf{A}_i\right)\mathbf{B}_t. \tag{16}$$

Based on the above derivation, it is apparent that $\frac{\partial\varphi_T(\mathbf{x})}{\partial\mathbf{x}}$ can be computed by an iterative algorithm summarized in Alg. 2. Actually, FAD allows the program to update parameters after each step, which may significantly speed up the dynamic iterator and take up fewer memory resources when the number of hyper-parameters is much smaller than the number of parameters. It can be time-prohibitive for many hyper-parameters in a more efficient and convenient way.

**Algorithm 2** Forward-mode AD (FAD)

**Input:** The UL variable at the current stage $\mathbf{x}$ and the LL initialization $\mathbf{y}_0$.
**Output:** The gradient of $\varphi_T$ with respect to $\mathbf{x}$, i.e., $\frac{\partial\varphi_T}{\partial\mathbf{x}}$.

$\mathbf{Z}_0 = \frac{\partial\Psi_0(\mathbf{x})}{\partial\mathbf{x}}$.
**for** $t = 1, \cdots, T$ **do**
  $\mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1};\mathbf{x})$.
  $\mathbf{Z}_t = \mathbf{A}_t\mathbf{Z}_{t-1} + \mathbf{B}_t$.
**end for**
**return** $\frac{\partial F(\mathbf{x},\mathbf{y}_T)}{\partial\mathbf{x}} + \mathbf{Z}_T'\frac{\partial F(\mathbf{x},\mathbf{y}_T)}{\partial\mathbf{y}_T}$.

**Reverse-mode AD (RAD):** RAD is a generalization of the backward propagation algorithm and based on a Lagrangian formulation associated with the parameter optimization dynamics. By replacing $\mathbf{y}^*(\mathbf{x})$ by $\mathbf{y}_T$ and incorporating Eq. (16) into Eq. (9), a series of RAD works (e.g., [12], [51], [71]) derived

$$\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{x}} + \mathbf{Z}_T' \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}_T}. \tag{17}$$

Rather than calculating $\mathbf{Z}_T$ by forward propagation as that in FAD (i.e., Alg. 2), the computation of Eq. (17) can also be implemented by backward propagation. That is, we first define $\mathbf{g}_T = \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{x}}$ and $\boldsymbol{\lambda}_T = \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}_T}$. Then we update $\mathbf{g}_{t-1} = \mathbf{g}_t + \mathbf{B}_t' \boldsymbol{\lambda}_t$, and $\boldsymbol{\lambda}_{t-1} = \mathbf{A}_t' \boldsymbol{\lambda}_t$, with $t = T, \cdots, 0$. Finally, we have that $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{g}_{-1}$. Indeed, the above RAD calculation is structurally identical to backward propagation through time [51]. Moreover, we can also derive it following the classical Lagrangian approach. That is, we reformulate Eq. (14) as the following constrained model

$$\min_{\mathbf{x} \in \mathcal{X}} \varphi_T(\mathbf{x}) \;\; s.t. \;\; \begin{cases} \mathbf{y}_0 = \Psi_0(\mathbf{x}), \\ \mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1}; \mathbf{x}), \; t = 1, \cdots, T. \end{cases} \tag{18}$$

The corresponding Lagrangian function can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \{\mathbf{y}_t\}, \{\boldsymbol{\lambda}_t\}) &= \varphi_T(\mathbf{x}) + \boldsymbol{\lambda}_0' \left( \Psi_0(\mathbf{x}) - \mathbf{y}_0 \right) \\ &+ \sum_{t=1}^{T} \boldsymbol{\lambda}_t' \left( \Psi_t(\mathbf{y}_{t-1}; \mathbf{x}) - \mathbf{y}_t \right), \end{aligned} \tag{19}$$

where $\boldsymbol{\lambda}_t$ denotes the Lagrange multiplier associated with the $t$-th stage of the dynamic system. The KKT optimality condition of Eq. (18) is obtained by setting all derivatives of $\mathcal{L}$ to zero, satisfying the condition that $\mathbf{y}_t(\mathbf{x})$ is a continuously differentiable function w.r.t. $\mathbf{x}$ for the case that $t = 1, \cdots, T$. Then by some simple algebras, we have $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}}$. Overall, we present the RAD algorithm in Alg. 3.

**Algorithm 3** Reverse-mode AD (RAD)

**Input:** The UL variable at the current stage $\mathbf{x}$ and the LL initialization $\mathbf{y}_0$.
**Output:** The gradient of $\varphi_T$ with respect to $\mathbf{x}$, i.e., $\frac{\partial \varphi_T}{\partial \mathbf{x}}$.

$\mathbf{y}_0 = \Psi_0(\mathbf{x})$.
**for** $t = 1, \cdots, T$ **do**
  $\mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1}; \mathbf{x})$.
**end for**
$\mathbf{g}_T = \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{x}}$ and $\boldsymbol{\lambda}_T = \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{y}_T}$.
**for** $t = T, \cdots, 0$ **do**
  $\mathbf{g}_{t-1} = \mathbf{g}_t + \mathbf{B}_t' \boldsymbol{\lambda}_t$ and $\boldsymbol{\lambda}_{t-1} = \mathbf{A}_t' \boldsymbol{\lambda}_t$.
**end for**
**return** $\mathbf{g}_{-1}$.

**Truncated RAD (TRAD):** The above two precise calculation methods in many practical applications are tedious and time-consuming with full backward propagation training. As aforementioned, due to the complicated long-term dependencies of the UL subproblem on $\mathbf{y}_T(\mathbf{x})$, calculating Eq. (17) in RAD is a challenging task. This difficulty is further aggravated when both $\mathbf{x}$ and $\mathbf{y}$ are high-dimensional vectors. More recently, the truncation idea has been revisited to address the above issue and shows competitive performance with significantly less computation time and memory [13], [207], [208]. Specifically, by ignoring the long-term dependencies and approximating Eq. (17) with partial sums (i.e., storing only the last $M$ iterations), we have

$$\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} \approx \mathbf{g}_{T-M} := \frac{\partial F(\mathbf{x}, \mathbf{y}_T)}{\partial \mathbf{x}} + \mathbf{Z}_{T-M}' \frac{\partial F(\mathbf{x}, \mathbf{y}_T(\mathbf{x}))}{\partial \mathbf{y}_T}, \tag{20}$$

where $\mathbf{Z}_{T-M} = \sum_{t=T-M+1}^{T} \left( \prod_{i=t+1}^{T} \mathbf{A}_i \right) \mathbf{B}_t$. It can be seen that ignoring the long-term dependencies can greatly reduce the time and space complexity for computing the approximate gradients. Recently, the work in [13] has investigated the theoretical properties of the above truncated RAD scheme, and confirmed this fact that using few-step backward propagation could perform comparably to optimization with the exact gradient, while requiring far less memory and half computation time.

**One-stage RAD:** Limited and expensive memory is often a bottleneck in modern massive-scale deep learning applications. For instance, multi-step iteration of the inner program will cause a lot of memory consumption [88]. Inspired by BLO, a variety of simplified and elegant techniques have been adopted to circumvent this issue. The work in [14] proposes another simplification of RAD, which considers a fixed initialization $\mathbf{y}_0$ and only performs one-step iteration in Eq. (11) to remove the recurrent structure for the gradient computation in Eq. (17), i.e.,

$$\frac{\partial \varphi_1(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}, \mathbf{y}_1(\mathbf{x}))}{\partial \mathbf{x}} + \left( \frac{\partial \mathbf{y}_1(\mathbf{x})}{\partial \mathbf{x}'} \right)' \frac{\partial F(\mathbf{x}, \mathbf{y}_1(\mathbf{x}))}{\partial \mathbf{y}_1(\mathbf{x})}. \tag{21}$$

By formulating the dynamic system like that in Eq. (12), we then write $\frac{\partial \mathbf{y}_1(\mathbf{x})}{\partial \mathbf{x}}$ as

$$\frac{\partial \mathbf{y}_1}{\partial \mathbf{x}'} = \frac{\partial \left( \mathbf{y}_0 - \frac{\partial f(\mathbf{x}, \mathbf{y}_0)}{\partial \mathbf{y}_0} \right)}{\partial \mathbf{x}'} = -\frac{\partial^2 f(\mathbf{x}, \mathbf{y}_0)}{\partial \mathbf{y}_0 \partial \mathbf{x}'}. \tag{22}$$

Since calculating Hessian in Eq. (22) is still time-consuming, to further simplify the calculation, we can adopt finite approximation [14] to cancel the calculation of the Hessian matrix (e.g., central difference approximation). The specific derivation can be formalized as follows:

$$\frac{\partial F(\mathbf{x}, \mathbf{y}_1)}{\partial \mathbf{y}_1} \frac{\partial^2 f(\mathbf{x}, \mathbf{y}_0)}{\partial \mathbf{y}_0 \partial \mathbf{x}'} \approx \frac{\frac{\partial f(\mathbf{x}, \mathbf{y}_0^+)}{\partial \mathbf{x}} - \frac{\partial f(\mathbf{x}, \mathbf{y}_0^-)}{\partial \mathbf{x}}}{2\epsilon}, \tag{23}$$

in which $\mathbf{y}_0^{\pm} = \mathbf{y}_0 \pm \epsilon \frac{\partial F(\mathbf{x}, \mathbf{y}_1)}{\partial \mathbf{y}_1}$, and $\epsilon$ is set to be a small scalar [59].

### 5.2 Initialization-based EGBR

The research community has started moving towards the challenging goal of building general purpose initialization-based optimization systems whose ability to learn the initial parameters is better. Regardless of the recurrent structure, we need to consider the special setting to analyze a family of algorithms for learning the initialization parameters, named initialization-based EGBR methods. In this series, MAML [88] is considered as the most representative and important work. By making more practical assumptions about the coupling dependence of two variables, these methods no longer use the full dynamic system to explicitly and accurately describe the dependency between $\mathbf{x}$ and $\mathbf{y}$ as discussed above in Eq. (18), but adopt a further simplified paradigm.

Specifically, by treating the iterative dynamic system with only the first step that $\mathbf{y}$ is explicitly related to $\mathbf{x}$, this process can be formulated as

$$\min_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \;\; s.t. \;\; \begin{cases} \mathbf{y}_0 = \Psi_0(\mathbf{x}), \\ \mathbf{y}_t = \Psi_t(\mathbf{y}_{t-1}),\ t = 1, \cdots, T, \end{cases} \tag{24}$$

where $\mathbf{x}$ represents the network initialization parameters, and $\mathbf{y}_t$ represents the network parameters after performing some sort of update. Given initial condition $\Psi_0(\mathbf{x})$, then we obtain the following simplified formula

$$\mathbf{y}_T = \Psi_0(\mathbf{x}) - \sum_{t=1}^{T} \mathbf{d}_f(\mathbf{y}_{t-1}), \tag{25}$$

where $\mathbf{d}_f(\mathbf{y}_{t-1})$ is the descent mapping of $f$ at the $t$-th stage (e.g., $\mathbf{d}_f(\mathbf{y}_{t-1}) = \frac{\partial f(\mathbf{x},\mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}$). Finally, we have the Jacobian matrix as follows

$$\frac{\partial \mathbf{y}_T}{\partial \mathbf{x}} = \frac{\partial \left( \Psi_0(\mathbf{x}) - \sum_{t=1}^{T} \mathbf{d}_f(\mathbf{y}_{t-1}) \right)}{\partial \mathbf{x}}. \tag{26}$$

Then we have to calculate the Hessian matrix term $\frac{\partial^2 f}{\partial \mathbf{y}_{t-1} \partial \mathbf{x}'}$, which is time-consuming in real computation scenarios. To reduce the computational load, we will introduce two remarkably simple algorithms via a series of approximate transformation operations below. Among various schemes to simplify the algorithm based on initialization-based EGBR approaches, first-order approximation (e.g., [89], [93]) and layer-wise transformation (e.g., [81], [86], [169], [171]) are among the more popular. Very recently, the works in [209], [210] also consider the initialization as an auxiliary variable to improve the performance of RAD.

**First-order Approximation:** For example, the most representative algorithms (i.e., FOMAML [89] and Reptile [93]) adopt the operation of first-order approximation, a way to alleviate the Hessian term computation problem without sacrificing much performance. Specifically, this approximation ignores the second derivative term by removing the Hessian matrix $\frac{\partial^2 f}{\partial \mathbf{y}_{t-1} \partial \mathbf{x}'}$, thus simplifying the substitution of $\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}}$ performed by

$$\frac{\partial \varphi_T(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial F(\mathbf{x}, \mathbf{y}_T(\mathbf{x}))}{\partial \mathbf{x}} + \left( \frac{\partial \Psi_0(\mathbf{x})}{\partial \mathbf{x}'} \right)' \frac{\partial F(\mathbf{x}, \mathbf{y}_T(\mathbf{x}))}{\partial \mathbf{y}_T(\mathbf{x})}. \tag{27}$$

In addition, there is another first-order extension to simplify Eq. (26) via the operation of difference approximation [93]. Instead of avoiding the Hessian term directly, it tries to approximate $\frac{\partial \mathbf{y}_T}{\partial \mathbf{x}}$ (i.e.,$\mathbf{y}_T - \mathbf{x}$ and $(\mathbf{y}_T - \mathbf{x})/\alpha$) in another soft way, where $\alpha$ is the step size used for the gradient descent operation. Contrary to [89], this method offers to employ different linear combinations of all steps, as opposed to using only the last step. Overall, the above algorithm can significantly reduce the computational cost while maintaining roughly comparable performance.

**Layer-wise Transformation:** Indeed, there are also a series of learning-based BLOs related to layer-wise transformation, i.e., Meta-SGD [81], T-Net [171], Meta-Curvature [86] and WarpGrad [169]. In addition to initial parameters, this type of work focuses on learning some additional parameters (or transformation) at each layer of the network. From the above Eq. (25), it can be uniformly formulated as

$$\mathbf{y}_T = \Psi_0(\mathbf{x}) - \sum_{t=1}^{T} \mathbf{P}(\mathbf{y}_{t-1}, \boldsymbol{\omega}) \mathbf{d}_f(\mathbf{y}_{t-1}), \tag{28}$$

where $\mathbf{P}(\mathbf{y}_{t-1}, \boldsymbol{\omega})$ defines the matrix transformation learned at each layer and $\boldsymbol{\omega}$ is an auxiliary vector (e.g., learning rate). For example, Meta-SGD [81] learns a vector $\boldsymbol{\omega}$ of learning rates and $\mathbf{P}$ corresponded to $\texttt{diag}(\boldsymbol{\omega})$, and T-Net [171] aims to learn block-diagonal preconditioning linear projections. Similarly, an additional block-diagonal preconditioning transformation is also performed by Meta-Curvature [86]. WarpGrad [169] is closely related to the concurrent work of Meta-Curvature [86], which defines the preconditions gradient from a geometrical point of view and replaces the linear projection with a non-linear preconditioning matrix as a warp layer.

### 5.3 Proxy-based EGBR

Generally speaking, calculating the BR mapping (or BR Jacobian) is key to solve BLOs. Recently, several proxy-based EGBR methods (e.g., [9], [77], [163]) utilize the differentiable hyper-network (denoted as $\Psi_{\boldsymbol{\theta}}(\mathbf{x})$ with parameters $\boldsymbol{\theta}$) to substitute the dynamic system $\Psi(\mathbf{x})$ and then approximate the BR mapping[6], i.e.,

$$\Psi_{\boldsymbol{\theta}}(\mathbf{x}) \to \Psi(\mathbf{x}) \approx \mathbf{y}^*(\mathbf{x}). \tag{29}$$

Specifically, they train a hyper-network that takes hyper-parameters $\mathbf{x}$ as input and outputs the approximate optimal set of weights as the optimal solution of the LL subproblem.

In fact, both global and local proxy techniques have been considered to approximate the BR mapping. From the perspective of global approximation, first, if the distribution $p(\mathbf{x}) \subseteq \mathcal{X}$ is fixed, they learn $\boldsymbol{\theta}$ by minimizing $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} f(\mathbf{x}, \Psi_{\boldsymbol{\theta}}(\mathbf{x}))$, so that $\Psi_{\boldsymbol{\theta}}(\mathbf{x})$ can approximate the BR mapping in a neighborhood around the current $\mathbf{x}$, and second update $\mathbf{x}$ with $\Psi_{\boldsymbol{\theta}}$ as a proxy substituted into Eq. (14), i.e.,

$$\mathbf{x}^* \approx \arg\min_{\mathbf{x}\in\mathcal{X}} F(\mathbf{x}, \Psi_{\boldsymbol{\theta}}(\mathbf{x})). \tag{30}$$

For local approximation, by introducing a small UL disturbing term, they first minimize the objective $\mathbb{E}_{\epsilon\sim p(\epsilon|\delta)} f(\mathbf{x} + \epsilon, \Psi_{\boldsymbol{\theta}}(\mathbf{x}+\epsilon))$, where $\epsilon$ represents the perturbation noise added to $\mathbf{x}$, and $p(\epsilon|\delta)$ is defined as a factorized Gaussian noise distribution with a fixed scale parameter $\delta$. After that, the UL variable $\mathbf{x}$ is updated by minimizing the proxy function, i.e., Eq. (30).

In comparison to other types EGBRs, proxy-based EGBRs can easily replace existing modules in deep learning libraries with hyper-counterparts that accept an additional vector of UL variable as input and adapt online, thereby requiring less memory consumption to meet the performance requirements.

## 6 Implicit Gradient for Best-Response

In contrast to the EGBR methods surveyed above, IGBR methods in essence can be interpreted as introducing Implicit Function Theory (IFT) to derive BR Jacobian [211]. In

6. Note that, these methods assume $\mathbf{y}^*(\mathbf{x})$ is a continuously differentiable function and $\mathcal{X}$ and $\mathcal{Y}$ denote the whole space [9], [77], [163].

particular, IGBR type BLOs only rely on the solution to the LL optimization and can effectively decouple the UL gradient computation from the choice of LL optimizer. Indeed, the gradient-based BLO methodologies with implicit differentiation are radically different from EGBR methods, which have been extensively applied in a string of applications (e.g., [72], [84], [191]). As an example, a set of early IGBR approaches (e.g., [160], [161]) used implicit differentiation to select hyper-parameters of kernel-based models. Recently, IGBR type approaches have been applied in different application scenarios, such as learning hyper-parameter for neural networks [72] and variational models [135].

Now we demonstrate how to derive IGBRs to solve BLOs. Specifically, in the LLS optimization scenario, we first require that $f(\mathbf{x}, \mathbf{y})$ satisfies the smooth condition (or at least twice continuously differentiable) w.t.r. both the UL and LL variables, and $\mathbf{y}^*(\mathbf{x})$ is a continuously differentiable function w.r.t. $\mathbf{x}$. Then we can directly obtain the implicit gradient of $\mathbf{x}$ (i.e., $\mathbf{G}(\mathbf{x})$) based on the first-order optimality condition (i.e., $\frac{\partial f(\mathbf{x},\mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^*(\mathbf{x})} = 0$). That is, by deriving the above equation w.r.t. $\mathbf{x}$, we have that

$$\frac{\partial \mathbf{y}^*(\mathbf{x})}{\partial \mathbf{x}'} + \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^*(\mathbf{x}) \partial \mathbf{y}^*(\mathbf{x})'}\right)^{-1} \frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{\partial \mathbf{y}^*(\mathbf{x}) \partial \mathbf{x}'} = 0.$$

By further assuming that $\frac{\partial^2 f(\mathbf{x},\mathbf{y}^*(\mathbf{x})}{\partial \mathbf{y}^*(\mathbf{x}) \partial \mathbf{y}^*(\mathbf{x})'}$ is invertible, and drawing upon the chain rule, the indirect gradient $\mathbf{G}(\mathbf{x})$ can be obtained as follows:

$$\mathbf{G}(\mathbf{x}) = -\left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*)}{\partial \mathbf{y}^* \partial \mathbf{x}'}\right)' \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}^*)}{\partial \mathbf{y}^* \partial \mathbf{y}^{*\prime}}\right)^{-1} \frac{\partial F(\mathbf{x}, \mathbf{y}^*)}{\partial \mathbf{y}^*}. \quad (31)$$

Intuitively, Eq. (31) has offered the exact indirect gradient formulation but is generally calculated based on numerical approximations in practice. From a computational point of view, due to involving a large number of repeated product operations of Hessian-vector and Jacobian-vector, EGBRs based on high-dimensional data are usually computationally expensive and time-consuming. Thus a few implicit techniques, such as IGBR based on linear system [74], [84] and Neumann series [72], have been proposed to address this computational issue.

**Based on Linear System:** To calculate the Hessian matrix inverse more efficiently, it is generally assumed that solving linear systems is a common operation (e.g., HOAG [74], IMAML [84]). Specially, $(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'})^{-1} \frac{\partial F}{\partial \mathbf{y}}$ can be computed as the solution to the linear system $(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'})\mathbf{q} = \frac{\partial F}{\partial \mathbf{y}}$ for $\mathbf{q}$. Based on the above derivation, it is apparent that $\frac{\partial F}{\partial \mathbf{x}}$ can be directly computed by the algorithm summarized in Alg. 4.

**Based on Neumann Series:** Instead of solving the linear system, another type of IGBM (i.e., Neumann IFT [72]) method aims to calculate the Neumann series to approximate the inverse of Hessian matrix. Specifically, rather than solving the linear system in the second step of Alg. 4, the inverse Hessian is expressed as the following Neumann series:

$$\left(\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'}\right)^{-1} = \lim_{i \to \infty} \sum_{j=0}^{i} \left(\mathbf{I} - \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'}\right)^j,$$

where $\mathbf{I}$ denotes an identity matrix with proper size. If the operator $\mathbf{I} - \frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'}$ is contractive, it leverages that unrolling differentiation for $i$ steps around locally optimal weights $\mathbf{y}^*(\mathbf{x})$ is equivalent to approximating the inverse with the first $i$ terms in Neumann series. In this way, the entire computation can efficiently perform vector-Jacobian products, thus providing a cheap approximation to the inverse-Hessian-vector product.

**Algorithm 4** Implicit Gradient by Solving Linear System

**Input:** The UL variable at the current state, i.e., $\mathbf{x}$.
**Output:** The gradient of $F$ with respect to $\mathbf{x}$, i.e., $\frac{\partial F}{\partial \mathbf{x}}$.

1: Optimize the LL variable up to tolerance $\epsilon$. That is, find $\mathbf{y}_\varepsilon$ such that
$$\|\mathbf{y}^*(\mathbf{x}) - \mathbf{y}_\varepsilon\| \leq \epsilon.$$
2: Solve the linear system
$$\left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}_\varepsilon)}{\partial \mathbf{y}_\varepsilon \partial \mathbf{y}_\varepsilon'}\right) \mathbf{q} = \frac{\partial F(\mathbf{x}, \mathbf{y}_\varepsilon)}{\partial \mathbf{y}_\varepsilon},$$
for $\mathbf{q}$ up to the tolerance $\epsilon$, i.e., $\left\| \left(\frac{\partial^2 f}{\partial \mathbf{y}_\varepsilon \partial \mathbf{y}_\varepsilon'}\right) \mathbf{q} - \frac{\partial F}{\partial \mathbf{y}_\varepsilon} \right\| \leq \epsilon$.
3: Compute approximate gradient by
$$\mathbf{p} = \frac{\partial F(\mathbf{x}, \mathbf{y}_\varepsilon)}{\partial \mathbf{x}} - \left(\frac{\partial^2 f(\mathbf{x}, \mathbf{y}_\varepsilon)}{\partial \mathbf{y}_\varepsilon \partial \mathbf{x}'}\right)' \mathbf{q}.$$
4: **return** $\mathbf{p}$.

## 7 BLO BEYOND LOWER-LEVEL SINGLETON

As stated in the above Sections 4-6, different categories of gradient-based algorithms have been proposed to address BLOs. However, most of these approaches rely on the LLS assumption (i.e., the solution set of the LL subproblem is a singleton) stated in Section 4 to simplify their optimization process and theoretical analysis. That is to say, the sequence $\{\mathbf{y}_t\}_{t=0}^T$ generated by these mainstream methods could converge to the true optimal solution only if the LLS condition is satisfied. Unfortunately, it has been demonstrated that such LLS assumption is too restrictive to be satisfied in most real-world learning and vision applications. For example, the works in [32], [189] have designed a series of counter-examples to illustrate that these existing EGBRs cannot obtain the correct solution if the LLS assumption is not satisfied.

In this section, we review some recent works [32], [189], [195], which can efficiently address the LLS issue in the optimistic BLO scenario. The key optimization process of these works is to obtain the solution set of the ISB (i.e., Eq. (6)). That is, these works actually adopted different techniques, such as the UL and LL gradient aggregation [32], [189] and value-function-based interior-point method [195] to solve Eq. (6) for BLOs without the LLS condition.

**UL and LL Gradient Aggregation:** Differing from previous EGBR type methods which only rely on the gradient information of the LL subproblem to update $\mathbf{y}$, a more generalized EGBR type method, Bi-level Descent Aggregation (BDA) method [32], characterizes an aggregate computation of both the LL and the UL descent information. With a given UL variable $\mathbf{x}$, the aggregated descent direction w.r.t. the ISB subproblem (i.e., Eq. (6)) can be defined as

$$\mathbf{d}(\mathbf{y}_{t-1}; \mathbf{x}) = \rho_t \frac{\partial F(\mathbf{x}, \mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}} + (1 - \rho_t) \frac{\partial f(\mathbf{x}, \mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}, \quad (32)$$

where $\rho_t \in (0, 1]$ is the aggregation parameter (tending to zero [212], [213]), and $\frac{\partial F(\mathbf{x},\mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}$ (or $\frac{\partial f(\mathbf{x},\mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}$) stands for the descent directions of the UL (or LL) objectives.

**Value-Function-based Interior-point Method:** Different from EGBRs and IGBRs, a more recent Value-Function Best-Response (VFBR) type BLO methods reformulate BLO into a ISB optimization problem by the value function of the UL objective. After that, they further transform it into a single-level optimization problem with an inequality constraint through the value function of the LL objective. Recently, a typical VFBR work, named Bi-level Value-Function-based Interior-point Method (BVFIM) [195], has designed a log-barrier penalty-based single-level reformulation for Eq. (6) to address the LLS issue in the non-convex scenario. Specifically, BVFIM first reformulates the ISB subproblem in Eq. (6) as follows:

$$\min_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}, \mathbf{y}), \ \text{ s.t. } \ f(\mathbf{x}, \mathbf{y}) \leq \psi_\mu(\mathbf{x}), \tag{33}$$

where $\psi_\mu(\mathbf{x})$ is a regularized value function of the LL subproblem, i.e.,

$$\psi_\mu(\mathbf{x}) = \min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y}) + \frac{\mu_1}{2}\|\mathbf{y}\|^2 + \mu_2. \tag{34}$$

Here $\mu_1, \mu_2$ are two positive constants and we denote $\mu = (\mu_1, \mu_2)$. Then the relaxed inequality constraint $f(\mathbf{x}, \mathbf{y}) \leq \psi_\mu(\mathbf{x})$ is penalized to the objective by a log-barrier penalty and thus Eq. (33) can be approximated by

$$\varphi_{\mu,\theta,\tau}(\mathbf{x}) = \min_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}, \mathbf{y}) + \frac{\theta}{2}\|\mathbf{y}\|^2 - \tau \ln(\psi_\mu(\mathbf{x}) - f(\mathbf{x}, \mathbf{y})), \tag{35}$$

where $(\mu, \theta, \tau) > 0$. Finally, BVFIM proves that indirect gradient $\mathbf{G}(\mathbf{x})$ in Eq. (10) can be obtained by solving a series of Eq. (35) with decreasing parameters $(\mu, \theta, \tau)$ (tending to zero). It should be noticed that BVFIM can successfully avoid these time-consuming Hessian-vector and Jacobian-vector products, which are necessary for previous gradient-based BLOs. So this method is more suitable for BLO tasks with complex LL subproblems.

## 8 Theoretical Investigations

In addition to modeling various learning and vision applications from the perspective of BLO and establishing a general algorithmic framework to unify different categories of existing BLO algorithms, in this section, we further investigate some important theoretical issues of BLOs, including the convergence behaviors and computational complexity of gradient-based BLOs, which actually can provide us insights and guidance in practical application scenarios (e.g., adopt/design proper BLO methods).

### 8.1 Convergence Properties and Required Conditions

In existing literature, two categories of convergence properties have been proved for gradient-based BLOs. The first type is "convergence towards stationarity", which guarantees that the UL value-function can converge to a first-order stationary point satisfying $\lim_{K\to\infty} \|\frac{\partial \varphi(\mathbf{x}_T^K)}{\partial \mathbf{x}_T^K}\| = 0$. Here we actually consider the convergence property of the UL variable, i.e., the number of UL iteration $K$ tends to infinity (with a fixed number of LL iteration $T$). The other convergence results actually characterize the following properties: $\mathbf{x}_T \xrightarrow{s} \mathbf{x}^*$[7] and $\inf_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \to \inf_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x}))$ when $T \to \infty$. That is, they prove that for any limit point $\bar{\mathbf{x}}$ of the sequence $\{\mathbf{x}_T\}$, if $\mathbf{x}_T$ is a global (resp. local) minimum of $\varphi_T(\mathbf{x})$, then $\bar{\mathbf{x}}$ is a global (resp. local) minimum of $\varphi(\mathbf{x})$. For convenience, we call this type of property as "convergence towards global/local minimum"[8]. In Table 3, we analyze the convergence properties and conditions required by the UL and LL subproblems for different categories of gradient-based BLOs, including EGBRs (e.g., RHG [12], TRAD [13], HF-MAML [214], STN [9] and BDA [32]), IGBRs (e.g., HOAG [74] and IMAML [84]) and VFBR (e.g., BVFIM [195]).

To guarantee the convergence to stationary solutions, some EGBRs (e.g., TRAD [13], HF-MAML [214] and STN [9]) required the first-order Lipshitz assumption for the UL and LL objectives (i.e., "$L_F$" and "$L_f$" for short) and the twice continuously differentiable property for the LL objective. In addition, there are also some EGBRs that require additional strong assumptions to obtain the first-order stationary points. For instance, HF-MAML [214] relies on second-order Lipshitz assumption (denoted as "Lipschitz-Hessian") for the LL objective, while STN [9] needs the nonsingular Hessian assumption for the LL objective. As for IGBRs (e.g., HOAG [74] and IMAML [84]), they generally require that the gradient (w.r.t. $\mathbf{y}$) of both the UL objective and the LL objective are Lipschitz continuous. As another mainstream EGBR, the work [12] requires that the LL dynamic system $\{\mathbf{y}_T(\mathbf{x})\}$ is uniformly bounded on $\mathcal{X}$ and $\mathbf{y}_T(\mathbf{x})$ uniformly converges to $\mathbf{y}^*(\mathbf{x})$ when $T \to \infty$. Then we can obtain the convergence towards the global/local minimum. As for IGBRs, both the Lipshitz Hessian and nonsingular Hessian are key properties to guarantee their stationarity convergence [74], [84], [214].

### 8.2 A General Proof Template for EGBRs

In this subsection, we would like to further provide a general proof template to analyze the convergence behaviors (i.e., convergence towards global/local minimum) of EGBR methods in more detail. In particular, given the output of the LL dynamic system (i.e., $\mathbf{y}_T(\mathbf{x})$), we first introduce two elementary properties on it as follows:

(1) **Uniform approximation quality to the LL solution:** $\{\mathbf{y}_T(\mathbf{x})\}$ is uniformly bounded on $\mathcal{X}$, and for any $\epsilon > 0$, there exists $t(\epsilon) > 0$ such that whenever $T > t(\epsilon)$, we have

$$\sup_{\mathbf{x}\in\mathcal{X}} \{f(\mathbf{x}, \mathbf{y}_T(\mathbf{x})) - \psi(\mathbf{x})\} \leq \epsilon,$$

or

$$\sup_{\mathbf{x}\in\mathcal{X}} \|\frac{\partial f(\mathbf{x}, \mathbf{y}_T(\mathbf{x}))}{\partial \mathbf{y}_T(\mathbf{x})}\| \leq \epsilon,$$

where $\psi(\mathbf{x})$ denotes the LL value-function, i.e., $\psi(\mathbf{x}) := \min_{\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$.

(2) **Pointwise approximation quality to the ISB solution:** For each $\mathbf{x} \in \mathcal{X}$, we have

$$\lim_{T\to\infty} \texttt{dist}(\mathbf{y}_T(\mathbf{x}), \widetilde{\mathcal{S}}(\mathbf{x})) = 0,$$

7. Here we use "$\xrightarrow{s}$" to denote subsequential convergence.

8. We will provide more details on this convergence property in the following subsection (i.e., Theorem 1).

TABLE 3
Summarizing the convergence results of mainstream gradient-based methods for BLOs within our framework.

| Category | Method | LLS | UL | | LL | | | | | Main convergence results |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $L_F$ | SC | $L_f$ | $\mathcal{C}^2$ | SC | Lip-Hess | NS-Hess | |
| EGBR | TRAD [13] | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | Stationarity: $\frac{\partial \varphi(\mathbf{x}_T^K)}{\partial \mathbf{x}_T^K} \to 0$. |
| | HF-MAML [214] | ✓ | ✓ | ✗ | ✓ | ✓ | ✗ | ✓ | ✗ | |
| | STN [9] | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | |
| | RHG [12] | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | Global/local minimum: $\mathbf{x}_T \xrightarrow{s} \mathbf{x}^*$, $\inf_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \to \inf_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x})$. |
| | BDA [32] | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | |
| | | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | |
| | BDA [189] | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | |
| VFBR | BVFIM [195] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | |
| IGBR | HOAG [74] | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | Stationarity: $\frac{\partial \varphi(\mathbf{x}_T^K)}{\partial \mathbf{x}_T^K} \to 0$. |
| | IMAML [84] | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | |

[1] Notice that $F(\mathbf{x}, \mathbf{y})$ is continuously differentiable on $\mathcal{X} \times \mathcal{Y}$ ($\mathcal{X}$ is a compact set) and $f(\mathbf{x}, \mathbf{y})$ is continuously differentiable on $\mathbb{R}^m \times \mathcal{Y}$. The feasible solution set $\mathcal{Y}$ represents the whole space $\mathbb{R}^n$.

[2] "$L_F$ (resp. $L_f$)" means the gradient of $F(\mathbf{x}, \cdot)$ (resp. $f(\mathbf{x}, \cdot)$) is Lipschitz continuous with Lipschitz constant $L_F$ (resp. $L_f$). SC means strongly convex and $\mathcal{C}^2$ implies that $f(\mathbf{x}, \cdot)$ is second-order continuously differentiable w.r.t. $\mathbf{y}$. "NS-Hess" and "Lip-Hess" represent the nonsingularity and Lipschitz properties of Hessian $\frac{\partial^2 f}{\partial \mathbf{y} \partial \mathbf{y}'}$, respectively. Please refer to [74], [84], [214] for more details on these variational analysis concepts.

[3] Here we respectively represent "required" and "not required" by "✓" and "✗" for these properties.

[4] We summarize two kinds of convergence properties, i.e., "stationarity" and "global/local minimum". The former implies that the gradient descent on the UL value-function converges to first-order stationary points satisfying $\lim_{K\to\infty} \|\frac{\partial \varphi(\mathbf{x}_T^K)}{\partial \mathbf{x}_T^K}\| = 0$ (with a fixed number of LL iterations $T$), while the latter characterizes the convergence towards global/local minimum satisfying $\mathbf{x}_T \xrightarrow{s} \mathbf{x}^*$ and $\inf_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \to \inf_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x})$ as $T \to \infty$.

where $\widetilde{\mathcal{S}}(\mathbf{x})$ represents the solution set of the ISB subproblem in Eq. (6) and $\texttt{dist}(\cdot, \cdot)$ denotes the point-to-set distance.

Equipped with the above two properties on $\{\mathbf{y}_T(\mathbf{x})\}$, we can present general convergence results of Eqs. (11)-(14) in the following theorem[9].

**Theorem 1.** *(Convergence towards global/local minimum) Suppose that the generated sequence $\{\mathbf{y}_t(\mathbf{x})\}$ satisfies the above two properties. Let $\mathbf{x}_T$ be a global (resp. local) minimum of $\varphi_T(\mathbf{x})$, i.e., $\mathbf{x}_T \in \arg\min_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x})$. Then we have*

(1) *Any limit point $\bar{\mathbf{x}}$ of the sequence $\{\mathbf{x}_T\}$ is a global (resp. local) minimum of $\varphi(\mathbf{x})$, i.e., $\bar{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x})$.*

(2) *$\inf_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \to \inf_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x})$ as $T \to \infty$.*

*Proof.* In the following, we first state the key steps for proving convergence properties in the global scenario and then demonstrate how to obtain the local convergence properties accordingly.

Step 1. We should first verify that for $\bar{\mathbf{x}} \in \mathcal{X}$, $\psi$ satisfies

$$\limsup_{\mathbf{x}\to\bar{\mathbf{x}}} \psi(\mathbf{x}) = \psi(\bar{\mathbf{x}}).$$

Step 2. Then for any limit point $\bar{\mathbf{x}}$ of the sequence $\{\mathbf{x}_T\}$, there exists $\mathbf{y}_m(\mathbf{x}_m) \to \bar{\mathbf{y}}$ for a subsequence $\{\mathbf{x}_m\}$ and some $\bar{\mathbf{y}}$. Thus we can obtain $\bar{\mathbf{y}} \in \mathcal{S}(\bar{\mathbf{x}})$.

Step 3. Next, we verify the convergence property of the UL objective as follows:

$$\lim_{T\to\infty} \varphi_T(\mathbf{x}) = \varphi(\mathbf{x}).$$

Step 4. For any $\epsilon > 0$, we verify the following inequality:

$$\varphi(\bar{\mathbf{x}}) \le F(\mathbf{x}_m, \mathbf{y}_m(\mathbf{x}_m)) + \epsilon \le \lim_{m\to\infty} \varphi_m(\mathbf{x}) + \epsilon, \ \forall \mathbf{x} \in \mathcal{X}.$$

9. Here we actually provide a brief proof roadmap, which is summarized based on theoretical studies in existing works [32], [51], [189], [195].

Step 5. Finally, we verify the following inequality

$$\limsup_{T\to\infty} \left\{ \inf_{\mathbf{x}\in\mathcal{X}} \varphi_T(\mathbf{x}) \right\} \le \inf_{\mathbf{x}\in\mathcal{X}} \varphi(\mathbf{x}).$$

Thus we can obtain convergence results stated in Theorem 1.

For convergence to the local minimum, we actually consider $\mathbf{x}_T$ as a local minimum of $\varphi_T(\mathbf{x})$ with uniform neighborhood modulus $\delta > 0$. Then any limit point $\bar{\mathbf{x}}$ of the sequence $\{\mathbf{x}_T\}$ is a local minimum of $\varphi(\mathbf{x})$, i.e., there exists $\tilde{\delta} > 0$ such that $\varphi(\bar{\mathbf{x}}) \le \varphi(\mathbf{x}), \forall \mathbf{x} \in \mathbb{B}_\delta(\bar{\mathbf{x}}) \cap \mathcal{X}$. According to the neighborhood property to spread out the analysis, the result of convergence towards local minimum can also be proved by the same steps. □

The above theoretical results actually provide us a general recipe to analyze the iteration behaviors and convergence properties of gradient-based BLOs, especially for EGBRs. In other words, we can understand that these existing numerical schemes and their required assumptions on the UL and LL subproblems are just to meet the above elementary iteration properties.

It can be observed that classical EGBRs (e.g., [12], [13]) require to first enforce the LLS assumption on the BLO problem. The work in [12] assumes that the UL and LL objectives are continuously differentiable and also enforces the restrictive (local) strong convexity assumption on the LL objective. In fact, such properties can ensure the uniform convergence of $\{\mathbf{y}_T(\mathbf{x})\}$ towards $\mathbf{y}^*(\mathbf{x})$, thus leading to the two elementary properties. In fact, the LLS assumption considered in [12] is stricter than that required in the proof template. The works in [32], [189] also consider that the UL and LL objectives are continuously differentiable, but make a weaker assumption on the LL objective, i.e., $f(\mathbf{x}, \mathbf{y})$ is level-bounded in $\mathbf{y}$ and locally uniform in $\mathbf{x} \in \mathcal{X}$ (or the gradient of $f(\mathbf{x}, \cdot)$ is Lipschitz continuous). Indeed,

it can be verified that the conditions in [32], [189] can also ensure two elementary properties required by our proof template. Therefore, we have that the above two elementary convergence properties hold and we can obtain the convergence results stated in Theorem 1.

It has been verified in [32], [189] that these classical EGBRs [12], [13] may lead to incorrect solutions if the LLS assumption is not satisfied. As stated in the above Section 7, BDA [32], [189] has been proposed to extend the EGBR method to address this issue. Theoretically, the work in [32] actually introduces the LL solution set property and the UL objective convergence property. Theoretical investigations in [189] further demonstrate that the iterative gradient-aggregation dynamics can solve the ISB subproblem without the LL singleton assumption and the UL strong convexity. Again, in order to remove the restrictive singleton and convex assumptions on the LL objective, BVFIM [195] further proves the same convergence results by introducing the strong constraints on a series of positive decreasing parameters $(\mu, \theta, \tau)$ when $f(\mathbf{x}, \mathbf{y})$ and $F(\mathbf{x}, \mathbf{y})$ are level-bounded in $\mathbf{y}$ and locally uniformly in $\mathbf{x} \in \mathcal{X}$.

### 8.3 Time and Space Complexity

In this subsection, we analyze the complexity of time and space for these mainstream gradient-based BLO methods (i.e., EGBRs [13], [32], [51], IGBRs [72], [74], [84] and VFBR [195]), as summarized in Table 4. Please notice that here we just follow most BLO literature (e.g., [13], [14], [51]) to only estimate the complexity of computing the gradient of $\varphi$ w.r.t. $\mathbf{x}$ (defined in Eq. (9)) with a fixed (e.g., $T$-step) LL iteration.

**EGBR:** As discussed in Section 5, EGBRs generally construct the BR mapping $\mathbf{y}^*(\mathbf{x})$ or the indirect gradient $\mathbf{G}(\mathbf{x})$ with the implementation of an unrolled dynamic system (see Eq. (13)). In [51], the dynamic system can be implemented in either a forward automatic differential mode (i.e., FAD) or a reverse automatic differential mode (i.e., RAD). Especially, BDA implements a reverse aggregated gradient flow from the UL and LL subproblems to approximate the BR mapping. More specifically, taking into account the fact that the Hessian-matrix product is repeatedly calculated (i.e., $\sum_{t=0}^{T}\left(\prod_{i=t+1}^{T}\mathbf{A}_i\right)\mathbf{B}_t$) in the forward propagation, FAD requires the space complexity $O(mn)$ and the time complexity $O(m^2nT)$. RAD in the backward pass needs to evaluate Hessian- and Jacobian-vector products, and stores all the intermediate variables $\{\mathbf{y}_t \in \mathbb{R}^n\}_{t=1}^{T}$ in memory. So we have that the time and space costs are $O(n(m+n)T)$ and $O(m+nT)$, respectively. By ignoring the long-term dependencies, TRAD uses the truncated backward propagation trajectory with a smaller number of steps (i.e., $M < T$). As for BDA, with the similar backward propagation manner, we have that the complexity of time and space is the same as that for RAD.

**IGBR:** As for IGBRs, we observe that they require to derive the indirect gradient based on the implicit function theorem, which results in the overloaded computation with respect to the inverse of Hessian (see Eq. (31)). To mitigate this problem, IGBRs generally solve a linear system by Conjugate Gradient (CG) [74], [84] or Neumann series [72], as stated in Section 6. Without loss of generality, we uniformly assume that these methods perform $J$-step iterations to solve the linear system. Each step contains a hessian-vector product computation requiring the time cost $O(m+n^2J)$. Then with a $T$-step gradient descent on the LL subproblem, we have that the overall time and space complexities can be written as $O(m+nT+n^2J)$ and $O(m+n)$, respectively. It should be noted that the iteration step $J$ generally relies on the properties of Hessian-matrix, thus it should be set much larger than $T$.

**VFBR:** It has been stated in Section 7 that VFBR type method (i.e., BVFIM) does not require to solve the unrolled dynamic system or approximate the inverse of Hessian, thus can obtain lower time and space complexity than EGBRs and IGBRs, especially on BLOs with high-dimensional LL subproblems. Specifically, we use $Q_1$ and $Q_2$ to represent the number of gradient iterations for solving the regularized subproblems in Eqs. (34) and (35), respectively. Then it can be checked that the time costs of calculating each gradient descent for the LL and UL value-functions are $O(nQ_1)$ and $O(nQ_2)$, respectively. Moreover, we require additional $O(m)$ time to perform the UL gradient updating. Thus the overall time cost of BVFIM is $O(m+n(Q_1+Q_2))$. As for the space complexity, it is easy to check that BVFIM requires $O(m+n)$ space cost and is the same as that in IGBRs.

TABLE 4
Comparison of the time and space complexity for several gradient-based mainstream BLOs.

| Category | Method | Time | Space |
|---|---|---|---|
| EGBR | FAD [51] | $O(m^2nT)$ | $O(mn)$ |
| | RAD [51] | $O(n(m+n)T)$ | $O(m+nT)$ |
| | BDA [32] | $O(n(m+n)T)$ | $O(m+nT)$ |
| | TRAD [13] | $O(n(m+n)M)$ | $O(m+nM)$ |
| IGBR | CG [74], [84] | $O(m+nT+n^2J)$ | $O(m+n)$ |
| | Neumann [72] | $O(m+nT+n^2J)$ | $O(m+n)$ |
| VFBR | BVFIM [195] | $O(m+n(Q_1+Q_2))$ | $O(m+n)$ |

It can be seen in Table 4 that the reverse propagation methods (i.e., RAD, TRAD and BDA) have benefited from the lightweight matrix-vector multiplication (rather than the overweight Hessian-matrix), thus can obtain less computational complexity in comparison to the forward propagation approach (e.g., FAD). Especially for TRAD, the time and space complexity can be further reduced by the truncated backward propagation strategy. Compared with EGBRs, IGBRs maintain higher computational complexity due to the overloaded computation in terms of the inverse of Hessian. In contrast, VFBR can obtain lower time-consuming than both EGBRs and IGBRs. It actually also outperforms EGBRs in costing less memory, especially when solving the LL subproblem on high-dimensional tasks (e.g., with extremely large $n$).

## 9 Potentials for New Algorithms Design

As the last but not least part of the survey, this section aims to demonstrate the potentials of our general algorithmic framework for designing new gradient schemes for challenging BLO formulations, such as pessimistic BLOs (stated in Eq. (5)).

In fact, pessimistic BLO formulation can be naturally interpreted as a non-cooperative game between two players

and has been utilized to formulate problems in the area of mathematical programming [215], [216], [217] and other application fields, such as economics [218], [219] and biology [220]. However, from the pessimistic viewpoint, the UL player (i.e., leader) cannot anticipate the LL player (i.e., follower)'s decision, the constraint must be satisfied for any rational decision of the follower, thus pessimistic BLO is perceived to be very difficult to solve, especially in high-dimensional application scenarios [196].

Now we demonstrate how to develop a practical algorithm within our BR mapping based BLO algorithmic framework for pessimistic BLO formulations[10]. Concretely, based on Eq. (5) and pessimistic BR mapping (defined in Eq. (7)), we can follow the similar idea in Eq. (32) to aggregate the UL and LL gradients

$$\widetilde{\mathbf{d}}(\mathbf{y}_{t-1};\mathbf{x}) = -\rho_t \frac{\partial F(\mathbf{x},\mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}} + (1-\rho_t)\frac{\partial f(\mathbf{x},\mathbf{y}_{t-1})}{\partial \mathbf{y}_{t-1}}.$$

With the above procedure, it can be seen that the only difference between $\mathbf{d}$ and $\widetilde{\mathbf{d}}$ is just the sign of the UL gradient. Thus we can adopt the same calculation scheme as that in [32], [53] to solve Eq. (5). The corresponding roadmap is also illustrated in Fig. 9.

## 10 Conclusions and Future Prospects

Bi-Level Optimization (BLO) is an important mathematical tool for modeling and solving machine learning and computer vision problems that have hierarchical optimization structures, such as hyper-parameter optimization, multi-task and meta learning, neural architecture search, adversarial learning and deep reinforcement learning, etc. In the above sections, we first demonstrated how to formulate different learning and vision tasks from a uniform BLO perspective. We then established a value-function-based single-level reformulation for different categories of BLO models and proposed a best-response-based optimization platform to uniformly understand and formulate a variety of existing gradient-based BLO methods. The convergence behaviors and complexity properties of these BLO algorithms have also been discussed. We also demonstrated potentials of our BLO platform for designing new algorithms to solve the more challenging pessimistic BLOs tasks. Future research of BLOs may focus but is not limited to the following aspects:

- **Theoretical breakthrough:** The convergence behaviors of gradient-based algorithms on various challenging BLOs, such as pessimistic BLOs [215], [221], [222], BLOs with complex constraints [223], [224], non-convex objectives [225] and multiple followers [226], should be investigated.
- **Computational improvement:** It is also urgent to design efficient acceleration techniques (e.g., momentum and its variations) to speed up gradient-based BLOs in high-dimensional optimization scenario [227], [228], [229].
- **Wider applications:** Recent deep learning tasks (e.g., knowledge distillation [230], self-supervised learning [231], and transformer [232]) are more and more sophisticated. BLOs should be a promising tool to formulate and analyze these complex learning paradigms.

10. We emphasize that we just present an example to demonstrate the potentials of our framework for a new algorithm design. Strict theoretical analysis and evaluations are definitely out of the scope in this paper and will be considered as future work.

## Acknowledgments

This work is partially supported by the National Key R&D Program of China (No. 2020YFB1313503), the National Natural Science Foundation of China (No. 61922019), and the Fundamental Research Funds for the Central Universities.

## References

[1] S. Dempe, "Bilevel optimization: Theory, algorithms, applications and a bibliography," in *Bilevel Optimization*, 2020, pp. 581–672.
[2] H. Von Stackelberg and S. H. Von, *The theory of the market economy*. Oxford University Press, 1952.
[3] J. Fortuny-Amat and B. McCarl, "A representation and economic interpretation of a two-level programming problem," *Journal of the Operational Research Society*, vol. 32, no. 9, pp. 783–792, 1981.
[4] S. Dempe, V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova, "Bilevel programming problems," *Energy Systems. Springer, Berlin*, 2015.
[5] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: from classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 276–295, 2017.
[6] S. Wogrin, S. Pineda, and D. A. Tejada-Arango, "Applications of bilevel optimization in energy and electricity markets," in *Bilevel Optimization*, 2020, pp. 139–168.
[7] J. Domke, "Generic methods for optimization-based modeling," in *Artificial Intelligence and Statistics*, 2012.
[8] C.-s. Foo, C. B. Do, and A. Y. Ng, "Efficient multiple hyperparameter learning for log-linear models," in *NeurPIS*, 2008.
[9] M. MacKay, P. Vicol, J. Lorraine, D. Duvenaud, and R. B. Grosse, "Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions," in *ICLR*, 2019.
[10] T. Okuno, A. Takeda, and A. Kawana, "Hyperparameter learning via bilevel nonsmooth optimization," *arXiv:1806.01520*, 2018.
[11] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "A bridge between hyperparameter optimization and learning-to-learn," *arXiv:1712.06283*, 2017.
[12] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *ICML*, 2018.
[13] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots, "Truncated back-propagation for bilevel optimization," in *AISTATS*, 2019.
[14] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *ICLR*, 2019.
[15] Y. Hu, X. Wu, and R. He, "TF-NAS: rethinking three search freedoms of latency-constrained differentiable neural architecture search," in *ECCV*, 2020.
[16] D. Lian, Y. Zheng, Y. Xu, Y. Lu, L. Lin, P. Zhao, J. Huang, and S. Gao, "Towards fast adaptation of neural architectures with meta learning," in *ICLR*, 2019.
[17] Y. Li, L. Song, X. Wu, R. He, and T. Tan, "Learning a bi-level adversarial network with global and local perception for makeup-invariant face verification," *Pattern Recognition*, vol. 90, pp. 99–108, 2019.
[18] H. Jiang, Z. Chen, Y. Shi, B. Dai, and T. Zhao, "Learning to defense by learning to attack," *arXiv:1811.01213*, 2018.
[19] Y. Tian, L. Shen, G. Su, Z. Li, and W. Liu, "Alphagan: Fully differentiable architecture search for generative adversarial networks," *arXiv:2006.09134*, 2020.
[20] H. Zhang, W. Chen, Z. Huang, M. Li, Y. Yang, W. Zhang, and J. Wang, "Bi-level actor-critic for multi-agent coordination," in *AAAI*, 2020.
[21] D. Pfau and O. Vinyals, "Connecting generative adversarial networks and actor-critic methods," *arXiv:1610.01945*, 2016.

[22] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost," in *NeurIPS*, 2019.

[23] P. Ochs, R. Ranftl, T. Brox, and T. Pock, "Bilevel optimization with nonsmooth lower level problems," in *SSVM*, 2015.

[24] J. Chen, P. Mu, R. Liu, X. Fan, and Z. Luo, "Flexible bilevel image layer modeling for robust deraining," in *ICME*, 2020.

[25] R. Liu, Z. Li, Y. Zhang, X. Fan, and Z. Luo, "Bi-level probabilistic feature learning for deformable image registration," in *IJCAI*, 2020.

[26] G. Kunapuli, K. P. Bennett, J. Hu, and J.-S. Pang, "Classification model selection via bilevel programming," *Optimization Methods & Software*, vol. 23, no. 4, pp. 475–489, 2008.

[27] S. Dempe and S. Franke, "On the solution of convex bilevel optimization problems," *Computational Optimization and Applications*, vol. 63, no. 3, pp. 685–703, 2016.

[28] P. Hansen, B. Jaumard, and G. Savard, "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 5, pp. 1194–1217, 1992.

[29] A. B. Zemkoho, "Solving ill-posed bilevel programs," *Set-Valued and Variational Analysis*, vol. 24, no. 3, pp. 423–448, 2016.

[30] L. Vicente, G. Savard, and J. Júdice, "Descent approaches for quadratic bilevel programming," *Journal of Optimization Theory and Applications*, vol. 81, no. 2, pp. 379–399, 1994.

[31] H. I. Calvete and C. Galé, "Algorithms for linear bilevel optimization," in *Bilevel Optimization*, 2020.

[32] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, "A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton," in *ICML*, 2020.

[33] A. Sharma, "Optimistic variants of single-objective bilevel optimization for evolutionary algorithms," *International Journal of Computational Intelligence and Applications*, no. 03, p. 2050020, 2020.

[34] H. I. Calvete, C. Galé, S. Dempe, and S. Lohse, "Bilevel problems over polyhedra with extreme point optimal solutions," *Journal of Global Optimization*, vol. 53, no. 3, pp. 573–586, 2012.

[35] J. Lu, C. Shi, G. Zhang, and D. Ruan, "An extended branch and bound algorithm for bilevel multi-follower decision making in a referential-uncooperative situation," *International Journal of Information Technology Decision Making*, vol. 6, no. 02, pp. 371–388, 2007.

[36] G. Savard and J. Gauvin, "The steepest descent direction for the nonlinear bilevel programming problem," *Operations Research Letters*, vol. 15, no. 5, pp. 265–272, 1994.

[37] E. S. H. Neto and Á. R. De Pierro, "On perturbed steepest descent methods with inexact line search for bilevel convex optimization," *Optimization*, vol. 60, no. 8-9, pp. 991–1008, 2011.

[38] G. Anandalingam and D. White, "A solution method for the linear static stackelberg problem using penalty functions," *IEEE Transactions on Automatic Control*, vol. 35, no. 10, pp. 1170–1173, 1990.

[39] Z. Wan, L. Mao, and G. Wang, "Estimation of distribution algorithm for a class of nonlinear bilevel programming problems," *Information Sciences*, vol. 256, pp. 184–196, 2014.

[40] B. El-Sobky and Y. Abo-Elnaga, "A penalty method with trust-region mechanism for nonlinear bilevel optimization problem," *Journal of Computational and Applied Mathematics*, vol. 340, pp. 360–374, 2018.

[41] S. Dempe and J. F. Bard, "Bundle trust-region algorithm for bilinear bilevel programming," *Journal of Optimization Theory and Applications*, vol. 110, no. 2, pp. 265–288, 2001.

[42] G. B. Allende and G. Still, "Solving bilevel programs with the kkt-approach," *Mathematical Programming*, vol. 138, no. 1-2, pp. 309–332, 2013.

[43] A. Sinha, T. Soun, and K. Deb, "Using karush-kuhn-tucker proximity measure for solving bilevel optimization problems," *Swarm and Evolutionary Computation*, vol. 44, pp. 496–510, 2019.

[44] A. Sinha, S. Bedi, and K. Deb, "Bilevel optimization based on kriging approximations of lower level optimal value function," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–8.

[45] S. E. Yimer, P. Kumam, and A. G. Gebrie, "Proximal gradient method for solving bilevel optimization problems," *Mathematical and Computational Applications*, no. 4, p. 66, 2020.

[46] A. U. Raghunathan and L. T. Biegler, "Mathematical programs with equilibrium constraints (mpecs) in process engineering," *Computers & chemical engineering*, no. 10, pp. 1381–1392, 2003.

[47] A. B. Zemkoho and S. Zhou, "Theoretical and numerical comparison of the karush–kuhn–tucker and value function reformulations in bilevel optimization," *Computational Optimization and Applications*, pp. 1–50, 2020.

[48] D. Aussel and C. Lalitha, *Generalized Nash equilibrium problems, Bilevel programming and MPEC*. Springer, 2018.

[49] G. Kunapuli, *A bilevel optimization approach to machine learning*. Citeseer, 2008.

[50] S. Dempe and J. Dutta, "Is bilevel programming a special case of a mathematical program with complementarity constraints?" *Mathematical programming*, vol. 131, no. 1, pp. 37–48, 2012.

[51] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil, "Forward and reverse gradient-based hyperparameter optimization," *arXiv:1703.01785*, 2017.

[52] T. Okuno and A. Takeda, "Bilevel optimization of regularization hyperparameters in machine learning," in *Bilevel Optimization*, 2020.

[53] Y. Liu and R. Liu, "Boml: A modularized bilevel optimization library in python for meta learning," in *ICME*, 2021.

[54] K. Ji, J. Yang, and Y. Liang, "Multi-step model-agnostic meta-learning: Convergence and improved algorithms," *arXiv:2002.07836*, 2020.

[55] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv:1707.09835*, 2017.

[56] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. Freitas, "Learning to learn without gradient descent by gradient descent," in *ICML*, 2017.

[57] A. Antoniou, H. Edwards, and A. Storkey, "How to train your maml," *arXiv:1810.09502*, 2018.

[58] H. Dong, B. Zou, L. Zhang, and S. Zhang, "Automatic design of cnns via differentiable neural architecture search for polsar image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 9, pp. 6362–6375, 2020.

[59] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *ICLR*, 2019.

[60] Y. Liu, W. Cai, X. Yuan, and J. Xiang, "Gl-gan: Adaptive global and local bilevel optimization model of image generation," *arXiv:2008.02436*, 2020.

[61] L. Wang, Q. Cai, Z. Yang, and Z. Wang, "On the global optimality of model-agnostic meta-learning: Reinforcement learning and supervised learning," in *ICML*, 2020.

[62] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic," *arXiv:2007.05170*, 2020.

[63] G. Anandalingam and T. L. Friesz, "Hierarchical optimization: An introduction," *Annals of Operations Research*, vol. 34, no. 1, pp. 1–11, 1992.

[64] U.-P. Wen and S.-T. Hsu, "Linear bi-level programming problems—a review," *Journal of the Operational Research Society*, vol. 42, no. 2, pp. 125–133, 1991.

[65] S. Ukkusuri, K. Doan, and H. A. Aziz, "A bi-level formulation for the combined dynamic equilibrium based traffic signal control," *Procedia-Social and Behavioral Sciences*, vol. 80, no. 7, pp. 729–752, 2013.

[66] K. Lachhwani and A. Dwivedi, "Bi-level and multi-level programming problems: taxonomy of literature review and research issues," *Archives of Computational Methods in Engineering*, vol. 25, no. 4, pp. 847–877, 2018.

[67] A. Chinchuluun, P. M. Pardalos, and H.-X. Huang, "Multilevel (hierarchical) optimization: complexity issues, optimality conditions, algorithms," in *Applied Mathematics and Global Optimization*, 2009.

[68] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo, "On differentiating parameterized argmin and argmax problems with application to bi-level optimization," *arXiv:1607.05447*, 2016.

[69] C.-A. Deledalle, S. Vaiter, J. Fadili, and G. Peyré, "Stein unbiased gradient estimator of the risk (sugar) for multiple parameter selection," *SIAM Journal on Imaging Sciences*, vol. 7, no. 4, pp. 2448–2487, 2014.

[70] W. Jiang and S. Siddiqui, "Hyper-parameter optimization for support vector machines using stochastic gradient descent and dual coordinate descent," *EURO Journal on Computational Optimization*, vol. 8, no. 1, pp. 85–101, 2020.

[71] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *ICML*, 2015.

[72] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *AISTATS*, 2020.

[73] L. Franceschi, R. Grazzi, M. Pontil, S. Salzo, and P. Frasconi, "Far-ho: A bilevel programming package for hyperparameter optimization and meta-learning," *arXiv:1806.04941*, 2018.

[74] F. Pedregosa, "Hyperparameter optimization with approximate gradient," *arXiv:1602.02355*, 2016.

[75] V. Likhosherstov, X. Song, K. Choromanski, J. Davis, and A. Weller, "Ufo-blo: Unbiased first-order bilevel optimization," *arXiv:2006.03631*, 2020.

[76] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *ICML*, 2017.

[77] J. Bae and R. B. Grosse, "Delta-stn: Efficient bilevel optimization for neural networks using structured response jacobians," in *NeurIPS*, 2020.

[78] S. Qiao, C. Liu, W. Shen, and A. L. Yuille, "Few-shot image recognition by predicting parameters from activations," in *CVPR*, 2018.

[79] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *CVPR*, 2018.

[80] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv:1707.03141*, 2017.

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

[82] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *ICLR*, 2018.

[83] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *ICML*, 2019.

[84] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *NeurIPS*, 2019.

[85] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," *arXiv:1805.08136*, 2018.

[86] E. Park and J. B. Oliva, "Meta-curvature," in *NeurIPS*, 2019.

[87] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2016.

[88] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[89] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv:1803.02999*, 2018.

[90] H. S. Behl, A. G. Baydin, and P. H. Torr, "Alpha maml: Adaptive model-agnostic meta-learning," *arXiv:1905.07435*, 2019.

[91] J. Guo, X. Zhu, C. Zhao, D. Cao, Z. Lei, and S. Z. Li, "Learning meta face recognition in unseen domains," in *CVPR*, 2020.

[92] Q. Wu, Z. Lin, G. Wang, H. Chen, B. F. Karlsson, B. Huang, and C.-Y. Lin, "Enhanced meta-learning for cross-lingual named entity recognition with minimal resources." in *AAAI*, 2020.

[93] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv:1803.02999*, 2018.

[94] P. Zhou, X. Yuan, H. Xu, S. Yan, and J. Feng, "Efficient meta learning via minibatch proximal update," in *NeurIPS*, 2019.

[95] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "Es-maml: Simple hessian-free meta learning," *arXiv:1910.01215*, 2019.

[96] G. Denevi, C. Ciliberto, R. Grazzi, and M. Pontil, "Learning-to-learn stochastic gradient descent with biased regularization," in *ICML*, 2019.

[97] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layerwise metric and subspace," *arXiv:1801.05558*, 2018.

[98] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," in *CVPR*, 2020.

[99] P. Tian, Z. Wu, L. Qi, L. Wang, Y. Shi, and Y. Gao, "Differentiable meta-learning model for few-shot semantic segmentation." in *AAAI*, 2020.

[100] J.-Y. Hsu, Y.-J. Chen, and H.-y. Lee, "Meta learning for end-to-end low-resource speech recognition," in *ICASSP*, 2020.

[101] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural automl," in *NeurIPS*, 2018.

[102] C. Jiang, H. Xu, W. Zhang, X. Liang, and Z. Li, "Sp-nas: Serial-to-parallel backbone search for object detection," in *CVPR*, 2020.

[103] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv:1812.09926*, 2018.

[104] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv:1802.03268*, 2018.

[105] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *CVPR*, 2019.

[106] A. Noy, N. Nayman, T. Ridnik, N. Zamir, S. Doveh, I. Friedman, R. Giryes, and L. Zelnik, "Asap: Architecture search, anneal and prune," in *AISTATS*, 2020.

[107] T. Elsken, B. Staffler, J. H. Metzen, and F. Hutter, "Meta-learning of neural architectures for few-shot learning," in *CVPR*, 2020.

[108] Q. Yao, J. Xu, W.-W. Tu, and Z. Zhu, "Efficient neural architecture search via proximal iterations." in *AAAI*, 2020.

[109] S. Hu, S. Xie, H. Zheng, C. Liu, J. Shi, X. Liu, and D. Lin, "Dsnas: Direct neural architecture search without parameter retraining," in *CVPR*, 2020.

[110] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019.

[111] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," in *NeurIPS*, 2019.

[112] H. Xu, L. Yao, W. Zhang, X. Liang, and Z. Li, "Auto-fpn: Automatic network architecture adaptation for object detection beyond classification," in *ICCV*, 2019.

[113] J. Chang, Y. Guo, G. MENG, S. XIANG, C. Pan *et al.*, "Data: Differentiable architecture approximation," in *NeurIPS*, 2019.

[114] Q. Yu, D. Yang, H. Roth, Y. Bai, Y. Zhang, A. L. Yuille, and D. Xu, "C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation," in *CVPR*, 2020.

[115] K. Zhou, Q. Song, X. Huang, and X. Hu, "Auto-gnn: Neural architecture search of graph neural networks," *arXiv:1909.03184*, 2019.

[116] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng, "Autost: Efficient neural architecture search for spatio-temporal prediction," in *KDD*, 2020.

[117] J. Guo, K. Han, Y. Wang, C. Zhang, Z. Yang, H. Wu, X. Chen, and C. Xu, "Hit-detector: Hierarchical trinity architecture search for object detection," in *CVPR*, 2020.

[118] C. He, H. Ye, L. Shen, and T. Zhang, "Milenas: Efficient neural architecture search via mixed-level reformulation," in *CVPR*, 2020.

[119] W. Cheng, Y. Shen, and L. Huang, "Differentiable neural input search for recommender systems," *arXiv:2006.04466*, 2020.

[120] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv:1611.02163*, 2016.

[121] H. Yin, D. Li, X. Li, and P. Li, "Meta-cotgan: A meta cooperative training paradigm for improving adversarial text generation." in *AAAI*, 2020.

[122] C. Gao, Y. Chen, S. Liu, Z. Tan, and S. Yan, "Adversarialnas: Adversarial neural architecture search for gans," in *CVPR*, 2020.

[123] C. Jin, P. Netrapalli, and M. Jordan, "What is local optimality in nonconvex-nonconcave minimax optimization?" in *ICML*, 2020.

[124] J. Hamm and Y.-K. Noh, "K-beam minimax: Efficient optimization for deep adversarial learning," in *ICML*, 2018.

[125] Z. Chen, D. Liu, X. Wu, and X. Xu, "Research on distributed renewable energy transaction decision-making based on multi-agent bilevel cooperative reinforcement learning," in *CIRED*, 2019.

[126] S. Tschiatschek, A. Ghosh, L. Haug, R. Devidze, and A. Singla, "Learner-aware teaching: Inverse reinforcement learning with preferences and constraints," in *NeurIPS*, 2019.

[127] Y. Zhang, Q. Cai, Z. Yang, and Z. Wang, "Generative adversarial imitation learning with neural network parameterization: Global optimality and convergence rate," in *ICML*, 2020.

[128] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *AAMAS*, 2020.

[129] Y. Gao, H. Yang, P. Zhang, C. Zhou, and Y. Hu, "Graphnas: Graph neural architecture search with reinforcement learning," *arXiv:1904.09981*, 2019.

[130] Y. Ye, D. Qiu, M. Sun, D. Papadaskalopoulos, and G. Strbac, "Deep reinforcement learning for strategic bidding in electricity markets," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1343–1355, 2019.

[131] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *NeurIPS*, 2016.

[132] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," in *NeurIPS*, 2017.

[133] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv:1807.06158*, 2018.

[134] H. Liu, R. Socher, and C. Xiong, "Taming maml: Efficient unbiased meta-reinforcement learning," in *ICML*, 2019.

[135] K. Kunisch and T. Pock, "A bilevel optimization approach for parameter learning in variational models," *SIAM Journal on Imaging Sciences*, vol. 6, no. 2, pp. 938–983, 2013.

[136] R. Liu, J. Liu, Z. Jiang, X. Fan, and Z. Luo, "A bilevel integrated model with data-driven layer ensemble for multi-modality image fusion," *IEEE Transactions on Image Processing*, vol. 30, pp. 1261–1274, 2020.

[137] H. Li and L. Zhang, "A bilevel learning model and algorithm for self-organizing feed-forward neural networks for pattern classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[138] P. Zhou, C. Zhang, and Z. Lin, "Bilevel model-based discriminative dictionary learning for recognition," *IEEE transactions on image processing*, vol. 26, no. 3, pp. 1173–1187, 2016.

[139] B. Fernando and S. Gould, "Learning end-to-end video classification with rank-pooling," in *ICML*, 2016, pp. 1187–1196.

[140] D. Pfau, S. Petersen, A. Agarwal, D. G. Barrett, and K. L. Stachenfeld, "Spectral inference networks: Unifying deep and spectral learning," in *ICLR*, 2019.

[141] M. D'Elia, J. De los Reyes, and A. M. Trujillo, "Bilevel parameter optimization for nonlocal image denoising models," *arXiv:1912.02347*, 2019.

[142] B. Stadie, L. Zhang, and J. Ba, "Learning intrinsic rewards as a bi-level optimization problem," in *UAI*, 2020, pp. 111–120.

[143] R. Liu, Z. Li, X. Fan, C. Zhao, H. Huang, and Z. Luo, "Learning deformable image registration from optimization: perspective, modules, bilevel training and beyond," *arXiv:2004.14557*, 2021.

[144] Q. Pham, D. Sahoo, C. Liu, and S. C. Hoi, "Bilevel continual learning," *arXiv:2007.15553*, 2020.

[145] R. Liu, P. Mu, J. Chen, X. Fan, and Z. Luo, "Investigating task-driven latent feasibility for nonconvex image modeling," *IEEE Transactions on Image Processing*, vol. 29, pp. 7629–7640, 2020.

[146] R. Liu, L. Ma, X. Yuan, S. Zeng, and J. Zhang, "Bilevel integrative optimization for ill-posed inverse problems," *arXiv:1907.03083*, 2019.

[147] R. Liu, L. Ma, J. Zhang, X. Fan, and Z. Luo, "Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement," *arXiv:2012.05609*, 2020.

[148] T. Stouraitis, I. Chatzinikolaidis, M. Gienger, and S. Vijayakumar, "Online hybrid motion planning for dyadic collaborative manipulation via bilevel optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1452–1471, 2020.

[149] S. Mounsaveng, I. Laradji, I. Ben Ayed, D. Vazquez, and M. Pedersoli, "Learning data augmentation with online bilevel optimization for image classification," in *WACV*, 2020.

[150] Z. Borsos, M. Tagliasacchi, and A. Krause, "Semi-supervised batch active learning via bilevel optimization," *arXiv:2010.09654*, 2020.

[151] Z. Borsos, M. Mutnỳ, and A. Krause, "Coresets via bilevel optimization for continual learning and streaming," in *NeurIPS*, 2020.

[152] N. K. Chada, C. Schillings, X. T. Tong, and S. Weissmann, "Consistency analysis of bilevel data-driven learning in inverse problems," *arXiv:2007.02677*, 2020.

[153] F. Yousefian, "Bilevel distributed optimization in directed networks," *arXiv:2006.07564*, 2020.

[154] Y. Liu, Y. Su, A.-A. Liu, B. Schiele, and Q. Sun, "Mnemonics training: Multi-class incremental learning without forgetting," in *CVPR*, 2020.

[155] O. Litany and D. Freedman, "Soseleto: A unified approach to transfer learning and training with noisy labels," in *ICLR*, 2018.

[156] K. P. Bennett, G. Kunapuli, J. Hu, and J.-S. Pang, "Bilevel optimization and machine learning," in *IEEE World Congress on Computational Intelligence*, 2008, pp. 25–47.

[157] N. Couellan and W. Wang, "Bi-level stochastic gradient for large scale support vector machine," *Neurocomputing*, vol. 153, 2015.

[158] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*, 2013.

[159] A. G. Baydin and B. A. Pearlmutter, "Automatic differentiation of algorithms for machine learning," *arXiv:1404.7456*, 2014.

[160] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 131–159, 2002.

[161] M. W. Seeger, "Cross-validation optimization for large scale structured classification kernel methods," *The Journal of Machine Learning Research*, vol. 9, no. Jun, pp. 1147–1178, 2008.

[162] L. Calatroni, C. Cao, J. C. De Los Reyes, C.-B. Schönlieb, and T. Valkonen, "Bilevel approaches for learning of variational imaging models," *Variational Methods: In Imaging and Geometric Control*, vol. 18, no. 252, p. 2, 2017.

[163] J. Lorraine and D. Duvenaud, "Stochastic hyperparameter optimization through hypernetworks," *arXiv:1802.09419*, 2018.

[164] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to Learn*, 1998.

[165] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv:1706.05098*, 2017.

[166] M. Zhao, B. An, Y. Yu, S. Liu, and S. J. Pan, "Data poisoning attacks on multi-task relationship learning." in *AAAI*, 2018.

[167] F. Alesiani, S. Yu, A. Shaker, and W. Yin, "Towards interpretable multi-task learning using bilevel programming," *arXiv:2009.05483*, 2020.

[168] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *NeurIPS*, 2016.

[169] S. Flennerhag, A. A. Rusu, R. Pascanu, H. Yin, and R. Hadsell, "Meta-learning with warped gradient descent," *arXiv:1909.00025*, 2019.

[170] M.-F. Balcan, M. Khodak, and A. Talwalkar, "Provable guarantees for gradient-based meta-learning," in *ICML*, 2019.

[171] Y. Lee and S. Choi, "Meta-learning with adaptive layerwise metric and subspace," in *ICML*, 2017.

[172] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv:1808.05377*, 2018.

[173] F. P. Casale, J. Gordon, and N. Fusi, "Probabilistic neural architecture search," *arXiv:1902.05116*, 2019.

[174] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *AutoML*, 2016.

[175] J. Wang, J. Wu, H. Bai, and J. Cheng, "M-nas: Meta neural architecture search." in *AAAI*, 2020.

[176] J. Kim, S. Lee, S. Kim, M. Cha, J. K. Lee, Y. Choi, Y. Choi, D.-Y. Cho, and J. Kim, "Auto-meta: Automated gradient based meta learner search," *arXiv:1806.06927*, 2018.

[177] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu, "V-nas: Neural architecture search for volumetric medical image segmentation," in *3DV*, 2019.

[178] T. Pang, X. Yang, Y. Dong, K. Xu, H. Su, and J. Zhu, "Boosting adversarial training with hypersphere embedding," *arXiv:2002.08619*, 2020.

[179] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[180] X. Wang, H. Chen, J. Wu, Y. Ding, Q. Lou, and S. Liu, "Bi-level multi-agents interactive decision-making model in regional integrated energy system," in *Energy Internet and Energy System Integration*, 2019, pp. 2103–2108.

[181] Z. Xu, H. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," *arXiv:1805.09801*, 2018.

[182] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-based reinforcement learning via meta-policy optimization," in *Conference on Robot Learning*, 2018, pp. 617–629.

[183] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv:1611.05763*, 2016.

[184] S. Arora, S. S. Du, S. Kakade, Y. Luo, and N. Saunshi, "Provable representation learning for imitation learning via bi-level optimization," *arXiv:2002.10544*, 2020.

[185] M. Pan, C. Jian, R. Liu, F. Xin, and Z. Luo, "Learning bilevel layer priors for single image rain streaks removal," *IEEE Signal Processing Letters*, vol. 26, no. 2, pp. 307–311, 2019.

[186] L. Ma, Q. Sun, B. Schiele, and L. Van Gool, "A novel bilevel paradigm for image-to-image translation," *arXiv:1904.09028*, 2019.

[187] J. Xu, L. Luo, C. Deng, and H. Huang, "Bilevel distance metric learning for robust image recognition," in *NeurIPS*, 2018.

[188] S. Zha, T. N. Pappas, and D. L. Neuhoff, "Hierarchical bilevel image compression based on cutset sampling," in *ICIP*, 2012.

[189] R. Liu, P. Mu, X. Yuan, S. Zeng, and J. Zhang, "A generic descent aggregation framework for gradient-based bi-level optimization," in *ICML*, 2021.

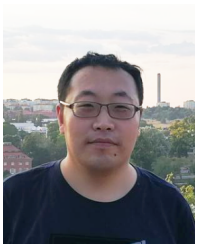
[190] S. Jenni and P. Favaro, "Deep bilevel learning," in *ECCV*, 2018.

[191] R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo, "On the iteration complexity of hypergradient computation," in *ICML*, 2020.

[192] S. Dempe, J. Dutta, and B. Mordukhovich, "New necessary optimality conditions in optimistic bilevel programming," *Optimization*, vol. 56, no. 5-6, pp. 577–604, 2007.

[193] B. Kohli, "Optimality conditions for optimistic bilevel programming problem using convexifactors," *Journal of Optimization Theory and Applications*, vol. 152, no. 3, pp. 632–651, 2012.

[194] L. Lampariello and S. Sagratella, "Numerically tractable optimistic bilevel problems," *Computational Optimization and Applications*, vol. 76, no. 2, pp. 277–303, 2020.
[195] R. Liu, X. Liu, X. Yuan, S. Zeng, and J. Zhang, "A value-function-based interior-point method for non-convex bi-level optimization," 2021.
[196] W. Wiesemann, A. Tsoukalas, P.-M. Kleniati, and B. Rustem, "Pessimistic bilevel optimization," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 353–380, 2013.
[197] P. Loridan and J. Morgan, "Weak via strong stackelberg problem: new results," *Journal of Global Optimization*, vol. 8, no. 3, pp. 263–287, 1996.
[198] R. Kicsiny, Z. Varga, and A. Scarelli, "Backward induction algorithm for a class of closed-loop stackelberg games," *European Journal of Operational Research*, vol. 237, no. 3, pp. 1021–1036, 2014.
[199] J. Dutta and T. Pandit, "Algorithms for simple bilevel programming," in *Bilevel Optimization*, 2020, pp. 253–291.
[200] E. Weinan, "A proposal on machine learning via dynamical systems," *Communications in Mathematics and Statistics*, vol. 5, no. 1, pp. 1–11, 2017.
[201] Y. Lu, A. Zhong, Q. Li, and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," in *ICML*, 2018.
[202] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5595–5637, 2017.
[203] B. A. Pearlmutter and J. M. Siskind, "Reverse-mode ad in a functional framework: Lambda the ultimate backpropagator," *ACM Transactions on Programming Languages and Systems*, vol. 30, no. 2, pp. 1–36, 2008.
[204] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *NeurIPS*, 2017.
[205] K. A. Khan and P. I. Barton, "A vector forward mode of automatic differentiation for generalized derivative evaluation," *Optimization Methods and Software*, vol. 30, no. 6, pp. 1185–1212, 2015.
[206] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in julia," *arXiv:1607.07892*, 2016.
[207] J. Luketina, M. Berglund, K. Greff, and T. Raiko, "Scalable gradient-based tuning of continuous regularization hyperparameters," in *ICML*, 2016.
[208] A. G. Baydin, R. Cornish, D. M. Rubio, M. Schmidt, and F. Wood, "Online learning rate adaptation with hypergradient descent," *arXiv:1703.04782*, 2017.
[209] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of maml," in *ICLR*, 2020.
[210] K. Ji, J. D. Lee, Y. Liang, and H. V. Poor, "Convergence of meta-learning with task-specific adaptation over partial parameters," *arXiv:2006.09486*, 2020.
[211] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.
[212] M. Solodov, "An explicit descent method for bilevel convex optimization," *Journal of Convex Analysis*, vol. 14, no. 2, p. 227, 2007.
[213] S. Sabach and S. Shtern, "A first order method for solving convex bilevel optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 640–660, 2017.
[214] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," in *SIATATS*, 2020.
[215] S. Dempe, B. S. Mordukhovich, and A. B. Zemkoho, "Necessary optimality conditions in pessimistic bilevel programming," *Optimization*, vol. 63, no. 4, pp. 505–533, 2014.
[216] A. Tsoukalas, W. Wiesemann, B. Rustem *et al.*, "Global optimisation of pessimistic bi-level problems," *Lectures on Global Optimization*, vol. 55, pp. 215–243, 2009.
[217] A. V. Malyshev and A. S. Strekalovsky, "On global search for pessimistic solution in bilevel problems," *International Journal of Biomedical Soft Computing and Human Sciences: the official journal of the Biomedical Fuzzy Systems Association*, vol. 18, no. 1, pp. 57–61, 2013.
[218] Y. Zheng, G. Zhang, J. Han, and J. Lu, "Pessimistic bilevel optimization model for risk-averse production-distribution planning," *Information Sciences*, vol. 372, pp. 677–689, 2016.
[219] T. Kis, A. Kovács, and C. Mészáros, "On optimistic and pessimistic bilevel optimization models for demand response management," *Energies*, vol. 14, no. 8, p. 2095, 2021.
[220] B. Zeng, "A practical scheme to compute the pessimistic bilevel optimization problem," *INFORMS Journal on Computing*, vol. 32, no. 4, pp. 1128–1142, 2020.
[221] J. Liu, Y. Fan, Z. Chen, and Y. Zheng, "Methods for pessimistic bilevel optimization," in *Bilevel Optimization*, 2020, pp. 403–420.
[222] S. Dempe and J. Dutta, "Is bilevel programming a special case of a mathematical program with complementarity constraints?" *Mathematical Programming*, vol. 131, pp. 37–48, 2012.
[223] D. Ralph and S. Dempe, "Directional derivatives of the solution of a parametric nonlinear program," *Mathematical programming*, vol. 70, no. 1, pp. 159–172, 1995.
[224] J. Y. Jane, "Constraint qualifications and optimality conditions in bilevel optimization," in *Bilevel Optimization*, 2020, pp. 227–251.
[225] P. Borges, C. Sagastizábal, and M. Solodov, "A regularized smoothing method for fully parameterized convex problems with applications to convex and nonconvex two-stage stochastic programming," *Mathematical Programming*, 2020.
[226] X. Zhou, R. Luo, Y. Tu, B. Lev, and W. Pedrycz, "Data envelopment analysis for bi-level systems with multiple followers," *Omega*, vol. 77, pp. 180–188, 2018.
[227] S. Ghadimi and M. Wang, "Approximation methods for bilevel programming," *arXiv:1802.02246*, 2018.
[228] Y. Shehu, P. T. Vuong, and A. Zemkoho, "An inertial extrapolation method for convex simple bilevel optimization," *Optimization Methods and Software*, vol. 36, no. 1, pp. 1–19, 2021.
[229] F. Huang and H. Huang, "Biadam: Fast adaptive bilevel optimization methods," *arXiv:2106.11396*, 2021.
[230] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *ICML*, 2019.
[231] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, no. 99, pp. 1–1, 2020.
[232] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *ICML*, 2018, pp. 4055–4064.



**Risheng Liu** received his B.Sc. (2007) and Ph.D. (2012) from Dalian University of Technology, China. From 2010 to 2012, he was doing research as joint Ph.D. in robotics institute at Carnegie Mellon University. From 2016 to 2018, He was doing research as Hong Kong Scholar at the Hong Kong Polytechnic University. He is currently a full professor with the Digital Media Department at International School of Information Science & Engineering, Dalian University of Technology. He was awarded the "Outstanding Youth Science Foundation" of the National Natural Science Foundation of China. His research interests include optimization, computer vision and multimedia.

**Jiaxin Gao** received the B.S. degree in Applied Mathematics from Dalian University of Technology, China, in 2018. She is currently pursuing the PhD degree in software engineering at Dalian University of Technology, Dalian, China. She is with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian University of Technology, Dalian, China. Her research interests include computer vision, machine learning and optimization.

**Jin Zhang** received the B.A. degree in Journalism from the Dalian University of Technology in 2007. He pursued a degree in mathematics and received the M.S. degree in Operational Research and Cybernetics from the Dalian University of Technology, China, in 2010, and the PhD degree in Applied Mathematics from University of Victoria, Canada, in 2015. After working in Hong Kong Baptist University for 3 years, he joined Southern University of Science and Technology as a tenure-track assistant professor in the Department of Mathematics. His broad research area is comprised of optimization, variational analysis and their applications in economics, engineering and data science.

**Deyu Meng** (Member, IEEE) received the B.Sc. degree in information science, the M.Sc. degree in applied mathematics, and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2001, 2004, and 2008, respectively. He is currently a Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University, and adjunct Professor with the Faculty of Information Technology, The Macau University of Science and Technology, Macao. From 2012 to 2014, he took his two-year sabbatical leave at Carnegie Mellon University, Pittsburgh, PA, USA. His current research interests include model-based deep learning, variational networks, and meta learning.

**Zhouchen Lin** (M'00-SM'08-F'18) received the PhD degree from Peking University in 2000. He is currently a professor with the Key Laboratory of Machine Perception, School of EECS, Peking University. His research interests include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He has been an area chair of CVPR, ICCV, NIPS/NeurIPS, AAAI, IJCAI, ICLR and ICML many times, and is a Program Co-Chair of ICPR 2022. He was an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and currently is an associate editor of the International Journal of Computer Vision. He is a Fellow of IAPR and IEEE.