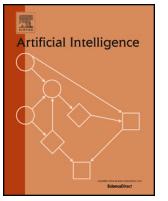
Artificial Intelligence 335 (2024) 104184

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Multi-objective meta-learning

Feiyang Ye [a,b], Baijiong Lin [c], Zhixiong Yue [a,b], Yu Zhang [a,f,*], Ivor W. Tsang [b,d,e]

[a] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
[b] Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, Australia
[c] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
[d] Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore
[e] Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore
[f] Shanghai Artificial Intelligence Laboratory, Shanghai, China

ARTICLE INFO

Keywords:
Meta learning
Multi-objective optimization
Multi-task learning

ABSTRACT

Meta-learning has arisen as a powerful tool for many machine learning problems. With multiple factors to be considered when designing learning models for real-world applications, meta-learning with multiple objectives has attracted much attention recently. However, existing works either linearly combine multiple objectives into one objective or adopt evolutionary algorithms to handle it, where the former approach needs to pay high computational cost to tune the combination coefficients while the latter approach is computationally heavy and incapable to be integrated into gradient-based optimization. To alleviate those limitations, in this paper, we aim to propose a generic gradient-based Multi-Objective Meta-Learning (MOML) framework with applications in many machine learning problems. Specifically, the MOML framework formulates the objective function of meta-learning with multiple objectives as a Multi-Objective Bi-Level optimization Problem (MOBLP) where the upper-level subproblem is to solve several possibly conflicting objectives for the meta-learner. Different from those existing works, in this paper, we propose a gradient-based algorithm to solve the MOBLP. Specifically, we devise the first gradient-based optimization algorithm by alternately solving the lower-level and upper-level subproblems via the gradient descent method and the gradient-based multi-objective optimization method, respectively. Theoretically, we prove the convergence property and provide a non-asymptotic analysis of the proposed gradient-based optimization algorithm. Empirically, extensive experiments justify our theoretical results and demonstrate the superiority of the proposed MOML framework for different learning problems, including few-shot learning, domain adaptation, multi-task learning, neural architecture search, and reinforcement learning. The source code of MOML is available at https://github.com/Baijiong-Lin/MOML.

* Corresponding author at: Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China.
E-mail addresses: feiyang.ye.uts@gmail.com (F. Ye), bj.lin.email@gmail.com (B. Lin), yuezhixiong915@gmail.com (Z. Yue), yu.zhang.ust@gmail.com (Y. Zhang), ivor_tsang@cfar.a-star.edu.sg (I.W. Tsang).
URLs: https://feiyang-ye.github.io (F. Ye), https://baijiong-lin.github.io (B. Lin), https://yuezhixiong.github.io (Z. Yue), https://yuzhanghk.github.io/ (Y. Zhang).

https://doi.org/10.1016/j.artint.2024.104184
Received 1 March 2023; Received in revised form 25 June 2024; Accepted 16 July 2024

Available online 25 July 2024

1. Introduction

In the past few years, deep learning has achieved remarkable success in various fields [54], as it can effectively and efficiently learn from massively high-dimensional data. However, training deep learning models from scratch often requires a large amount of labeled data to learn a large number of model parameters and requires manual selection of hyperparameters, leading to a huge dependence on data volume and the choice of hyperparameters.

Meta-learning has been originally suggested as one efficient strategy to overcome these issues by making models learn how to learn [23,24]. The key idea is that meta-learning agents gain knowledge from multiple meta-training tasks and then quickly reuse such knowledge to learn from new tasks with a handful of training examples. One view of meta-learning is to learn a general purpose learning algorithm that can generalize across tasks, and ideally enable each new task to be learned better. An alternating optimization-based view is to frame meta-learning as a bi-level optimization procedure, where the optimization of the lower subproblem (a.k.a. the inner subproblem) represents an adaptation to a given task with learned meta-parameters, and the objective of the upper subproblem (a.k.a. the outer subproblem) is to learn meta-parameters [23]. From this perspective, meta-learning can be viewed as a procedure to solve a bi-level optimization problem and hence it has broad applications such as hyperparameter optimization [16], Neural Architecture Search (NAS) [38], and Reinforcement Learning (RL) [80]. In this paper, we also take this viewpoint.

Many studies on conventional meta-learning methods and applications only consider one single meta-objective. For example, the Model-Agnostic Meta-Learning (MAML) method [13] has been shown to be applicable to a broad range of meta-learning problems [24], but it only measures the performance on a validation dataset in the upper-level subproblem to evaluate the learned initialization of parameters. However, in real-world applications, more than one objective usually needs to be considered. For example, as an application of meta-learning, few-shot learning may need to consider not only the performance but also the robustness of the learned initialization of parameters, which can help the model adapt to new tasks. Another example is the DARTS method [38] in NAS. This differentiable method only evaluates the performance of the searched architecture on the validation dataset. However, in real-world applications, the network size and performance should be balanced in NAS, especially when the searched architecture will be deployed to resource-constrained devices, such as mobile phones and edge facilities. In those applications, we can see an urgent need to optimize multiple (possibly conflicting) objectives in meta-learning.

Meta-learning with multiple objectives thus has drawn much attention in recent studies. Specifically, some works study specific meta-learning problems in the multi-objective case, such as multi-objective NAS [78,69,2,44], multi-objective RL [5], and so on. However, those works either linearly combine multiple objectives into a single objective for the upper-level subproblem [77,75,12] or utilize multi-objective bi-level evolutionary algorithms [6,64,57] to handle it. The former approach needs to tune weights associated with all the objectives, which is time-consuming, and its performance depends on the set of candidate weights in, for example, the cross validation method. The latter approach, whose computational complexity is even higher, has no convergence guarantee in the optimization process and is not easy to be integrated into gradient-based learning models such as deep neural networks, which limits its use in many learning models.

To alleviate those limitations in existing works, in this paper we propose a unified gradient-based Multi-Objective Meta-Learning (MOML) framework with convergence guarantee. The MOML framework formulates objective functions in meta-learning with multiple objectives as a Multi-Objective Bi-Level optimization Problem (MOBLP), where the lower-level subproblem is to learn the adaptation to a task in a similar way to vanilla meta-learning and the upper-level subproblem minimizes a vector-valued function corresponding to multiple objectives for the meta-learner. To solve MOBLP, we devise the first gradient-based optimization algorithm by alternately solving the lower-level and upper-level subproblems via a gradient descent method and a gradient-based multi-objective optimization method such as [8], respectively. We theoretically prove the convergence properties of the proposed gradient-based optimization method. To show the effectiveness of the MOML framework, we apply it to several meta-learning problems, including few-shot learning, domain adaptation, multi-task learning, neural architecture search, and reinforcement learning.

In summary, the main contributions of this paper are three-fold.

1. We propose a unified MOML framework based on the MOBLP and devise a gradient-based optimization algorithm for the proposed MOML framework.
2. We prove the convergence properties of the proposed optimization algorithm.
3. We formulate several learning problems as instances of the MOML framework and experiments show that MOML achieves state-of-the-art performance on those learning tasks.

A preliminary version of this work appeared in a conference paper [84]. Compared with the conference version, we rephrased the whole paper, added finite-step convergence analysis in Section 5.2, added a new application of the proposed MOML framework to reinforcement learning in Section 6.5, gave more details on experiments from Section 6.1 to Section 6.4, studied the use of another solver for multi-objective optimization in Section 6.4, and analyzed the training cost of the proposed method in Section 7.

The rest of this paper is organized as follows. In Section 2, we review some related works. In Section 3, we introduce the proposed MOML framework. The optimization details of the proposed model are described in Section 4. In Section 5, we present a theoretical analysis of the convergence of the proposed method. In Section 6, we present extensive experimental results to demonstrate the effectiveness of the proposed model. We provide the time complexity analysis for the proposed method in Section 7. Finally, we make conclusions in Section 8.

2. Related work

2.1. Meta-learning

Meta-learning (a.k.a. learning to learn) learns knowledge from multiple tasks and then quickly adapts it to new tasks with a small number of samples. One of the most common applications of meta-learning is Few-Shot Learning (FSL). Meta-learning methods in FSL generally fall into three categories, including metric-based [66,68], model-based [34], and optimization-based [13,50] techniques. For the meta-training process, a widely used approach is to cast as a bi-level optimization problem. To see this, we take MAML, a representative optimization-based method, as an example. In MAML, the meta-training process simulates transfer learning by fine-tuning, so the learned initialization of model parameters is updated to solve each task based on a few examples and a few gradient descent steps. By training the initialization such that all simulated testing tasks fine-tune well, meta-learning produces a good initialization of the model that is easy to adapt to different tasks. Therefore, MAML adapts ω initialized by α to new tasks using the associated training dataset and then updates the initialization α according to the validation performance, where α and ω denote the meta-initialized parameter and the task-specific parameter, respectively. Correspondingly, we also called these two steps as the leader and follower in bi-level optimization problems [39], respectively. Similarly, most meta-learning methods can be formulated as a bi-level optimization problem. Therefore, from this perspective, meta-learning is a general learning paradigm with more applications [23].

2.2. Multi-objective optimization

Multi-objective optimization is to address the problem of simultaneously optimizing several potentially conflicting functions. Multi-objective optimization has wide applications in science and engineering due to the prevalent existence of conflicting objectives or criteria. If the objectives conflict with each other, no unique solution exists that optimizes all of them simultaneously. In such cases, the goal of multi-objective optimization is to find Pareto-optimal solutions. One widely used approach to solve multi-objective optimization problems is scalarizations. These methods convert the original problem to a single-objective optimization problem via the linear scalarization approach which is to linearly combine multiple objectives. Actually, many machine learning algorithms adopt this approach. For example, the training loss and the regularization term used to control the model complexity are two conflicting objectives, and most learning models minimize a linear combination of those two terms. Many other kinds of multi-objective optimization algorithms have been proposed recently, such as evolutionary algorithms [88], population-based algorithms [18], gradient-based algorithms [8,47], and so on. In this work, we focus on gradient-based multi-objective optimization algorithms because this approach can easily be integrated into gradient-based machine learning models such as deep learning.

2.3. Bi-level optimization

Bi-level optimization is a special kind of optimization, where one problem is nested within another one. It has been recognized as a powerful optimization tool for meta-learning methods [23]. The generic Bi-Level optimization Problem (BLP) is formulated as

$$\min_{\alpha\in\mathcal{A},\omega\in\mathbb{R}^p} F(\omega,\alpha) \ \text{ s.t. } \ \omega\in\mathcal{S}(\alpha), \tag{1}$$

where the function F is called the Upper-Level (UL) objective and $\mathcal{S}(\alpha)$ is the solution set of the Lower-Level (LL) subproblem, and α, ω denote the UL and LL variables, respectively.

There have recently been many algorithms [16,40,39] proposed to solve bi-level optimization. However, those existing methods mostly assume that the UL subproblem is a single-objective optimization problem (*i.e.*, $F:\mathbb{R}^p\times\mathbb{R}^n\to\mathbb{R}$) and aim to propose optimization algorithms to maintain convergence properties under different assumptions such as non-convexity [39]. There are only a handful of works for solving Multi-Objective Bi-Level optimization Problems (MOBLP) [65,6,26]. However, all those works solve MOBLP via evolutionary algorithms and analyze it from a game theoretic point of view. In contrast, we are the first to analyze the bi-level multi-objective problem from the optimization perspective and propose a gradient-based algorithm to ensure the convergence with the notion of the Pareto optimality.

3. The MOML framework

To introduce the MOML framework, we first introduce a motivating example whose detailed formulation is in Section 6.1 and then present the formulation of the MOML framework.

3.1. A motivating example

Consider a robust N-way M-shot Few-shot Learning (FSL) problem, where each task is a N-way classification, and it is to learn the meta-initialized parameter such that each task can be solved only with M samples. α and $\omega^{(i)}$ denote the meta-initialized parameter and the task-specific parameter, respectively. For the i-th meta-training episode, we have a meta-training data set $\mathcal{D}^i=\mathcal{D}^{s(i)}\cup\mathcal{D}^{q(i)}$, where $\mathcal{D}^{s(i)}$ and $\mathcal{D}^{q(i)}$ represent the support set and the query set, respectively. We utilize the cross-entropy loss as the task-specific loss and thus the objective of the LL subproblem can be defined as $f(\omega^{(i)},\alpha)=\mathcal{L}_F(\omega^{(i)},\alpha;\mathcal{D}^{s(i)})$, where $\mathcal{L}_F(\cdot,\cdot,\cdot)$ denotes the average

loss on a dataset in terms of the cross-entropy loss function. The original FSL problem only evaluates the learned meta-initialized parameter in terms of the performance on the query set, thus it only has one objective in the UL subproblem. However, if we need to make the FSL model not only have good performance but also be robust to adversarial attack, what can the FSL model do? Such two objectives are also reasonable to make the FSL model possess good generalization performance. To achieve that, we could generate a perturbed query set from the query set in the i-th episode and denote it by $\mathcal{D}^{q(i),adv}$. Different from the original FSL problem, in the UL subproblem, we now need to consider the cross-entropy loss both on the query set $\mathcal{D}^{q(i)}$ and the perturbed query set $\mathcal{D}^{q(i),adv}$. So, we can formulate the UL subproblem as

$$F(\omega^{(i)}, \alpha) = \left(\mathcal{L}_F(\omega^{(i)}, \alpha; \mathcal{D}^{q(i)}), \mathcal{L}_F(\omega^{(i)}, \alpha; \mathcal{D}^{q(i),adv})\right).$$

Since there are two objectives, the UL subproblem is a multi-objective optimization problem. This case has not been studied by existing bi-level optimization in machine learning and it motivates us to propose a more general framework to address such kind of problems.

3.2. The formulation

Therefore, in this paper, the proposed MOML framework has a unified objective function, which is formulated as a Multi-Objective Bi-Level optimization Problem (MOBLP), as

$$\min_{\alpha\in\mathcal{A},\omega\in\mathbb{R}^p} F(\omega,\alpha) = \left(F_1(\omega,\alpha), F_2(\omega,\alpha), ..., F_m(\omega,\alpha)\right)^T \quad \text{s.t.} \quad \omega \in S(\alpha), \tag{2}$$

where function $F : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued jointly continuous function for the m desired meta-objectives and $\mathcal{A}$ is a nonempty compact subset of $\mathbb{R}^n$. The goal of solving problem (2) is to find a Pareto-optimal solution, which is defined in Appendix A. In problem (2), $S(\alpha)$ is defined as the set of optimal solutions to minimize $f(\omega,\alpha)$ *w.r.t.* ω, *i.e.*,

$$S(\alpha) = \arg\min_{\omega} f(\omega,\alpha). \tag{3}$$

When m equals 1, problem (2) reduces to a standard BLP, and hence from this perspective, the MOML framework is a generalization of meta-learning. In problems (2) and (3), F is the UL subproblem and $f : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$ is the LL subproblem. The LL subproblem can be considered as a constraint for the UL subproblem. In MOML, F contains multiple meta-objectives to be achieved for the meta-learner and f defines the objective function for current task such as the training loss. In Section 6, we will see the application of MOML in different learning problems, including few-shot learning, domain adaptation, multi-task learning, NAS, and RL.

To solve problem (2), there exist several works [6,64,57] which adopt multi-objective evolutionary algorithms. However, such methods have a high complexity without convergence guarantee and are not easy to be integrated with gradient-based models such as deep neural networks. Hence, we do not include them in experiments. To the best of our knowledge, there is no gradient-based optimization algorithm with a convergence guarantee to solve an MOBLP, which is what we will do in the next section.

4. Optimization

In this section, we devise a general algorithm to solve the objective function in the proposed MOML framework (*i.e.*, problem (2)).

4.1. Lower-level singleton condition

We now discuss an important assumption for problem (2). Due to the complicated dependency between UL and LL variables, solving it is challenging, especially when optimal solutions of the LL subproblem are not unique.

For a standard BLP with a single objective in the UL subproblem, many studies [9,16,62] potentially require that the LL subproblem only admits a unique minimizer ω for every $\alpha \in \mathcal{A}$, which is formally introduced as the Lower-Level Singleton (LLS) condition in [40]. If the LLS condition does not hold, then for a fixed point α_0, it is unclear which $\omega \in S(\alpha_0)$ should be evaluated in the UL subproblem. Thus, this condition can simplify both the optimization process and convergence analyses.

To solve problem (2) with multiple objectives in the UL subproblem, the LLS condition is necessary. If not, the MOBLP is even ill-defined [10]. Moreover, since the UL objective F is vector-valued, it is more challenging than standard BLP with a scale-valued F to determine which ω should be evaluated in the UL subproblem. Although we can select one specific ω by adding some constraints (*e.g.*, choosing the minimum-norm solution), it is not a general solution and would complicate the problem. Thus, in this work, we use the LLS condition to simplify the analyses.

With the LLS condition, the optimal solution for a given α in the LL subproblem is denoted by ω^*, and then problem (2) can be simplified as

$$\min_{\alpha\in\mathcal{A}} \quad \varphi(\alpha) = F\left(\omega^*(\alpha), \alpha\right) \quad \text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} f(\omega,\alpha). \tag{4}$$

4.2. Gradient-based optimization algorithm

Here we design a general gradient-based optimization algorithm to solve problem (4). Usually, there is no closed form for the solution $\omega^*(\alpha)$ of the LL subproblem and so it is difficult to optimize the UL subproblem directly. Another approach is to use the

optimality condition of the LL subproblem (*i.e.*, $\nabla_\omega f(\omega, \alpha) = 0$) as equality constraints for the UL subproblem in a way similar to [52]. However, this approach only works for LL subproblems with simple forms and cannot work for general learning models.

To solve problem (4), we take a strategy similar to the alternating optimization. In the first part of each iteration (corresponding to steps 3-6 in Algorithm 1), we solve the LL subproblem via gradient descent methods. Specifically, with an initialization ω_0 for the LL variable where the iteration index t is omitted for notation simplicity, the solution of the LL subproblem can be updated for K steps as $\omega_{k+1}(\alpha) = \mathcal{T}_k(\omega_k(\alpha), \alpha), k = 1, \ldots, K-1$, where $\mathcal{T}_k$ represents an operator to update ω. Here we consider a first-order gradient descent method for $\mathcal{T}_k$ such as the Stochastic Gradient Descent (SGD) method and $\mathcal{T}_k$ can be formulated explicitly as $\mathcal{T}_k(\omega_k(\alpha), \alpha) = \omega_k(\alpha) - \mu\nabla_\omega f(\omega_k(\alpha), \alpha)$, where $\mu > 0$ denotes the step size and $\nabla_\omega f(\omega_k(\alpha), \alpha)$ denotes the derivative of f *w.r.t.* ω at $\omega = \omega_k(\alpha)$. In the second part of each iteration (corresponding to steps 7-9 in Algorithm 1), by fixing the value of ω as the current solution $\omega_K(\alpha)$ obtained in the first part, we solve the UL subproblem as

$$\min_\alpha \varphi_K(\alpha) = F\left(\omega_K(\alpha), \alpha\right). \tag{5}$$

Problem (5) is an unconstrained Multi-Objective optimization Problem (MOP) and we can use any multi-objective optimization algorithms to solve it. Here we choose gradient-based multi-objective optimization methods as they can be seamlessly integrated into any gradient-based learning framework. There are several gradient-based multi-objective optimization algorithms [55,8,70] and they commonly find an appropriate descent direction d for all the objectives in F by aggregating their gradients *w.r.t.* α. So such process is denoted by $d = \text{MOPSolver}(\{\nabla_\alpha F_i(\omega_K(\alpha), \alpha)\}_{i=1}^m)$ in Algorithm 1, where the derivative of $F_i(\omega_K(\alpha), \alpha)$ (*w.r.t.* α) can be computed by automatic differentiation techniques. In this paper, we mainly adopt a simple gradient-based MOP method called Multiple Gradient Descent Algorithm (MGDA) [8], whose details are introduced in Appendix A.2. In MGDA, the descent direction d for multiple objectives can be found in the convex hull of the gradients of each objective. Therefore, the descent direction can be determined by $d = \sum_{i=1}^m \gamma_{i,t}\nabla_\alpha F_i(\omega_K(\alpha), \alpha)$, where $\sum_{i=1}^m \gamma_{i,t} = 1$ and $\gamma_{i,t} \geq 0$. Here $\gamma_{i,t}$ could be viewed as the weight of the i-th objective in the t-th iteration, but different from the weighted sum algorithm that uses a fixed weight for each objective, MGDA determines the weights $\{\gamma_{i,t}\}$ by minimizing the ℓ_2 norm of the convex hull of the gradients of each objective in each iteration.

Algorithm 1 Optimization Algorithm for MOML.

Input: numbers of iterations (T, K), step size (μ, ν)
1: Randomly initialized α_0;
2: **for** $t = 1$ **to** T **do**
3: Initialize $\omega_0^t(\alpha_t)$;
4: **for** $j = 1$ **to** K **do**
5: $\omega_j^t(\alpha_t) \leftarrow \omega_{j-1}^t(\alpha_t) - \mu\nabla_\omega f(\omega_{j-1}^t(\alpha_t), \alpha_t)$;
6: **end for**
7: Compute gradients $\nabla_\alpha F_i(\omega_K^t(\alpha_t), \alpha_t)$ for all the i's;
8: Compute the gradient as $d(\omega_K^t(\alpha_t), \alpha_t) = \text{MOPSolver}(\{\nabla_\alpha F_i(\omega_K^t(\alpha_t), \alpha_t)\})$;
9: $\alpha_{t+1} = \alpha_t - \nu_t d(\omega_K^t(\alpha_t), \alpha_t)$ with a step size ν_t;
10: **end for**

The entire algorithm to solve problem (4) is shown in Algorithm 1, which, to the best of our knowledge, is the first gradient-based optimization algorithm for MOBLPs.

5. Theoretical analysis

Algorithm 1 is simple and intuitive, but its convergence properties are unclear. In this section, we provide convergence analyses for the approximated problem (5) and Algorithm 1 under certain assumptions.

As an MOBLP, problem (4) cannot be reduced to a BLP with a scalar-valued objective function in the upper-level subproblem when using Algorithm 1 to solve it. Therefore, it has different theoretical properties from BLP as we need to focus on the convergence properties of a minimal point set instead of a minimum scalar in the UL subproblem.

We first recall some notions about vector-valued functions. Consider a vector-valued function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ ($m, n \in \mathbb{N}, m \geq 2$). We denote by Min $g(z)$ the set of all the minimal points of the function $g(z)$. Min $g(z)$ is also called the Pareto frontier or Pareto-optimal set. The corresponding efficient solution or Pareto-optimal solution set of $g(z)$ is denoted by Eff(g(z)). The convexity of vector-valued functions is called the P-convex. The details of these definitions can be found in Appendix A.

5.1. Convergence properties

To analyze the convergence properties of problem (5), we first make the following assumptions.

Assumption 1. We assume that (i) the set $\mathcal{A}$ is a nonempty compact subset of $\mathbb{R}^n$; (ii) the functions $f(\omega, \alpha)$ and $F(\omega, \alpha)$ are both jointly continuous functions; (iii) $\arg\min_\omega f(\omega, \alpha)$ is a singleton for every $\alpha \in \mathcal{A}$; (iv) $\omega^*(\alpha)$ is uniformly bounded on $\mathcal{A}$.

Note that the third assumption in Assumption 1 is the LLS condition introduced in Section 4.1 and is widely adopted in BLPs [62,11]. With Assumption 1, we can obtain the following result.

Theorem 1. *Suppose Assumption 1 holds, then the function $F(\omega^*(\alpha), \alpha)$ is continuous w.r.t. α.*

The proof of Theorem 1 is provided in Appendix B.1. Because $\mathcal{A}$ is a compact set, Theorem 1 implies the existence of solutions. Theorem 1 and the uniform convergence of $\omega_K(\alpha)$ can further imply the convergence for the solution of the LL subproblem, which is similar to that of the standard BLP problem [16].

For the convergence of Algorithm 1, we need to analyze minimal point sets of the images of perturbed functions $\varphi_K(\alpha)$ and $\varphi(\alpha)$ which are defined in problems (5) and (4), respectively. We consider the most natural set convergence under this setting, *i.e.*, the Kuratowski-Painlevé set-convergence. Please refer to these definitions in Appendix A. Under certain assumptions that are used in analyses of BLP and MOP [16,46], we have the following convergence results.

Theorem 2. *In addition to Assumption 1, it is assumed that (i) The iterative sequence $\{\omega_k(\alpha)\}_{k=1}^K$ converges uniformly to $\omega^*(\alpha)$ on $\mathcal{A}$ as $K \to +\infty$; (ii) $\mathcal{A}$ is a convex set; (iii) φ_K is P-convex and φ is strictly P-convex. Then, the Kuratowski-Painlevé set-convergence of both the minimal point set and efficient solution set in Algorithm 1 holds, i.e.,*

$$\operatorname{Min} \varphi_K(\alpha) \to \operatorname{Min} \varphi(\alpha), \ \operatorname{Eff} \varphi_K(\alpha) \to \operatorname{Eff} \varphi(\alpha).$$

The proof of Theorem 2 is provided in Appendix B.2. Theorem 2 implies that, under some specific assumptions, both the minimal point set and solution set of the objective $\varphi_K(\alpha)$ will converge to that of the objective $\varphi(\alpha)$ as $K \to +\infty$. Thus, Theorem 2 provides a theoretical justification our proposed approximation procedure (5).

5.2. Finite-step convergence

In Theorem 2, we have proved the convergence of Algorithm 1 when $K \to +\infty$. In practice, K only takes a finite value to reduce the computational cost. For example, K is set to 5 or 10 in few-shot learning methods (e.g., MAML [13] and BOIL [51]) and is set to 1 in neural architecture search methods (e.g., DARTS [38]). To further analyze how K and T affect the convergence of Algorithm 1, in this section, we provide a finite-step convergence analysis in a concrete case, where gradient descent is used to update ω and MGDA is used as a MOPSolver to update α.

To analyze the finite-step convergence of Algorithm 1, we first make the following assumptions.

Assumption 2. We assume that (i) the function $f(\omega, \alpha)$ is ϑ-strongly-convex w.r.t. ω; (ii) the functions ∇F_i are $\hat{c}_i$-Lipschitz continuous; (iii) the gradient functions ∇f and ∇F_i are Lipschitz function; (iv) the functions $\nabla_\alpha \nabla_\omega f$ and $\nabla^2_\omega f$ are Lipschitz continuous.

Note that the strongly convexity assumption in Assumption 2 can ensure that the LL subproblem satisfies the LLS condition and is commonly used in the analysis for the BLPs [15,16].

In Algorithm 1, it first runs K steps of gradient descent to find an approximation point ω_K, and then calculates the gradient of φ w.r.t. the UL variable α. For the i-th entry of φ, Algorithm 1 computes the gradient $\frac{\partial F_i(\omega_K(\alpha), \alpha)}{\partial \alpha}$ as an approximation of the true hyper-gradient that is computed as

$$\nabla \varphi_i(\alpha) = \frac{\partial F_i(\omega^*(\alpha), \alpha)}{\partial \alpha} = \nabla_\alpha F_i\left(\omega^*(\alpha), \alpha\right) + \frac{\partial \omega^*(\alpha)}{\partial \alpha} \nabla_\omega F_i\left(\omega^*(\alpha), \alpha\right). \tag{6}$$

Such approximation causes an estimation error in each iteration of the outer loop. With Assumption 2, we provide the following theorem to analyze this estimation error.

Theorem 3. *Suppose Assumption 2 holds, then if the step-size $\mu \le 1/\vartheta$ in Algorithm 1, we have*

$$\left\|\nabla_\alpha F_i(\omega_K(\alpha), \alpha) - \nabla_\alpha \varphi_i(\alpha)\right\| \le \left(c_1(1-\mu\vartheta)^{\frac{K}{2}} + c_2(1-\mu\vartheta)^{\frac{K-1}{2}}\right)\|\omega^0 - \omega^*(\alpha)\| + c_3(1-\mu\vartheta)^K, \tag{7}$$

where $\|\cdot\|$ denotes the ℓ_2 norm of a vector, ω^0 is the initialization of ω in the inner loop, and c_1, c_2, c_3 are constants, which rely on the Lipschitz constants.

The proof of Theorem 3 is provided in Appendix B.3. Theorem 3 shows that the gradient estimation error $\left\|\nabla_\alpha F_i(\omega_K(\alpha), \alpha) - \nabla_\alpha \varphi_i(\alpha)\right\|$ decays exponentially w.r.t. the number of steps in the inner loop. For notation simplicity, we denote by $\Gamma(K)$ the right-hand side of the inequality (7). Therefore, according to Theorem 3, we have $\Gamma(K) \to 0$ as $K \to +\infty$ if the step size of the inner loop satisfies $\mu \le 1/\vartheta$.

In each iteration of the outer loop, the convex combination coefficients are determined by the MGDA method. We denote by $\widetilde{\gamma}_t$ the convex combination vector determined by the estimated gradients $\nabla_\alpha F_i(\omega_K(\alpha), \alpha)$ in the t-th iteration. We denote by γ_t the corresponding convex combination vector calculated by the true gradients $\nabla \varphi_i(\alpha_t)$ in the t-th iteration. Then, the solution sequence $\{\alpha_t\}$ generated by MGDA method in Algorithm 1 can be formulated as $\alpha_{t+1} = \alpha_t - \nu \Lambda(\nabla_\alpha F_i(\omega_K(\alpha), \alpha), \widetilde{\gamma}_t)$, where the function $\Lambda(\cdot, \cdot)$ denotes a combination of the first argument weighted by the second argument, *i.e.*, $\Lambda(\varphi, \gamma) = \sum_{i=1}^m \gamma_i \varphi_i$. Then we have the following theoretical result for the sequence $\{\alpha_t\}$.

Theorem 4. *Suppose Assumption 2 holds, then for $1 \le i \le m$, if the i-th entry of the function φ_K is c_i-strongly-convex, the i-th entry of the function φ is L_i-Lipschitz continuous, the step size ν_t in the outer loop of Algorithm 1 equals $\frac{2}{c(t+1)}$, and $\mu \le 1/\vartheta$, then for any point $\alpha^* \in \mathcal{A}$, we have*

$$\min_{t=1,\dots,T} \Lambda\left(\varphi_K(\alpha_t), \widetilde{\gamma}_t\right) - \Lambda\left(\varphi_K(\alpha^*), \bar{\gamma}_t\right) \le \frac{4L^2 + 4\Gamma(K)^2}{c(T+1)}, \tag{8}$$

holds for the sequence $\{\alpha_t\}_{t=1}^T$, where $\bar{\gamma}_t = \frac{\sum_{t=1}^T t\widetilde{\gamma}_t}{\sum_{t=1}^T t}$, $L = \max_{1\le i\le m} L_i$, and $c = \min_{1\le i\le m} c_i$.

The proof of Theorem 4 is provided in Appendix B.4. For any initialization, in Algorithm 1 we have a sequence of simplex vectors $\{\widetilde{\gamma}_t\}_{t=1}^T$ by the MGDA method as the MOP solver. This sequence is bounded and it has one limit point denoted by $\widetilde{\gamma}^*$. Since each entry of the function φ_K is strongly-convex, the weighted objective $\Lambda(\varphi_K(\alpha), \gamma)$ has only one minimizer for any $\gamma \in \Delta^{m-1}$, where Δ^{m-1} denotes an $(m-1)$-dimensional simplex. Let α^* be the unique solution of the objective $\Lambda(\varphi_K(\alpha), \widetilde{\gamma}^*)$. Then α^* is a Pareto-optimal solution associated with the vector $\widetilde{\gamma}^*$. Since the sequence $\bar{\gamma}_t$ converges to $\widetilde{\gamma}^*$, Theorem 4 implies that $\min_{t=1,\dots,T} \Lambda(\varphi_K(\alpha_t), \widetilde{\gamma}_t)$ converges to the Pareto-optimal solution $\Lambda(\varphi_K(\alpha^*), \widetilde{\gamma}^*)$ with the convergence rate as $1/T$.

Theorem 4 only compares $\{\alpha_t\}$ with the Pareto-optimal solution of the approximation objective function φ_K. To fully analyze the relation between $\{\alpha_t\}$ and the Pareto-optimal solution of the original objective function φ, we have the following theorem.

Theorem 5. *Let α^* be the Pareto-optimal solution of $\varphi_K(\alpha)$ corresponding to the limit point $\widetilde{\gamma}^*$ of the sequence $\{\widetilde{\gamma}_t\}$. With the assumptions in Assumption 2 and Theorem 4, if the i-th entry of the function φ is $\hat{c}_i$-strongly-convex and $\mu \le 1/\vartheta$, we have*

$$\left\|\Lambda\left(\varphi(\bar{\alpha}^*), \widetilde{\gamma}^*\right) - \Lambda\left(\varphi_K(\alpha^*), \widetilde{\gamma}^*\right)\right\| \le \frac{\delta L}{\hat{c}}\Gamma(K) + (1-\mu\vartheta)^K \hat{L}\|\omega^0 - \omega^*(\alpha_t)\|, \tag{9}$$

where $\bar{\alpha}^*$ is the Pareto-optimal solution of $\varphi(\alpha)$ corresponding to the weighted vector $\widetilde{\gamma}^*$, δ is the diameter of the bounded set $\mathcal{A}$, $\hat{L} = \max_{1\le i\le m} \hat{L}_i$, and $\hat{c} = \min_{1\le i\le m} \hat{c}_i$.

The proof of Theorem 5 is provided in Appendix B.5. Theorem 5 implies that for the limit point of the sequence $\{\widetilde{\gamma}_t\}$, the distance between the corresponding Pareto-optimal points in $\mathrm{Min}(\varphi(\alpha))$ and $\mathrm{Min}(\varphi_K(\alpha))$ decays exponentially w.r.t. K.

To analyze the convergence of $\Lambda(\varphi_K(\alpha_t), \widetilde{\gamma}_t)$ to the optimal minimal point $\Lambda(\varphi(\bar{\alpha}^*), \widetilde{\gamma}^*)$, based on Theorem 5 as well as an assumption that the generated sequence $\{\widetilde{\gamma}_t\}$ approximates well the limit point $\widetilde{\gamma}^*$ in the Pareto-optimal set $\mathrm{Min}(\varphi_K(\alpha))$, we have the following theorem.

Theorem 6. *Let α^* be the Pareto-optimal solution of $\varphi_K(\alpha)$ corresponding to the limit point $\widetilde{\gamma}^*$ of the sequence $\{\widetilde{\gamma}_t\}$. With the assumptions in Assumption 2 and Theorem 4-5, we also assume that $\Lambda(\nabla\varphi_K(\alpha^*), \widetilde{\gamma}_t)^\top(\alpha_t - \alpha^*) \ge 0$ and the i-th entry of the function φ_K is M_i-Lipschitz. Then if the step size ν_t in the outer loop of Algorithm 1 satisfies $\nu_t = \xi/t$ where $\xi \ge 1/2c$, and $\mu \le 1/\vartheta$, we have*

$$\left\|\Lambda\left(\varphi_K(\alpha_t), \widetilde{\gamma}^*\right) - \Lambda\left(\varphi(\bar{\alpha}^*), \widetilde{\gamma}^*\right)\right\| \le \frac{\delta L}{\hat{c}}\Gamma(K) + (1-\mu\vartheta)^K \hat{L}\|\omega^0 - \omega^*(\alpha_t)\| + \frac{\max\{2\xi LM(2c\xi-1)^{-1}, M\|\alpha^0 - \alpha^*\|^2\}}{t}, \tag{10}$$

where α^0 is the initialization of α in the inner loop, $\bar{\alpha}^*$ is the Pareto-optimal solution of $\varphi(\alpha)$ corresponding to the weighted vector $\widetilde{\gamma}^*$, and $M = \max_{1\le i\le m} M_i$.

The proof of Theorem 6 is provided in Appendix B.6. Theorem 6 gives the finite-step convergence result, which depends on both numbers of steps in the inner and outer loops (*i.e.*, K and T). It demonstrates the efficacy of the proposed algorithm. When $K \to +\infty$, the error bound has a $\mathrm{O}(1/t)$ convergence rate that matches the original MGDA algorithm [14].

6. Applications of MOML

In this section, we introduce applications of the proposed MOML framework in several learning problems, including few-shot learning, semi-supervised domain adaptation, multi-task learning, neural architecture search, and reinforcement learning. All the experiments are conducted on one single NVIDIA Tesla V100S GPU.

A simple baseline: SOML Before presenting different applications of the MOML framework, we introduce a simple baseline method used in experiments. For an MOBLP such as problem (2), a common approach is to transform the UL subproblem into a single objective problem via the linear scalarization approach [77,75,12]. Specifically, with fixed weights $\{\gamma_i\}_{i=1}^m$, which satisfy $\sum_{i=1}^m \gamma_i = 1$ and $\gamma_i \ge 0$, problem (2) is transformed to a standard bi-level optimization problem as

$$\min_{\alpha\in\mathcal{A}, \omega\in\mathbb{R}^p} \sum_{i=1}^m \gamma_i F_i(\omega, \alpha) \quad \text{s.t.} \quad \omega \in S(\alpha), \tag{11}$$

which has a scalar-valued function $\sum_{i=1}^m \gamma_i F_i(\omega, \alpha)$ as the UL subproblem. In contrast with MOML, this approach is named as Single-Objective Meta-Learning (SOML) in this paper and it will be used as an extra baseline method in our experiments. In SOML, we need to carefully tune $\{\gamma_i\}$ with the cost to search them increasing exponentially with respect to m.

6.1. Few-shot learning

Few-Shot Learning (FSL) aims to tackle the problem of training a model with only a few training samples [76]. Recently, FSL is widely studied from the perspective of meta-learning by using prior knowledge in the meta-training process. Most studies in FSL only consider the classification performance. However, in real-world applications, the performance is not the only important factor to consider. For example, we expect FSL models to not only have good performance but also be robust to adversarial attacks [32], which may help improve the generalization of FSL models. In the following, we can see that this setting can naturally be modeled by the proposed MOML framework.

6.1.1. Problem formulation

Suppose there is a base dataset $\mathcal{D}_{base}$ with a category set $\mathcal{C}_{base}$ and a novel dataset $\mathcal{D}_{novel}$ with a category set $\mathcal{C}_{novel}$, where $\mathcal{C}_{base} \cap \mathcal{C}_{novel} = \emptyset$. The goal of FSL is to adapt the knowledge learned from $\mathcal{D}_{base}$ to help the learning of $\mathcal{D}_{novel}$. In the i-th meta-training episode, we generate from $\mathcal{D}_{base}$ an N-way k-shot classification task, which consists of a support set $\mathcal{D}_{base}^{s(i)}$ and a query set $\mathcal{D}_{base}^{q(i)}$. For the robustness, we add perturbations generated by the Projected Gradient Descent (PGD) method [32] into each data point in $\mathcal{D}_{base}^{q(i)}$ to generate a perturbed query set $\mathcal{D}_{base}^{q(i),adv}$. The objective function of the FSL model that considers both the performance and the robustness can be formulated as

$$\min_{\alpha} \left(\mathcal{L}_F(\omega^{*(i)}(\alpha), \alpha, \mathcal{D}_{base}^{q(i)}), \mathcal{L}_F(\omega^{*(i)}(\alpha), \alpha, \mathcal{D}_{base}^{q(i),adv})\right) \quad \text{s.t.} \quad \omega^{*(i)}(\alpha) = \arg\min_{\omega} \mathcal{L}_F(\omega, \alpha, \mathcal{D}_{base}^{s(i)}), \tag{12}$$

where ω represents model parameters, α denotes meta-parameters to encode common knowledge that can be transferred to novel tasks, and $\mathcal{L}_F(\omega, \alpha, \mathcal{D})$ denotes the average classification loss of a model with model parameters α and meta-parameters ω on a dataset $\mathcal{D}$. In the UL subproblem of problem (12), the first objective measures the classification loss on the query set based on $\omega^{*(i)}(\alpha)$ obtained by solving the LL subproblem and the second objective measures the robustness via the classification performance on the perturbed query set. Problem (12) is a general formulation for a robust FSL model under the MOML framework, which depends on what α and ω represent.

We adapt problem (12) to three representative FSL methods, including MAML [13], BOIL [51], and ProtoNet [66]. Specifically, we present the definition of model parameters α, meta-parameters ω, and the classification loss $\mathcal{L}_F(\omega, \alpha, \mathcal{D})$ of problem (12) based on these algorithms to show how to adapt the MOML framework to them.

MAML is an optimization-based meta-learning algorithm. In MAML, α represents the meta-initialized parameter and ω represents the task-specific parameter. In this paper, we focus on classification tasks and so the loss function $\mathcal{L}_F(\omega, \alpha, \mathcal{D})$ adopts the cross-entropy loss with α and ω on a dataset $\mathcal{D}$. Given the meta-initialized parameter α of the backbone, we use α to initialize task-specific parameters ω_0 and update ω_0 to ω_K in K steps on the support set for the LL subproblem. Then, we compute the loss on the query set in the UL subproblem by using ω_K for the corresponding task. Thus, we can update α and find universally good meta-initialized parameters that can quickly adapt to new tasks with a small number of samples. In problem (12), we aim to find the meta-initialized parameters α that not only have good performance but also be robust to adversarial attacks when adapting to new tasks with a few examples.

BOIL is also an optimization-based meta-learning algorithm. Similar to MAML, BOIL aims to find universally good meta-initialized parameters. However, BOIL updates only the feature extractor of the model and freezes the classification layer in the LL subproblem. In the UL subproblem, BOIL updates the meta-initialized parameters of the feature extractor and classification layer, which is the same as MAML. Specifically, let $\alpha = \{\alpha_\theta, \alpha_\psi\}$, where α_θ denotes the meta-initialized parameter of the task-specific feature extractor θ and α_ψ denotes the meta-initialized parameter of the task-specific classifier parameter ψ, respectively. In BOIL, we update α in the UL subproblem and only update $\omega = \{\theta\}$ in the LL subproblem.

ProtoNet is a metric-based FSL algorithm. In ProtoNet, α represents the parameter of the embedding function $f(x; \alpha)$, which encodes inputs into a vector space $\mathcal{V}$. ω^k represents the prototype of the k-th class, which can be considered as a class center. Suppose there are n labeled examples in dataset D and C is the number of classes. Then, given a squared Euclidean distance function for $\mathcal{V}$, the conditional probability of an example belonging to a class is formulated as

$$P(y = k \mid x; \alpha, \omega) = \frac{\exp(-\|f(x; \alpha) - \omega^k\|^2)}{\sum_{k'=1}^{C} \exp(-\|f(x; \alpha) - \omega^{k'}\|^2)}, \tag{13}$$

where ω represents the set of all prototypes. Suppose that $\mathcal{D}_k \subset \mathcal{D}$ is the set of examples labeled with class k and n_k is the corresponding number of examples. Then, the loss function is the negative log-likelihood

$$\mathcal{L}_F(\omega, \alpha, \mathcal{D}) = -\frac{1}{n} \sum_{k=1}^{C} \sum_{x_i \in D_k} \log P(y_i = k \mid x_i; \alpha, \omega), \tag{14}$$

where y_i denotes the class label for x_i. Different from optimization-based meta-learning algorithm such as MAML, in the LL subproblem, by minimizing this loss function *w.r.t.* ω, we have a closed-form solution $\omega^k = \frac{1}{n_k} \sum_{x_i \in \mathcal{D}_k} f(x_i; \alpha)$. In the UL subproblem, we minimize this loss *w.r.t.* α on the clean and perturbed query sets.

Table 1
Classification accuracy (abbreviated as "Clean Acc.") and PGD accuracy (abbreviated as "PGD Acc.") on the *mini*-ImageNet dataset and CUB dataset for 5-way k-shot FSL. The best result in each group of methods is highlighted in **bold** and the best result in each setting is annotated with underline.

	Method	*mini*-ImageNet			CUB		
		Clean Acc.	PGD Acc.	B-score	Clean Acc.	PGD Acc.	B-score
1-shot	MAML [13]	45.24±0.81	1.18±0.15	2.10±0.52	54.62±0.87	3.92±0.49	7.28±0.61
	MAML+SOML	40.78±0.75	23.91±0.67	29.83±0.43	49.60±0.81	36.42±0.87	41.89±0.84
	MAML+MOML	39.23±0.76	25.80±0.67	**31.12**±0.70	48.66±0.87	38.37±0.90	**42.75**±0.89
	BOIL [51]	47.64±0.85	3.05±0.22	5.53±0.61	61.79±0.94	6.53±0.48	11.81±0.61
	BOIL+SOML	40.44±0.79	25.94±0.69	31.29±0.75	54.29±0.83	33.65±0.67	41.34±0.71
	BOIL+MOML	41.22±0.83	27.77±0.75	**32.98**±0.79	52.15±0.93	40.44±0.94	**45.55**±0.94
	ProtoNet [66]	44.67±0.75	2.60±0.21	3.73±0.35	52.93±0.91	2.00±0.29	3.53±0.47
	ProtoNet+SOML	38.65±0.72	23.10±0.65	28.67±0.67	48.04±0.91	28.53±0.85	35.42±0.90
	ProtoNet+MOML	35.06±0.70	27.24±0.65	**30.51**±0.66	42.26±0.89	32.19±0.82	**36.24**±0.85
5-shot	MAML [13]	61.88±0.77	2.82±0.21	5.01±0.43	75.57±0.72	8.76±0.78	15.61±0.76
	MAML+SOML	56.16±0.72	34.85±0.72	42.91±0.71	68.50±0.69	52.96±0.87	59.63±0.77
	MAML+MOML	55.66±0.78	39.38±0.77	**45.89**±0.77	67.57±0.78	55.26±0.87	**60.68**±0.83
	BOIL [51]	66.02±0.72	4.85±0.30	1.11±0.51	78.97±0.67	14.12±0.57	23.75±0.60
	BOIL+SOML	58.54±0.76	34.28±0.75	42.94±0.78	76.25±0.60	44.86±0.81	56.28±0.73
	BOIL+MOML	60.21±0.79	35.47±0.78	**44.37**±0.78	71.03±0.74	56.05±0.84	**62.65**±0.81
	ProtoNet [66]	66.55±0.70	0.68±0.09	1.31±0.18	78.01±0.71	1.89±0.21	3.58±0.56
	ProtoNet+SOML	59.11±0.71	39.41±0.73	46.93±0.71	72.51±0.68	52.61±0.77	60.81±0.72
	ProtoNet+MOML	58.72±0.74	41.59±0.75	**48.59**±0.74	71.10±0.74	56.11±0.87	**62.73**±0.76

6.1.2. Experimental settings

Experiments in FSL are conducted on two FSL benchmark datasets, *i.e.*, *mini*-ImageNet [73] and CUB-200-2011 (referred to as CUB) [74]. The *mini*-ImageNet dataset contains 100 classes with 600 images per class, sampled from the ImageNet dataset [7]. By following [56], this dataset is partitioned into 64, 16, and 20 classes for the base, validation, and novel datasets, respectively. The CUB dataset contains 200 classes and 11,788 images in total. Following [22], we randomly split this dataset into 100, 50, and 50 classes for the base, validation, and novel datasets, respectively.

For all methods, each task is a 5-way k-shot classification problem, where k equals 1 or 5. The input images are resized to 84×84 for both two datasets and applied data augmentation including random crop, random horizontal flip, and color jitter. A four-layer convolutional neural network (Conv-4) is used as the backbone, which consists of four blocks each of which consists of a convolution layer with 64 kernels of size 3×3, stride 1, and zero padding, a batch normalization layer, a ReLU activation function, and a max-pooling layer with the pooling size 2×2. After the backbone, a fully-connected linear layer with 5 neurons is used as a classifier to output the prediction for the input image. The Adam optimizer [29] with the learning rate 0.001 is used for the optimization. Following [13], we set the number of LL iterations to be 5 (i.e., $K = 5$) for the MAML-based and BOIL-based methods.

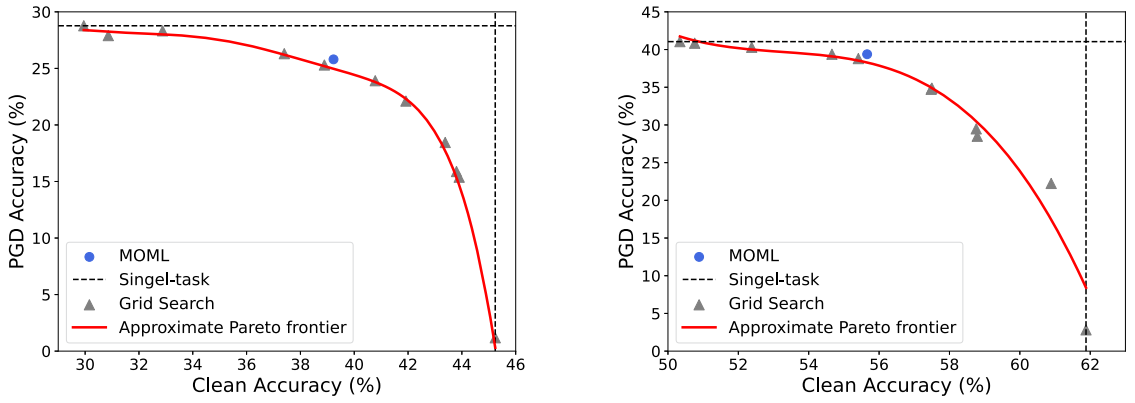
In the meta-training process, we randomly sample k and 16 instances per class as the support set and the query set, respectively, in each episode. The adversarial attack on the query set is performed by the PGD attack with a perturbation size $\epsilon = 2/255$ and it takes 7 iterative steps with the step size of 2.5ϵ. In the meta-testing process, we generate 600 5-way k-shot tasks from D_{novel}, where each task has k samples for the training and 16 samples for testing. We compute the average results on all the 600 testing tasks. We also compare with the vanilla FSL models (*i.e.*, MAML, ProtoNet, and BOIL) since the problem (12) can be reduced to each of them when the second objective in the UL subproblem vanishes.

The classification accuracy and PGD accuracy are used to measure the performance. As the clean accuracy and the PGD accuracy are reported to be conflicting with each other [83], inspired by the widely-used F1-score based on precision and recall, we propose a new metric called Balance score (B-score), which is defined as $\text{B-score} = 2 \times (\text{CA} \times \text{PA})/(\text{CA} + \text{PA})$ with CA and PA denoting the clean accuracy and PGD accuracy, respectively, to fully measure the performance in terms of the clean accuracy and PGD accuracy.

6.1.3. Experimental results

The average results over testing tasks on the *mini*-ImageNet and CUB datasets are presented in Table 1. From the results, we can see that the adversarial robustness of the original FSL methods (*i.e.*, MAML, ProtoNet, and BOIL) is poor, while SOML and the proposed MOML can significantly improve the PGD accuracy. For example, MOML can improve the PGD accuracy of ProtoNet by about 40.91% under the 5-shot setting on the *mini*-ImageNet dataset. Since the classification accuracy and adversarial robustness are conflicting [83], the clean accuracy of SOML and MOML slightly drops when compared with the original FSL baselines. The B-score of MOML is higher than SOML, which indicates the multi-objective formulation in the UL subproblem of MOML is better than the single-objective formulation in SOML.

To demonstrate that MOML can achieve a nearly Pareto-optimal solution, we conduct experiments on the *mini*-ImageNet dataset under 5-way 1-shot and 5-way 5-shot FSL settings. The results are shown in Fig. 1. The grey triangles denote approximated Pareto-optimal solutions computed by SOML with different combination coefficients on the UL subproblem. Those approximated Pareto-



(a) 5-way 1-shot. (b) 5-way 5-shot.

Fig. 1. An illustration to show that MOML (the blue circle) can achieve the Pareto optimum. Experiments are conducted on the *mini*-ImageNet dataset under 5-way 1-shot and 5-way 5-shot FSL settings. The red line denotes the approximated Pareto frontier. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

optimal solutions form the approximated Pareto frontier (*i.e.*, the red line). As shown in Fig. 1, the solution of MOML (in the blue circle) falls very well on the approximated Pareto front, demonstrating the effectiveness of MOML.

6.2. Semi-supervised domain adaptation

Semi-Supervised Domain Adaptation (SSDA) aims to address the domain shift between two domains so that the model trained in a label-rich source domain can be adapted to a target domain with limited labeled samples and abundant unlabeled samples [81]. A widely used approach for SSDA is to align the distributions of two domains by finding some domain-invariant components. There are usually three objectives to be considered, including classification losses on both source and target domains and an alignment loss to measure the domain discrepancy. While existing works such as [59,89] optimize all the objectives by simply computing a weighted sum of them, we formulate the SSDA problem under the MOML framework.

6.2.1. Problem formulation

Given a source domain $\mathcal{S}$ and a target domain $\mathcal{T}$, the source domain has a large labeled dataset $\mathcal{D}_{\mathcal{S}}$, the target domain has a limited labeled dataset $\mathcal{D}_{\mathcal{T}}^{l}$ as well as a large unlabeled dataset $\mathcal{D}_{\mathcal{T}}^{u}$, and $\mathcal{D}_{\mathcal{T}} = \mathcal{D}_{\mathcal{T}}^{l} \bigcup \mathcal{D}_{\mathcal{T}}^{u}$ denotes the entire dataset for the target domain. The average classification losses in the source and target domains are represented by $\mathcal{L}_D(\omega, \mathcal{D}_{\mathcal{S}})$ and $\mathcal{L}_D(\omega, \mathcal{D}_{\mathcal{T}}^{l})$, respectively, where ω is the model parameter. The alignment loss denoted by $\mathcal{L}_A(\omega, \alpha, \mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}^{u})$ aims to learn domain-invariant components such as a domain-invariant projection space by minimizing the local maximum mean discrepancy in DSAN [89] or domain-invariant prototypes by maximizing the entropy in MME [59], where α is the initialization of ω. Then we can formulate the SSDA problem under the MOML framework as

$$\min_{\alpha} \left(\mathcal{L}_D(\omega^*(\alpha), \mathcal{D}_{\mathcal{S}}), \mathcal{L}_D(\omega^*(\alpha), \mathcal{D}_{\mathcal{T}}^{l}), \mathcal{L}_A(\omega, \alpha, \mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}^{u})\right) \quad \text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} \mathcal{L}_A(\omega, \alpha, \mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}^{u}). \tag{15}$$

In the LL subproblem of problem (15), we aim to learn a model to find a domain-invariant component between two domains via the alignment loss $\mathcal{L}_A$ by optimizing ω with an initialization α, and in the UL subproblem, we expect to improve the model further by updating α via minimizing the two classification losses together with the alignment loss. Here α acts similarly to the parameter initialization in MAML (*i.e.*, α in problem (12)) and helps learn ω in the LL subproblem, but it does not require any adaptation on the testing process.

Although a bi-level objective function is also formulated in Meta-MME [33], there exist two significant differences between Meta-MME and MOML (*i.e.*, problem (15)). Firstly, Meta-MME aims to learn the initialization of the network parameters by minimizing the source classification loss and the alignment loss in its LL subproblem and then validate it on few target labeled samples in the UL subproblem, which is different from MOML. Secondly, the proposed MOML method considers a multi-objective optimization problem in the UL subproblem, which is different from Meta-MME. Empirically, MOML outperforms the Meta-MME method as shown in Table 2.

We adapt problem (15) to MME [59] and DSAN [89], two domain adaptation models. We now give details about how to adapt the MOML framework to them. Let $\omega = \{\theta, \psi\}$, where θ denotes parameters in the backbone network and ψ represents parameters in the classification layer. Correspondingly, let $\alpha = \{\alpha_\theta, \alpha_\psi\}$, where α_θ and α_ψ represent the initialization of θ and ψ, respectively. Let $\hat{\mathbf{x}} = f(\theta, \alpha_\theta, x)$ denote the feature representation of an input x extracted by the backbone network parameterized by θ, which has an initialization α_θ. C denotes the number of classes. $n_{\mathcal{S}}$ and $n_{\mathcal{T}}^{u}$ denote the size of $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}^{u}$, respectively. The difference in applying MOML to different domain adaptation methods is the design of the alignment loss $\mathcal{L}_A(\omega, \alpha, \mathcal{D}_{\mathcal{S}}, \mathcal{D}_{\mathcal{T}}^{u})$.

Table 2
Classification accuracy (%) on the Office-31 dataset. † means the corresponding method is appropriately modified to adapt to our experimental setting according to the released code. ‡ indicates that the corresponding model is reimplemented by us. ↑, ↓, and − in the subscript indicate an increase, a decrease, and no change, respectively, when compared with the original method in two groups of models based on DSAN or MME. The best results in each group of models (*i.e.*, DSAN and MME) are highlighted in **bold** and the best results of each transfer task are annotated with underlines.

Method	A→D	D→A	A→W	W→A	D→W	W→D	Avg
S+T	93.58	74.16	92.17	74.08	98.01	100	88.67
DANN† [17]	93.09	74.52	91.60	75.07	98.58	100	88.81
ENT† [19]	93.33	74.16	94.44	75.03	97.86	100	89.13
APE† [28]	94.81	76.10	91.80	76.02	97.29	99.75	89.29
ADR† [60]	93.33	76.73	93.30	76.51	97.86	100	89.62
CDAN† [42]	94.32	75.25	92.88	76.98	98.43	100	89.64
DSAN† [89]	93.83	76.82	93.59	75.68	**98.43**	**100**	89.73
DSAN+SOML	**94.32**$_{\uparrow}$	76.91$_{\uparrow}$	94.16$_{\uparrow}$	**75.99**$_{\uparrow}$	97.72$_{\downarrow}$	**100**$_{-}$	89.85$_{\uparrow}$
DSAN+MOML	94.08$_{\uparrow}$	**77.13**$_{\uparrow}$	**94.59**$_{\uparrow}$	75.96$_{\uparrow}$	98.36$_{\downarrow}$	**100**$_{-}$	**90.02**$_{\uparrow}$
MME† [59]	92.09	77.71	93.73	77.27	98.01	**100**	89.80
Meta-MME‡ [33]	92.10	77.01	94.30	76.87	98.29	**100**	89.76
MME+SOML	92.09$_{-}$	77.57$_{\downarrow}$	94.30$_{\uparrow}$	77.20$_{\downarrow}$	98.29$_{\uparrow}$	**100**$_{-}$	89.90$_{\uparrow}$
MME+MOML	**94.32**$_{\uparrow}$	**78.30**$_{\uparrow}$	**94.44**$_{\uparrow}$	**77.71**$_{\uparrow}$	**98.43**$_{\uparrow}$	**100**$_{-}$	**90.53**$_{\uparrow}$

When adapting to MME [59], the alignment loss $\mathcal{L}_A$ is defined as the entropy loss to find domain-invariant prototypes:

$$\mathcal{L}_A(\omega, \alpha, \mathcal{D}_S, \mathcal{D}_{\mathcal{T}}^u) = \sum_{i=1}^{n_{\mathcal{T}}^u} \sum_{c=1}^{C} p(y=c \mid \hat{\mathbf{x}}_i^{\mathcal{T}}) \log p(y=c \mid \hat{\mathbf{x}}_i^{\mathcal{T}}), \tag{16}$$

where $\sigma(\cdot)$ denotes the softmax function and $p(y=c \mid \hat{\mathbf{x}}) = \left[\sigma\left(\frac{\psi^T \hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|}\right)\right]_c$ computes the conditional probability that x belongs to class c. ψ can be considered as the prototypes based on the cosine distance. Although Eq. (16) is not explicitly dependent on source data $\mathcal{D}_S$, the prototypes ψ are computed on $\mathcal{D}_S$ in previous iterations. Therefore, minimizing Eq. (16) (*i.e.*, maximizing the entropy) can promote the prediction distribution of unlabeled samples from the target domain to a uniform distribution, *i.e.*, all target features are close to the prototypes, which indicates learning the domain-invariant prototypes.

When adapting to DSAN [89], the alignment loss is defined as the local maximum mean discrepancy to measure the difference between two domains. Thus, following DSAN, $\mathcal{L}_A$ is formulated as

$$\mathcal{L}_A(\theta, \alpha_\theta, \mathcal{D}_S, \mathcal{D}_{\mathcal{T}}^u) = \frac{1}{C} \sum_{c=1}^{C} \left[\sum_{i,j=1}^{n_S} w_{i,c}^S w_{j,c}^S k(\hat{\mathbf{x}}_i^S, \hat{\mathbf{x}}_j^S) + \sum_{i,j=1}^{n_{\mathcal{T}}^u} w_{i,c}^{\mathcal{T}} w_{j,c}^{\mathcal{T}} k(\hat{\mathbf{x}}_i^{\mathcal{T}}, \hat{\mathbf{x}}_j^{\mathcal{T}}) - 2 \sum_{i=1}^{n_S} \sum_{j=1}^{n_{\mathcal{T}}^u} w_{i,c}^S w_{j,c}^{\mathcal{T}} k(\hat{\mathbf{x}}_i^S, \hat{\mathbf{x}}_j^{\mathcal{T}}) \right],$$

where $k(\cdot,\cdot)$ denotes a kernel function. Here $w_{i,c}^S = \frac{y_{i,c}}{\sum_{j=1}^{n_S} y_{j,c}}$ represents the possibility of sample x_i^S belonging to class c, where $y_{i,c}$ is the c-th element of the one-hot label vector of x_i^S. The definition of weight $w_{i,c}^{\mathcal{T}}$ is similar, while we use the prediction distribution as the pseudo label of $x_i^{\mathcal{T}}$ from $\mathcal{D}_{\mathcal{T}}^u$ since its true label is unavailable.

6.2.2. Experimental settings

Experiments are conducted on the Office-31 dataset [58], which has 3 domains: Amazon (**A**), Webcam (**W**) and DSLR (**D**). It contains 4,110 labeled images in total and each domain consists of 31 categories. By following [71,43], we construct all six transfer tasks. Each class in the target domain has three labeled images in the training process by following [59]. Baseline models in comparison include a deep neural network (denoted by '**S+T**') that is trained on $\mathcal{D}_S \bigcup \mathcal{D}_{\mathcal{T}}^l$ and eight domain adaptation methods: **ENT** [19], **DANN** [17], **ADR** [60], **CDAN** [42], **MME** [59], **Meta-MME** [33], **APE** [28], and **DSAN** [89]. As DSAN and MME are two representative domain adaptation models, to improve their performance further, SOML and MOML adopt the alignment loss in DSAN and MME, respectively, leading to two groups of models, *i.e.*, (DSAN, DSAN+SOML, DSAN+MOML) and (MME, Meta-MME, MME+SOML, MME+MOML). All the methods except S+T are trained on $\mathcal{D}_S \bigcup \mathcal{D}_{\mathcal{T}}$.

We use the ResNet-50 model [21] pretrained on the ImageNet dataset as the backbone network followed by a Fully-Connected (FC) layer. The same network architecture is used for all baseline methods. All baselines use the same experimental setting as the original methods. For the training of all the DSAN-related models, the SGD optimizer with the learning rate 10^{-3}, the momentum 0.9 and the weight decay 5×10^{-4} is used for optimization. Following [33], the number of LL iterations K of SOML and MOML is set to 1. The batch size is set to 96, including 32 images in the source, labeled target, and unlabeled target domains, respectively. For all the MME-related models, we use the same experimental settings as the original MME method.

6.2.3. Experimental results

Experimental results on the Office-31 dataset are shown in Table 2. The incorporation of SOML into DSAN and MME consistently improves the performance of those two models and the proposed MOML method further improves the classification accuracy. For example, MME+MOML significantly improves the performance of the first five tasks compared with several baselines and achieves the perfect performance (*i.e.*, the 100% accuracy) on the last task as most baselines did. Moreover, compared with several baselines, DSAN+MOML achieves the best result (*i.e.*, 94.59%) on transfer task A→W and MME+MOML has the best accuracy (*i.e.*, 78.30% and 77.71%) on transfer tasks D→A and W→A. Moreover, MME+MOML achieves the best average classification accuracy of 90.53% and significantly outperforms all of the baselines. These experimental results indicate the effectiveness of the proposed MOML framework for SSDA.

6.3. Multi-task learning

Multi-Task Learning (MTL) [3,87,35] aims to improve the performance of multiple tasks simultaneously by leveraging useful information contained in these tasks. Learning the loss weighting is a challenge in MTL and there are some works [61,41,85] to solve this problem. Among those works, Sener and Koltun [61] formulate multi-task learning problems from the perspective of multi-objective optimization and implicitly learn the task weights via MGDA, Liu et al. [41] estimate the task weight of each task as the ratio of the training losses in the last two iterations for the corresponding task, and Yu et al. [85] project each task's gradient onto the normal plane of the other. Different from those works which are all based on single-level optimization problems on the entire training set, for the first time we formulate this problem as an MOBLP based on the split of the entire training dataset and solve this problem based on the MOML framework.

6.3.1. Problem formulation

Suppose there are m tasks. The i-th task has a dataset $\mathcal{D}_i$ for model training. Here each $\mathcal{D}_i$ is partitioned into two subsets: the training dataset $\mathcal{D}_i^{tr}$ and the validation dataset $\mathcal{D}_i^{val}$, where $\mathcal{D}_i^{tr}$ is used to train a multi-task model and $\mathcal{D}_i^{val}$ is to measure the performance of the multi-task model on the i-th task. $f(\cdot;\omega)$, the learning function of the multi-task model parameterized by ω, receives data points from the m tasks and outputs predictions. Let $\alpha_i \in [0,1]$ denote the loss weight for the i-th task. The goal is to learn the simplex weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)^\top \in \Delta^{m-1}$ and the model parameter ω. The objective function of the proposed method under the MOML framework is formulated as

$$\min_{\boldsymbol{\alpha}\in\Delta^{m-1}} \left(\mathcal{L}_{MTL}(\omega^*(\boldsymbol{\alpha}), \mathcal{D}_1^{val}), \ldots, \mathcal{L}_{MTL}(\omega^*(\boldsymbol{\alpha}), \mathcal{D}_m^{val})\right) \quad \text{s.t.} \quad \omega^*(\boldsymbol{\alpha}) = \arg\min_{\omega} \sum_{i=1}^{m} \alpha_i \mathcal{L}_{MTL}(\omega, \mathcal{D}_i^{tr}), \tag{17}$$

where $\mathcal{L}_{MTL}(\omega, \mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f(\mathbf{x};\omega), y)$ denotes the average loss of $f(\cdot;\omega)$ on a dataset $\mathcal{D}$ with $|\mathcal{D}|$ denoting the size of $\mathcal{D}$ and $\ell(\cdot,\cdot)$ denoting a loss function. In the LL subproblem of problem (17), when given weights in $\boldsymbol{\alpha}$, we aim to learn a MTL model to get optimal parameters ω^* on the training dataset and in the UL subproblem, we expect to update $\boldsymbol{\alpha}$ via minimizing the loss of the trained MTL model with parameters ω^* on the validation dataset of each task.

6.3.2. Experiments settings

Experiments in MTL are conducted on the NYUv2 [63], Office-31, and Office-Home [72] datasets. The NYUv2 dataset is an indoor scene RGB-D image dataset, which consists of three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. We use the NYUv2 dataset pre-processed by [41]. It contains 795 and 654 labeled images for training and test, respectively. The Office-31 dataset has been introduced in Section 6.2.2 and we consider the classification problem on each domain as one task, leading to three tasks. The Office-Home dataset contains four tasks: artistic images **(Ar)**, clip art **(Cl)**, product images **(Pr)**, and real-world images **(Rw)**. It has 15,500 labeled images in total and each task consists of 65 object categories in office and home settings.

Baseline methods in the comparison include different loss weighting strategies such as Equal Weights (**EW**), Dynamic Weight Average (**DWA**) [41] with the temperature parameter T as 2, **MGDA** [61], **PCGrad** [85], and **SOML**, and they are built on two multi-task architectures, including Deep Multi-Task Learning (**DMTL**) that adopts the hard-sharing structure to share the first several layers and Multi-Task Attention Network (**MTAN**) [41]. All methods are implemented based on the open-source `LibMTL` library [36].

NYUv2 The ResNet-50 pretrained on the ImageNet dataset is used as the backbone to extract features and we do not use data augmentation. Atrous Spatial Pyramid Pooling (ASPP) [4] modules are used as the decoder for each task-specific output. For the MTAN architecture, we add m task-specific attention networks into the backbone based on the DMTL architecture. The MTL model with parameter ω in problem (17) is trained by the Adam optimizer with the learning rate as 10^{-4} and weight decay as 10^{-5}. The loss weight $\boldsymbol{\alpha}$ in problem (17) is initialized with equal values and is optimized by the Adam optimizer with the learning rate as 10^{-4}. For SOML and the proposed MOML method, we randomly split 795 training images into two parts: 636 for training and the rest 159 for validation and test on the same test dataset as all baselines. The number of LL iterations K is set to 1 for the SOML and MOML methods. A batch size 8 is used for the DMTL architecture and 4 for the MTAN architecture.

Office-31 and office-home The ResNet-18 pretrained on the ImageNet dataset is used as the backbone to extract features and m linear layers are used for task-specific classification. We also add m task-specific attention networks to build the MTAN architecture. The

Table 3
Performance on the NYUv2 dataset with three tasks: 13-class semantic segmentation, depth estimation, and surface normal prediction. The best combinations of the architecture and weighting strategy are highlighted in **bold**. The best results for each task on each measure are annotated with underlines. ↑ (↓) means the higher (lower) the result, the better the performance.

Architecture	Weighting Strategy	Segmentation		Depth		Surface Normal Prediction				
		mIoU↑	Pix Acc↑	Abs Err↓	Rel Err↓	Angle Distance		Within t°		
						Mean↓	Median↓	11.25↑	22.5↑	30↑
DMTL	EW	52.71	74.78	0.3886	**0.1581**	23.8568	17.3537	34.57	60.45	71.63
	DWA [41]	52.72	75.11	0.3931	0.1631	23.7894	17.3320	34.57	60.51	71.67
	MGDA [61]	52.89	74.87	0.3963	0.1638	23.7513	17.2685	34.74	60.64	71.75
	PCGrad [85]	53.22	75.45	0.3920	0.1658	23.2904	16.8728	35.47	61.58	72.56
	SOML	53.20	75.22	0.3923	0.1628	23.6885	17.0228	35.42	61.03	71.97
	MOML	**54.98**	**75.98**	**0.3877**	0.1618	**23.2401**	**16.7388**	**35.90**	**61.81**	**72.76**
MTAN [41]	EW	53.97	75.90	**0.3794**	0.1580	22.8743	16.5502	36.54	62.52	73.31
	DWA [41]	54.12	75.79	0.3902	0.1595	22.9691	16.6212	36.23	62.41	73.28
	MGDA [61]	54.38	75.55	0.3854	0.1583	22.9396	16.4670	36.70	62.58	73.23
	PCGrad [85]	**54.40**	**76.13**	0.3830	0.1581	23.0040	16.4636	36.67	62.65	73.34
	SOML	54.03	75.48	0.3829	0.1581	22.8279	16.4259	36.74	62.67	73.46
	MOML	54.23	75.63	0.3843	**0.1567**	**22.7530**	**16.2468**	**37.20**	**63.09**	**73.65**

Table 4
Classification accuracy (%) on the Office-31 and Office-Home datasets. The best combinations of the architecture and weighting strategy are highlighted in **bold**. The best results for each task on each measure are annotated with underlines.

Architecture	Weighting Strategy	Office-31				Office-Home				
		A	D	W	Avg	Ar	Cl	Pr	Rw	Avg
DMTL	EW	87.17	98.36	**99.44**	94.99	68.88	80.93	**91.73**	81.72	80.81
	DWA [41]	87.52	99.18	**99.44**	95.38	70.39	79.95	90.36	**82.05**	80.69
	MGDA [61]	87.52	99.18	**99.44**	95.38	69.44	79.30	91.63	81.72	80.52
	PCGrad [85]	87.00	98.36	98.33	94.56	68.31	80.71	90.57	81.94	80.38
	SOML	87.35	**100**	98.89	95.41	**70.77**	81.14	90.46	80.97	80.84
	MOML	**87.69**	99.18	**99.44**	**95.43**	69.63	**81.79**	91.20	**82.05**	**81.17**
MTAN [41]	EW	87.52	98.36	**99.44**	95.10	69.63	80.60	91.94	82.91	81.27
	DWA [41]	87.52	**100**	**99.44**	95.65	69.63	81.14	91.10	82.48	81.09
	MGDA [61]	87.35	99.18	**99.44**	95.32	69.25	**81.36**	91.73	82.81	81.29
	PCGrad [85]	87.69	**100**	**99.44**	95.71	69.25	**81.36**	**92.37**	82.27	81.31
	SOML	87.52	**100**	**99.44**	95.65	70.77	81.25	91.20	82.59	81.45
	MOML	**88.20**	**100**	**99.44**	**95.88**	**70.96**	81.14	92.05	**83.02**	**81.80**

MTL model is trained by the Adam optimizer with the learning rate as 10^{-4}. The loss weight $\boldsymbol{\alpha}$ is initialized with equal values and optimized by the Adam optimizer with the learning rate as 10^{-3}. Both the Office-Home and Office-31 datasets are split into three parts, including 60% for training, 20% for validation, and the remaining 20% for testing. The number of LL iterations K is set to 1 for the SOML and MOML methods. All the baselines are trained on training and validation datasets. We set the batch size to 64 for both datasets.

6.3.3. *Experimental results*

Experimental results on the NYUv2 dataset are shown in Table 3. Firstly, SOML outperforms the EW, DWA, and MGDA strategies when using the DMTL architecture and achieves comparable performance with the four baselines with the MTAN architecture. It indicates the proposed bi-level formulation (*i.e.*, problem (17) with weighted combined objectives in the UL subproblem used in SOML) can achieve very good performance when compared with several baselines. Secondly, the MOML method outperforms the SOML method under both DMTL and MTAN architectures, which means the multi-objective formulation in the UL subproblem is better than the single-objective formulation. Finally, it is noticeable that the proposed MOML method achieves better results in many metrics. For example, when training with the DMTL architecture, MOML can achieve 54.98% in terms of the mIoU, which is significantly higher than several baselines even with the advanced MTAN architecture. MOML, with the MTAN architecture, achieves the best results on six metrics.

The loss weights learned by the proposed MOML are $(0.3120, 0.3804, 0.3076)$ and $(0.2541, 0.4860, 0.2599)$ for the three tasks (*i.e.*, semantic segmentation, depth estimation, and surface normal prediction) when using the DMTL and MTAN architectures, respectively. It is interesting to find that when using different architectures, the weight of the depth estimation task is commonly the highest, while the other two tasks have similar weights.

Experimental results on the Office-31 and Office-Home datasets are shown in Table 4. Firstly, we notice that SOML achieves comparable performance with several baselines under both DMTL and MTAN architectures on both datasets, which means the proposed bi-level formulation (*i.e.*, problem (17) with weighted combined objectives in the UL subproblem) is competitive when comparing

with the baselines with the single-objective formulation. Secondly, the proposed MOML slightly outperforms SOML in all cases. This indicates that the multi-objective formulation in the UL subproblem is better than the single-objective formulation in SOML. Finally, compared with several baselines, the proposed MOML achieves the best result in some tasks, such as the best classification accuracy 88.20% in task **A** on the Office-31 dataset. Those experimental results show that the proposed MOML framework performs better for learning loss weights in MTL.

6.4. Neural architecture search

Neural Architecture Search (NAS) aims to design the architecture of neural networks in an automated way. Most NAS methods focus on searching architectures with the best classification accuracy. However, in real-world applications, we need to consider multiple factors in the architecture design. For example, we expect that the searched architecture has good performance, behaves robustly to noises, and consumes low resource. In this case, we formulate NAS with multiple objectives as a case of MOML.

6.4.1. Problem formulation

By following the DARTS method [38], in an operation space denoted by $\mathcal{O}$, each element is an operation function $o(\cdot)$ and each cell is a directed acyclic graph with N nodes, where each node represents a hidden representation and each edge (i,j) denotes a candidate operation $o(\cdot)$ with a probability $\alpha_o^{(i,j)}$. Therefore, $\boldsymbol{\alpha} = \{\alpha_o^{(i,j)}\}_{(i,j)\in \boldsymbol{E}, o\in\mathcal{O}}$ is a representation of the neural architecture, where $\boldsymbol{E}$ denotes the set of all the edges in all the cells. The entire dataset is split into a training dataset denoted by $\mathcal{D}_{tr}$ and a validation dataset denoted by $\mathcal{D}_{val}$.

The multi-objective NAS considers three objectives: classification accuracy, adversarial robustness, and the number of parameters. We formulate three corresponding losses as $\mathcal{L}_N(\omega, \boldsymbol{\alpha}, \mathcal{D}_{val})$, $\mathcal{L}_N(\omega, \boldsymbol{\alpha}, \mathcal{D}_{val}^{adv})$, and $\mathcal{L}_{nop}(\boldsymbol{\alpha})$, where $\mathcal{D}_{val}$ denotes the validation dataset and $\mathcal{D}_{val}^{adv}$ denotes the perturbed validation dataset by adding perturbations on each data point, and the objective function under the MOML framework is formulated as

$$\min_{\boldsymbol{\alpha}} \ \big(\mathcal{L}_N(\omega^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}, \mathcal{D}_{val}), \mathcal{L}_N(\omega^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}, \mathcal{D}_{val}^{adv}), \mathcal{L}_{nop}(\boldsymbol{\alpha})\big) \quad \text{s.t.} \quad \omega^*(\boldsymbol{\alpha}) = \arg\min_{\omega} \ \mathcal{L}_N(\omega, \boldsymbol{\alpha}, \mathcal{D}_{tr}), \tag{18}$$

where ω denotes all the model parameters in the neural network.

In problem (18), loss function $\mathcal{L}_N(\omega, \boldsymbol{\alpha}, \mathcal{D})$ denotes the average classification loss on a dataset $\mathcal{D}$ of a neural network with parameters ω and an architecture $\boldsymbol{\alpha}$. To formulate $\mathcal{L}_{nop}(\boldsymbol{\alpha})$, we denote by n_o the number of parameters associated with an operation o and by $N_{nop}(\boldsymbol{\alpha})$ the number of parameters in a searched architecture $\boldsymbol{\alpha}$. Then $N_{nop}(\boldsymbol{\alpha})$ can be computed by $N_{nop}(\boldsymbol{\alpha}) = \sum_{(i,j)\in\boldsymbol{E}} n^{(i,j)}$, where $n^{(i,j)}$ is the number of parameters of the searched operation on the edge (i,j). As we determine the operation of each edge by selecting the one with the largest probability, hence we have $n^{(i,j)} = n_{\arg\max_{o\in\mathcal{O}} \alpha_o^{(i,j)}}$. As the $\arg\max$ operation is non-differentiable, we use the softmax function to approximate $N_{nop}(\boldsymbol{\alpha})$ as

$$\hat{N}_{nop}(\boldsymbol{\alpha}) = \sum_{(i,j)\in\boldsymbol{E}} \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} n_o. \tag{19}$$

Therefore, to search a network architecture with an expected size L, $\mathcal{L}_{nop}(\boldsymbol{\alpha})$ can be formulated as

$$\mathcal{L}_{nop}(\boldsymbol{\alpha}) = |\hat{N}_{nop}(\boldsymbol{\alpha}) - L|. \tag{20}$$

In the LL subproblem of problem (18), when given the architecture $\boldsymbol{\alpha}$, we can train a model with optimal parameters ω on the training dataset and in the UL subproblem, we expect to update the architecture $\boldsymbol{\alpha}$ by making a trade-off among the validation loss, the adversarial robustness, and the number of parameters. It is easy to see that the DARTS method is a special case of problem (18) when its UL subproblem contains the first objective only and hence problem (18) generalizes the DARTS method by considering two more objectives.

Compared with the NSGANetV2 method [44] that utilizes a multi-objective bi-level evolutionary algorithm, MOML is more efficient and has a convergence guarantee. Moreover, NSGANetV2 uses ensembled surrogate models to predict the accuracy of an architecture, which may incur a performance gap between the UL and LL subproblems. In the LL subproblem of NSGANetV2, it only chooses over 300 candidate architectures for evaluation with a supernet constructed for weight sharing, which may lead to suboptimal solutions.

6.4.2. Experimental settings

Experiments in NAS are conducted on the CIFAR-10 dataset [31]. We compare the proposed MOML with the DARTS method and three baselines, including SNAS [79], RC-DARTS [27], and ENAS [53]. We search for neural networks with different expected sizes (*i.e.*, different L's used in Eq. (20)) via the MOML method. To make the network size searched by DARTS comparable with that of MOML under different settings, we use different numbers of initial channels in DARTS during the evaluation process. Hence, according to the parameter size (*i.e.*, $L = 1$, 2, and 3), we split the results of DARTS, SOML and MOML into three groups.

As claimed in Section 4.2, we can use any gradient-based optimization algorithm to update the UL variable. To verify this, we here use the CAGrad method [37] as an alternating MOPSlover to update the UL variable in Algorithm 1. CAGrad explicitly controls the minimum decreasing rate to optimize the average loss of all tasks. The setting of the CAGrad method follows [37]. This approach can be considered as an extended version of MOML and we call it as MOML+CAGrad.

Table 5
Comparison between MOML and DARTS and the other three baselines on the CIFAR-10 dataset. ↑ indicates that a larger value is better, while ↓ implies that a lower value is better. "{DARTS-C#channels}" means that the architecture searched by DARTS is evaluated with the initial number of channels as "channels". "{SOML-V#size-C#channels}" means that the architecture searched by SOML with L as "size" is evaluated by the initial number of channels as "channels". "{MOML-V#size}" denotes the architecture searched by MOML with L as "size". "{MOML + CAGrad-V#size}" means using CAGrad method as the MOPSlover with L as "size". The B-score, which is defined in Section 6.1.2, measures the balance between the clean accuracy and PGD accuracy when the numbers of parameters in different architectures are comparable.

Architecture	Params (MB) ↓	Clean Acc. (%) ↑	PGD Acc. (%) ↑	B-score
SNAS (moderate) [79]	2.8	97.15	-	-
RC-DARTS-C42 [27]	3.3	97.19	-	-
ENAS [53]	4.6	97.11	-	-
DARTS-C26 [38]	1.787	96.91	28.45	43.98
SOML-V1-C38	1.750	96.36	40.20	56.73
MOML-V1	1.754	96.48	42.66	**59.16**
MOML+CAGrad-V1	1.751	96.40	41.00	57.53
DARTS-C30 [38]	2.354	97.13	31.53	47.60
SOML-V2-C42	2.402	97.03	31.44	47.49
MOML-V2	2.367	97.18	36.15	**52.69**
MOML+CAGrad-V2	2.395	97.16	35.68	52.19
DARTS-C34 [38]	2.998	97.34	30.31	46.22
SOML-V3-C36	3.018	97.18	35.36	51.85
MOML-V3	3.018	97.25	35.22	51.71
MOML+CAGrad-V3	3.015	97.20	35.40	**51.89**

The search space and training procedure of MOML adopt the same settings as DARTS [38]. Specifically, in both normal and reduction cells, the set of operations $\mathcal{O}$ contains eight operations, including 3×3 separable convolutions, 5×5 separable convolutions, 3×3 dilated separable convolutions, 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. Half of the training set is used to train a model, and the other half is used for validation. A small network of 8 cells is trained with a batch size of 64 and 16 initial channels for 50 epochs. Following [38], the number of LL iterations K is set to 1 for all the methods in comparison. The Adam optimizer with the learning rate 3×10^{-4}, the momentum $\beta=(0.5,0.999)$, and the weight decay 1×10^{-3} are used to update $\boldsymbol{\alpha}$ in the UL subproblem. The SGD optimizer with the decayed learning rate down from 0.025 to 0 by a cosine schedule, the momentum 0.9, and the weight decay 3×10^{-4} is used to update ω in the LL subproblem.

In the evaluation stage, a neural network of 20 searched cells is trained on the full training set for 600 epochs with the batch size as 96, the initial number of channels as 36, the length of a cutout as 16, the dropout probability as 0.2, and auxiliary towers of weight as 0.4. The full testing set is used for testing. Adversarial examples are generated using the PGD attack with the perturbation size $\epsilon=1/255$ and the PGD attack takes 10 iterative steps with the step size of 2.5ϵ as suggested in [32].

6.4.3. Experimental results

Experimental results on the CIFAR-10 dataset are shown in Table 5. Compared with three baselines (*i.e.*, SNAS, RC-DARTS, and ENAS), the proposed MOML and MOML+CAGrad can search architecture with similar or less parameter size but higher clean accuracy than all baselines. Compared with DARTS, MOML and MOML+CAGrad achieve a better trade-off in terms of accuracy, network size, and robustness. Specifically, with a comparable number of parameters, the MOML method improves the robustness while testing accuracies are comparable or even slightly better. For example, compared MOML-V1 with DARTS-C26, the PGD accuracy increases by about 14%, while the clean test accuracy only drops around 0.5%. In addition, compared with SOML, MOML has a higher B-score when L equals 1 and 2 and has a comparable B-score when L equals 3. It indicates that MOML with the multi-objective formulation in the UL subproblem can achieve a better trade-off between clean accuracy and PCG accuracy than SOML with the single-objective formulation. Moreover, MOML+CAGrad has a higher B-score than SOML in all the settings, which further illustrates the generality of the proposed MOML framework that different optimization algorithms can be used to solve the UL subproblem. According to Table 5, experimental results show that the MOML and MOML+CAGrad methods can search more robust architectures with comparable model size and comparable classification accuracy when compared with all the baseline models.

6.5. Reinforcement learning

Reinforcement learning aims to deal with the problem of how intelligent agents ought to take action in an environment to maximize the notion of cumulative reward. We consider the Multi-Task Reinforcement Learning (MTRL) problem, which aims to improve the performance of agents in multiple tasks and is a promising approach to train effective real-world agents [86]. In the following, we formulate the MTRL problem under the MOML framework.

6.5.1. Problem formulation

Suppose there are m reinforcement learning tasks, each of which can be defined as policy search in the Markov decision process (MDP) [86]. Each task $\mathcal{T}$ corresponds to an MDP, represented by a tuple (S, A, P, R, H, γ), where S defines the state space, A defines the action space, $P(s_{t+1}|s_t, a_t)$ represents the state transition probability that the next state s_{t+1} occurs given the current state s_t and action a_t, $R(s, a)$ is a reward function, H is the horizon, and γ is the discount factor. The goal of MTRL is to learn a policy $\pi(a|s, z)$ that maximizes the expected return in all the tasks formulated as $\mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathbb{E}_{\pi}[\sum_{t=1}^{T}\gamma^t R_t(s_t, a_t)]]$, where z represents an encoding of a task.

We now introduce the Actor-Critic (AC) method [30] as an example to show how to view a reinforcement learning problem as a bi-level optimization problem. The AC method is a representative reinforcement learning technique that simultaneously learns a policy function and a value function. Consider a parameterized state-action value function $Q(s_t, a_t)$ and a tractable policy π. The AC method first uses the actor π to interact with the environment and to learn the value function Q by minimizing the temporal difference (TD) error. Then it uses the given value function to update the policy network by maximizing the expected discounted cumulative reward. Therefore, the value function can be considered to be optimized with respect to the optimum of the policy function. Thus, the AC method can be viewed as a bi-level optimization problem where the actor and critic correspond to the LL and UL variables, respectively [39].

In MTRL, suppose each task shares the same model, and we consider a parameterized state-action function $Q_\alpha(s_t, a_t)$ and a tractable policy $\pi_\omega(a_t|s_t, z)$, where α denotes parameters of the state-action value-function and ω denotes parameters of the policy network. Then, according to the above discussion, we can formulate the MTRL problem under the MOML framework as

$$\min_{\boldsymbol{\alpha}} \ \big(\mathcal{L}_Q(\omega^*(\boldsymbol{\alpha}), \alpha; z_1), \ldots, \mathcal{L}_Q(\omega^*(\boldsymbol{\alpha}), \alpha; z_m)\big) \quad \text{s.t.} \quad \omega^*(\boldsymbol{\alpha}) = \arg\min_{\omega} \sum_{i=1}^{m} J_\pi(\omega, \alpha; z_i), \tag{21}$$

where $\mathcal{L}_Q(\omega, \alpha; z_i)$ and $J_\pi(\omega, \alpha; z_i)$ represent the loss function for the state-action function Q and the policy network of the i-th task, respectively, and z_i represents the encoding vector of the i-th task.

We adapt problem (21) into the Soft Actor-Critic (SAC) method [20], which is an off-policy actor-critic deep RL algorithm and has been widely used in most MTRL models [67,86]. For the i-th task, we use $D_{\pi_i}(s_t, a_t)$ and $D_{\pi_i}(s_t)$ to denote the state-action marginals and state of the trajectory distribution induced by a policy $\pi(a_t|s_t, z_i)$. The SAC method for the i-th task is to find the policy to maximize $\mathbb{E}_{(s_t,a_t)\sim D_{\pi_i}}\sum_t[R(s_t, a_t) + \beta_i\mathcal{H}(\pi(\cdot|s_t, z_i))]$, where β_i represents the temperature for task i and $\mathcal{H}(\cdot)$ denotes the entropy function. In SAC method, the soft Q-function parameters can be trained to minimize the soft Bellman residual. Therefore, the objective function of the soft Q-function for the i-th task is formulated as

$$\mathcal{L}_Q(\omega, \alpha; z_i) = \mathbb{E}_{(s_t,a_t)\sim D_{\pi_i}}\left[\frac{1}{2}\left(Q_\alpha(s_t, a_t) - \left(R(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim P}[V_{\hat{\alpha}}(s_{t+1}, z_i)]\right)\right)^2\right], \tag{22}$$

where $V(s_t, z_i) = \mathbb{E}_{a_t\sim\pi(a_t|s_t,z_i)}[Q(s_t, a_t) - \beta_i \log\pi(a_t|s_t, z_i)]$ and parameters in $\hat{\alpha}$ are an exponentially moving average of the soft Q-function weights, which is to stabilize training [49]. The policy parameters α can be learned by directly minimizing the expected KL-divergence, and the corresponding objective for the i-th task is formulated as

$$J_\pi(\omega, \alpha; z_i) = \mathbb{E}_{s_t\sim D_{\pi_i}}\left[D_{\mathrm{KL}}\left(\pi_\omega(\cdot \mid s_t, z_i)\| \frac{\exp(Q_\alpha(s_t \mid \cdot))}{Z_\alpha(s_t)}\right)\right], \tag{23}$$

where $Z_\alpha(s_t)$ denotes the partition function to do the normalization. By following Haarnoja et al. [20], we minimize the objective $\mathcal{L}_Q$ w.r.t. α and J_π w.r.t. ω in problem (21) separately.

6.5.2. Experimental settings

Experiments in MTRL are conducted on the MT10 and MT50 benchmarks [86] from the Meta-World environment. MT10 and MT50 contain 10 and 50 robot manipulation tasks. The MT10 evaluation uses 10 tasks: reach, push, pick and place, open door, open drawer, close drawer, press button top-down, insert peg side, open window, and open box. The larger MT50 evaluation uses all 50 Meta-World tasks. Baseline models in comparison include single-task SAC [86], multi-task SAC [86], multi-headed SAC [86], PCGrad [85], soft modularization [82], and CAGrad [37]. Among those methods, multi-task SAC, PCGrad, and CAGrad methods use one shared model, multi-headed SAC uses a shared backbone and task-specific heads, soft modularization estimates per-task routing for different tasks in a shared model, single-task SAC uses one SAC model for each task. For the SOML and the MOML methods, we also use one shared model, which is the same as the multi-task SAC, PCGrad, and CAGrad methods.

Similar to experimental settings in [67], the policy network is provided with a one-hot vector as the task representation. The batch size of all the baseline methods is set to $128m$, where m denotes the number of tasks. To ensure that the number of data used in each iteration is consistent with baseline methods for a fair comparison, we split the sampled replay into two parts in the SOML and MOML methods, including $64m$ for the upper-level subproblem and $64m$ for the lower-level subproblem. The number of LL iterations K is set to 1 for the SOML and MOML methods. Other settings follow [67]. We train all methods over 2 million steps and evaluate each agent at every $10,000$ environment steps. We report the mean success rate over 10 random seeds.

6.5.3. Experimental results

Experimental results on the MT10 and MT50 benchmarks are shown in Table 6. In MTRL, most baselines do not achieve performance comparable to the single-task SAC method, and the performance gap increases as the number of tasks increases. For example, the success rate of the CAGrad method is 13% lower in MT10 and 22% lower in MT50 when compared with single-task SAC. This could

Table 6
Results on the MT10 and MT50 benchmarks. Results are obtained over 10 independent runs and the best result except single-task SAC is shown in bold.

Method	Metaworld MT10 Success±SEM	Metaworld MT50 Success±SEM
Single-task SAC [86]	0.90±0.032	0.74±0.041
Multi-task SAC [86]	0.49±0.073	0.36±0.013
Multi-headed SAC [86]	0.61±0.036	0.45±0.064
PCGrad [85]	0.72±0.022	0.50±0.017
Soft Modularization [82]	0.73±0.043	0.50±0.035
CAGrad [37]	0.83±0.045	0.52±0.023
SAC+SOML	0.88±0.075	0.51±0.034
SAC+MOML	**0.90**±0.054	**0.53**±0.036

Table 7
Running time (second/epoch) of solving the LL and UL subproblems on the *mini*-ImageNet dataset under 5-way 5-shot FSL setting. K is the number of LL iterations.

Method	$K = 5$		$K = 10$	
	LL subproblem	UL subproblem	LL subproblem	UL subproblem
MAML [13]	5.22±0.31	50.50±0.05	10.15±0.67	101.78±0.13
MAML+SOML	5.23±0.27	51.70±0.07	10.52±0.57	102.96±0.15
MAML+MOML	5.16±0.28	105.63±0.09	10.48±0.58	209.61±0.15

be due to the fact that not all the tasks are highly related, making it difficult to learn a shared model. The SOML method performs comparable and even better than all the baselines except single-task SAC in both MT10 and MT50 benchmarks. This indicates that the proposed bi-level formulation in MTRL can achieve good performance. Moreover, the MOML method outperforms the SOML method and firstly achieves comparable performance (*i.e.*, the 90% success rate) with single-task SAC in the MT10 benchmark. Those results indicate that the multi-objective formulation in the UL subproblem is better than the single-objective one, and the proposed MOML framework achieves state-of-the-art performance for the MTRL problem.

7. Analysis of training efficiency

In this section, we provide the time complexity analysis and empirical evaluation in terms of the running time for the proposed MOML method (*i.e.*, Algorithm 1), where gradient descent is used to update ω and MGDA [8] is used as the solver to update α.

In every training iteration, it takes $\mathcal{O}(pK)$ time to execute the K-iteration update for $\omega \in \mathbb{R}^p$ to solve the LL subproblem (*i.e.*, step 5 in Algorithm 1). Then calculating the hyper-gradient via Eq. (6) in the UL subproblem needs $\mathcal{O}(p(n+p)K)$ time, where n denotes the dimension of the UL objective α. Since the UL subproblem contains m objectives, we need to calculate m hyper-gradients (*i.e.*, step 7 in Algorithm 1), which is typical in gradient-based MOP methods [37,61,85]. The cost of solving the quadratic programming problem in MGDA (*i.e.*, step 8 in Algorithm 1) is negligible [37], since m is typically very small in real-world applications. Thus, for each training iteration, MOML costs $\mathcal{O}(mp(n+p)K)$ in total.

Empirically, Table 7 shows the per-epoch running time of the proposed MOML method and baselines on the *mini*-ImageNet dataset with different numbers of LL iterations (*i.e.*, $K = 5$ and 10) under the 5-way 5-shot FSL setting. The experimental settings are the same as those in Section 6.1. All methods are run for 100 epochs on a single NVIDIA GeForce RTX 3090 GPU, and the average running time and its standard deviation per epoch are reported. As can be seen, all the methods have similar running times in solving the LL subproblem and the time spent on the UL subproblem is more than that on the LL subproblem for all the methods since we need to calculate the hyper-gradient in the UL subproblem. MOML spends twice the time to solve the UL subproblem because it needs to compute two hyper-gradients of the two objectives in the UL subproblem. When the number of the LL iterations doubles (*i.e.*, $K = 10$), the running time for all methods to solve the UL and LL subproblems also doubles. Those empirical results are consistent with the analysis on the time complexity.

8. Conclusions

As a generalization of meta-learning based on the bi-level formulation, the MOML framework based on multi-objective bi-level optimization is proposed in this paper. In the MOML framework, the upper-level subproblem takes multiple objectives of a learning problem into consideration. To solve the objective function of the MOML framework, a gradient-based optimization algorithm is proposed, and its convergence properties are studied. Moreover, several use cases of the MOML framework are investigated to demonstrate the effectiveness of the MOML framework in different learning problems. In our future work, we are interested in applying MOML to other learning problems.

CRediT authorship contribution statement

Feiyang Ye: Investigation, Methodology, Writing – original draft. **Baijiong Lin:** Methodology, Writing – original draft. **Zhixiong Yue:** Methodology. **Yu Zhang:** Conceptualization, Investigation, Methodology, Supervision, Writing – review & editing. **Ivor W. Tsang:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

In the submission, we have shared the link or the reference to the code and data used.

Acknowledgements

This work is supported by National Key R&D Program of China (No. 2022ZD0160300), NSFC Key Grant 62136005, NSFC General Grant 62076118, and Shenzhen Fundamental Research Program JCYJ20210324105000003.

Appendix A. Review of multi-objective optimization

A.1. Notations and terminologies

We first recall some definitions in multi-objective optimization, including the definition of the minimality and convexity of vector-valued functions and Kuratowski-Painlevé set-convergence [45].

Let P be the set of non-negative real vectors $\mathbb{R}^m_+ = \{l \in \mathbb{R}^m : \forall l_i \geq 0\}$, where l_i denote the i-th entry in l. The interior $\operatorname{int} P$ denotes the set of positive real vectors $\operatorname{int} P = \{l \in \mathbb{R}^m : \forall l_i > 0\}$. P is a closed and convex cone, and the interior $\operatorname{int} P$ induce a partial order for any two points in $\mathbb{R}^m$. That is, for any $l^1, l^2 \in \mathbb{R}^m$, we define

$$l^1 \leq l^2 \iff l^2 - l^1 \in P$$
$$l^1 < l^2 \iff l^2 - l^1 \in \operatorname{int} P.$$

That is, for $l^1, l^2 \in \mathbb{R}^m$, the partial ordering $l^1 \leq l^2$ and $l^1 < l^2$ imply that $l^1_i \leq l^2_i$ and $l^1_i < l^2_i$ for all $i \in \{1, ..., m\}$, respectively. Given $l^1 \in \mathbb{R}^m$, we define $l^1 - P = \{l \in \mathbb{R}^m : l \leq l^1\}$ and $l^1 - \operatorname{int} P = \{l \in \mathbb{R}^m : l < l^1\}$.

We now recall the notions of minimality for a subset in $\mathbb{R}^m$.

Definition 1. For a nonempty set $C \in \mathbb{R}^m$, the set of all minimal points in C *w.r.t.* the ordering cone P is defined as

$$\operatorname{Min} C := \left\{l \in C : C \cap (l - P) = \{l\}\right\}.$$

The weakly minimal points of the set C are

$$\operatorname{WMin} C := \left\{l \in C : C \cap (l - \operatorname{int} P) = \emptyset\right\}.$$

In the MOP, for the given objective function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ $(m, n \in \mathbb{N}, m \geq 2)$, where $z \in \mathcal{Z}$, we denote by $\operatorname{Min} g(z)$ the set of all the minimal points of the function g. We also call it as the Pareto frontier or Pareto-optimal set. Thus, the corresponding efficient solution or Pareto-optimal solution of $g(z)$ can be defined as

$$\operatorname{Eff}(g(z)) := \left\{z \in \mathcal{Z} : g(z) \in \operatorname*{Min}_{z \in \mathcal{Z}} g(z)\right\}.$$

Similarly, we denote by $\operatorname{WMin} g(z)$ the set of weakly minimal points of the function $g(z)$ and by $\operatorname{WEff}(g(z))$ the corresponding weakly efficient solution set.

Definition 2. The function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ is a P-convex function if for every $z_1, z_2 \in \mathbb{R}^n$ and for every $\lambda \in [0, 1]$, we have

$$g(\lambda z_1 + (1 - \lambda) z_2) \leq \lambda g(z_1) + (1 - \lambda) g(z_2).$$

$g(z)$ is a strictly P-convex function, if for every $z_1, z_2 \in \mathbb{R}^n$, $z_1 \neq z_2$ and for every $\lambda \in (0, 1)$,

$$g_i(\lambda z_1 + (1 - \lambda) z_2) < \lambda g_i(z_1) + (1 - \lambda) g_i(z_2).$$

Remark 1. For a given vector-valued function $g(z)$, we have Min $g(z) \subseteq$ WMin $g(z)$. If g is strictly P-convex, we have Min $g(z) =$ WMin $g(z)$ and WEff $(g(z)) =$ Eff $(g(z))$.

Definition 3. Consider $\{A_n\}$ as a sequence of subsets of a set X in an Euclidean space. Li A_n is defined as the lower limit of the sequence of sets $\{A_n\}$, that is,

$$\text{Li } A_n := \{a \in X : a = \lim_{n\to+\infty} a_n, a_n \in A_n, \text{ for sufficiently large } n\}.$$

Ls A_n is defined as the upper limit of the sequence of sets $\{A_n\}$, that is,

$$\text{Ls } A_n := \{a \in X : a = \lim_{n\to+\infty} a_n, a_n \in A_{n_k}, \text{ for } n_k \text{ as a selection of integers.}\}.$$

A sequence $\{A_n\}$ converges in the Kuratowski sense to one set $A \subseteq X$, when

$$\text{Ls } A_n \subseteq A \subseteq \text{Li } A_n,$$

and we denote such convergence by $A_n \to A$.

A.2. Gradient-based optimization algorithm

To solve an unconstrained multi-objective optimization problem, we adopt the Multiple Gradient Descent Algorithm (MGDA) [8]. MGDA finds the minimum-norm point in the convex hull composed by the gradients of multiple objectives. Specifically, MGDA performs the following two steps alternately:

Step 1. Compute the gradients $\nabla_z g_i(z)$ for $i = 1, \dots, m$, and solve the following quadratic programming problem

$$\min_{\gamma} \left\| \sum_{i=1}^{m} \gamma_i \nabla_z g_i(z) \right\|^2 \quad \text{s.t. } \gamma_i \geq 0, \ \sum_{i=1}^{m} \gamma_i = 1, \tag{24}$$

to determine the weights γ_i in the current iteration. γ_i can be viewed as a weight for the i-th objective. To solve problem (24), we can use the Frank-Wolfe algorithm [61]. Then, the descent direction searched is computed as $d = \sum_{i=1}^{m} \gamma_i \nabla_z g_i(z)$.

Step 2. If $d = 0$, the MGDA stops. Otherwise, a line step is determined as the largest positive scalar v, with which all objectives are decreasing. Then we update z as $z - vd$ and go to Step 1.

Remark 2. The original MGDA searches the step size to ensure that all the objectives decrease in each iteration. However, this will result in significant computational complexity for learning models with many parameters such as deep neural networks. Therefore, in Algorithm 1, similar to [61,48], we use a fixed and small step size in MGDA to reduce the computational complexity.

Appendix B. Proofs of theorems in Section 5

For the sake of clarity, we first introduce some notation from [45]. The sublevel set of a function $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ at height $h \in \mathbb{R}^m$ is defined as $g^h := \{z \in \mathbb{R}^n : g(z) \leq h\}$. If A is a closed convex set, we can define then the recession cone of A as $0^+(A) := \{d \in \mathbb{R}^n : a + td \in A, \forall a \in A, \forall t \geq 0\}$. The recession cone of the sublevel set of the function $g(z)$ is denoted by H_g.

To prove theorems in Section 5, we will use Theorems 3.1 and 3.2 in [46], and list them in the following for completeness.

Theorem 7. *Suppose that $\mathcal{Z}$ is a nonempty closed, convex set in $\mathbb{R}^n$ and $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function with $z \in \mathcal{Z}$. Then, if $g_n(z) \to g(z)$ w.r.t. the continuous convergence, we have* LsWMin $g_n(z) \subseteq$ WMin $g(z)$.

Theorem 8. *Suppose that $\mathcal{Z}$ is a nonempty closed, convex set in $\mathbb{R}^n$ and $g(z) : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function with $z \in \mathcal{Z}$. Then, if $g_n(z) \to g(z)$ w.r.t. the continuous convergence, $g_n(z)$ and $g(z)$ are both P-convex functions and $0^+(\mathcal{Z}) \cap H_g = \{0\}$, we have* Min $g(z) \subseteq$ LiMin $g_n(z)$.

B.1. Proof of Theorem 1

Proof. To show that $F(\omega^*(\alpha), \alpha)$ is continuous on α, we need to prove that for any convergent sequence $\alpha_n \to \bar{\alpha}$, $F(\omega^*(\alpha_n), \alpha_n)$ converges to $F(\omega^*(\bar{\alpha}), \bar{\alpha})$.

Suppose that $\{\alpha_n\}$ is a sequence in $\mathcal{A}$ satisfying $\alpha_n \to \bar{\alpha}$. As $\arg\min_\omega f(\omega, \alpha)$ is a singleton, we have $\omega^*(\alpha_n) = \arg\min_\omega f(\omega, \alpha_n)$. Since $\{\omega^*(\alpha)\}$ is bounded for $\alpha \in \mathcal{A}$, according to Bolzano-Weierstrass theorem [1], there exists a convergent subsequence $\{\omega^*(\alpha_{kn})\}$ such that $\omega^*(\alpha_{kn}) \to \bar{\omega}$ for some $\bar{\omega} \in \mathbb{R}^p$. Since $\alpha_{kn} \to \bar{\alpha}$ and $f(\omega, \alpha)$ is jointly continuous, we can get that $\forall \omega \in \mathbb{R}^p$, $f(\bar{\omega}, \bar{\alpha}) = \lim_n f(\omega^*(\alpha_{kn}), \alpha_{kn}) \leq \lim_n f(\omega(\alpha_{kn}), \alpha_{kn}) = f(\omega(\bar{\alpha}), \bar{\alpha})$. Therefore, we obtain $\omega^*(\bar{\alpha}) = \bar{\omega}$. This means $\{\omega^*(\alpha_{kn})\}$ has only one cluster point $\omega^*(\bar{\alpha})$. Thus, $\omega^*(\alpha_n)$ converges to $\omega^*(\bar{\alpha})$ as $\alpha_n \to \bar{\alpha}$. Because F is jointly continuous, we have $F(\omega^*(\alpha_n), \alpha_n) \to F(\omega^*(\bar{\alpha}), \bar{\alpha})$ as $\alpha_n \to \bar{\alpha}$. □

B.2. Proof of Theorem 2

Proof. To prove the first claim of Theorem 2, we firstly show that $\varphi_K(\alpha)$ continuously converges to $\varphi(\alpha)$. Suppose there exists a sequence $\{\alpha_n\}$ in $\mathcal{A}$ satisfying $\alpha_n \to \alpha$. Then for any $\varphi_K(\alpha)$ and sequence α_n, we have

$$\|\varphi_K(\alpha_n) - \varphi(\alpha)\| = \|F(\omega_K(\alpha_n), \alpha_n) - F(\omega^*(\alpha), \alpha)\| \tag{25}$$

$$\leq \|F(\omega_K(\alpha_n), \alpha_n) - F(\omega^*(\alpha_n), \alpha_n)\| + \|F(\omega^*(\alpha_n), \alpha_n) - F(\omega^*(\alpha), \alpha)\|. \tag{26}$$

According to the continuity property in Theorem 1, we have $F(\omega^*(\alpha_n), \alpha_n) \to F(\omega^*(\alpha), \alpha)$ as $\alpha_n \to \alpha$. Furthermore, because $F(\cdot, \alpha)$ is uniformly Lipschitz continuous, we have

$$\|F(\omega_K(\alpha_n), \alpha_n) - F(\omega^*(\alpha_n), \alpha_n)\| \leq L\|\omega_K(\alpha_n) - \omega^*(\alpha_n)\|. \tag{27}$$

According to assumption (i) in Theorem 2, $\omega_K(\alpha)$ converges to $\omega^*(\alpha)$ uniformly as $K \to +\infty$. Therefore, $\varphi_K(\alpha)$ continuously converges to $\varphi(\alpha)$.

Since $\operatorname{Min} \varphi(\alpha) \subseteq \operatorname{WMin} \varphi(\alpha)$ and Theorem 7, we have

$$\operatorname{LsMin} \varphi_K(\alpha) \subseteq \operatorname{LsWMin} \varphi_K(\alpha) \subseteq \operatorname{WMin} \varphi(\alpha). \tag{28}$$

Because $\mathcal{A}$ is a compact convex set in $\mathbb{R}^n$, $0^+(\mathcal{A}) = \{0\}$. Then, the condition $0^+(\mathcal{A}) \cap H_\varphi = \{0\}$ is naturally satisfied for function $\varphi(\alpha)$. According to assumption (iii) in Theorem 2, $\varphi(\alpha)$ and $\varphi_K(\alpha)$ are both P-convex functions. Then we obtain the lower part of the set convergence from Theorem 8 as

$$\operatorname{Min} \varphi(\alpha) \subseteq \operatorname{LiMin} \varphi_K(\alpha) \subseteq \operatorname{LiWMin} \varphi_K(\alpha). \tag{29}$$

Because $\varphi(\alpha)$ is strictly P-convex, we have $\operatorname{WMin} \varphi = \operatorname{Min} \varphi$ and then we get $\operatorname{Min} \varphi_K(\alpha) \to \operatorname{Min} \varphi(\alpha)$ according to Definition 3.

For the second claim, let $\alpha_n \in \operatorname{Eff} \varphi_K(\alpha)$ and $\alpha_n \to \bar{\alpha}$. Since $\operatorname{Min} \varphi_K(\alpha) \to \operatorname{Min} \varphi(\alpha)$, we get $\varphi_K(\alpha_n) \to \varphi(\bar{\alpha})$ and $\bar{\alpha} \in \operatorname{Min} \varphi(\alpha)$, which implies $\operatorname{LsEff} \varphi_K(\alpha) \subseteq \operatorname{Eff} \varphi(\alpha)$.

For the lower limit, by defining $\bar{\alpha} \in \operatorname{Eff} \varphi(\alpha)$, the corresponding minimal point satisfies $\bar{l} = \varphi(\bar{\alpha}) \in \operatorname{Min} \varphi(\alpha)$. Based on the first claim, there is a sequence $\{l_K\}$ in $\operatorname{Min} \varphi_K(\alpha)$ such that $l_K \to \bar{l}$. Then we can take a bounded sequence $\{\alpha_K\}$, where $\alpha_K = \varphi_K^{-1}(l_K)$ and the subsequence of $\{\alpha_K\}$ has a cluster point. Because $\varphi(\alpha)$ is strictly P-convex, this cluster point is $\bar{\alpha}$. Then, we have $\alpha_K \to \bar{\alpha}$, which implies $\operatorname{Eff} \varphi(\alpha) \subseteq \operatorname{LiEff} \varphi_K(\alpha)$. Combined with the upper limit convergence, we can get $\operatorname{Eff} \varphi_K(\alpha) \to \operatorname{Eff} \varphi(\alpha)$. □

Remark 3. In fact, if we consider the weakly minimal points under the same assumptions in Theorems 1 and 2, we can still obtain similar convergence results to those in Theorem 2, *i.e.*,

$$\operatorname{WMin} \varphi_K(\alpha) \to \operatorname{WMin} \varphi(\alpha), \ \operatorname{WEff} \varphi_K(\alpha) \to \operatorname{WEff} \varphi(\alpha).$$

Since $\varphi(\alpha)$ is strictly P-convex, the first claim can be directly obtained according to Eqs. (28) and (29). Then, the proof of the convergence of the weakly efficient solution follows that of Theorem 2.

B.3. Proof of Theorem 3

Proof. Theorem 3 can be directly obtained from Lemma 6 of [25]. □

B.4. Proof of Theorem 4

Proof. For the sake of notation simplicity, we denote the i-th entry of the true gradient by $g_i(\alpha_t) = \nabla\varphi_i(\alpha)$ and the approximated gradient by $\widetilde{g}_i(\alpha_t) = \frac{\partial F_i(\omega_K(\alpha_t), \alpha_t)}{\partial \alpha_t}$. We denote by γ_t the corresponding convex combination vector calculated based on $g(\alpha_t)$. The corresponding weights calculated by $\widetilde{g}(\alpha_t)$ are denoted by $\widetilde{\gamma}_t$. According to Theorem 3, we have $\|\widetilde{g}_i(\alpha_t) - g_i(\alpha_t)\| \leq \Gamma(K)$. Then we get

$$\|\Lambda(\widetilde{g}(\alpha_t), \widetilde{\gamma}_t)\|^2 \leq \|\widetilde{g}_{\max}(\alpha_t)\|^2 \leq 2L^2 + 2\Gamma(K)^2, \tag{30}$$

where the second inequality is due to the triangle inequality and $L = \max_{1\leq i\leq m} L_i$. By setting $A_t = \alpha_t - \alpha_*$, we have

$$\|A_{t+1}\|^2 = \|A_t - v_t\Lambda(\widetilde{g}(\alpha_t), \widetilde{\gamma}_t)\|^2 = \|A_t\|^2 - 2v_t\langle A_t, \Lambda(\widetilde{g}(\alpha_t), \widetilde{\gamma}_t)\rangle + v_t^2\|\Lambda(\widetilde{g}(\alpha_t), \widetilde{\gamma}_t)\|^2. \tag{31}$$

Since the i-th entry of φ_K is c_i-strongly-convex, then for any $\widetilde{\gamma}$, function $\Lambda(\varphi_K(\alpha), \widetilde{\gamma})$ is c-strongly-convex, where $c = \min_{1\leq i\leq m} c_i$. Then, for a given vector $\widetilde{\gamma}$, we have

$$\Lambda(\varphi_K(\alpha^*), \widetilde{\gamma}_t) \geq \Lambda(\varphi_K(\alpha_t), \widetilde{\gamma}_t) + \Lambda(\nabla\varphi_K(\alpha_t), \widetilde{\gamma}_t)^\top(\alpha^* - \alpha_t) + \frac{c}{2}\|\alpha^* - \alpha_t\|^2. \tag{32}$$

We define $S_t = \Lambda(\varphi_K(\alpha_t), \widetilde{\gamma}_t) - \Lambda(\varphi_K(\alpha^*), \widetilde{\gamma}_t)$. Then, plugging inequalities (30) and (31) into (32), we can have

$$2\nu_t S_t \le (1-\nu_t c)\|A_t\|^2 + \nu_t^2(2L^2 + 2\Gamma(K)^2) - \|A_{t+1}\|^2. \tag{33}$$

By setting $\nu_t = \frac{2}{c(t+1)}$, we obtain

$$S_t \le \frac{c(t-1)}{4}\|A_t\|^2 - \frac{c(t+1)}{4}\|A_{t+1}\|^2 + \frac{2(L^2+\Gamma(K)^2)}{c(t+1)}. \tag{34}$$

Multiplying by t on both sides of (34), and summing over $t = 1, ..., T$ yields

$$\sum_{t=1}^{T} tS_t \le \frac{-ct(t+1)}{4}\|A_{t+1}\|^2 + \sum_{t=1}^{T}\frac{2t(L^2+\Gamma(K)^2)}{c(t+1)} \le \frac{2T(L^2+\Gamma(K)^2)}{c}. \tag{35}$$

Dividing both sides by $\sum_{t=1}^{T} t$ gives us

$$\frac{\sum_{t=1}^{T} t\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}_t) - \sum_{t=1}^{T} t\Lambda(\varphi_K(\alpha_*),\widetilde{\gamma}_t)}{\sum_{t=1}^{T} t} \le \frac{4(L^2+\Gamma(K)^2)}{c(T+1)}. \tag{36}$$

By setting $\bar{\gamma}_t = \frac{\sum_{t=1}^{T} t\widetilde{\gamma}_t}{\sum_{t=1}^{T} t}$, we get

$$\min_{t=1,...,T} \Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}_t) - \Lambda(\varphi_K(\alpha_*),\bar{\gamma}_t) \le \frac{\sum_{t=1}^{T} t\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}_t) - \sum_{t=1}^{T} t\Lambda(\varphi_K(\alpha_*),\widetilde{\gamma}_t)}{\sum_{t=1}^{T} t}. \tag{37}$$

Thus, the proof is completed by combining (36) and (37). □

B.5. Proof of Theorem 5

Proof. Since the $\bar{\alpha}^*$ and α^* are the unique solution of the objectives $\Lambda(\alpha(\alpha),\widetilde{\gamma}^*)$ and $\Lambda(\alpha_K(\alpha),\widetilde{\gamma}^*)$, respectively. Then, according to the optimality condition, we have $\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi_K(\alpha^*),\widetilde{\gamma}^*)\| = 0$. For a given $\widetilde{\gamma}^*$, by using Theorem 3, we have $\|\Lambda(\nabla\varphi_K(\alpha^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \le \Gamma(K)$. Therefore, using triangle inequality, we have

$$\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \le \Gamma(K). \tag{38}$$

Since the i-th entry of φ is $\hat{c}_i$-strongly-convex, then for any $\widetilde{\gamma}^*$, function $\Lambda(\varphi(\alpha),\widetilde{\gamma}^*)$ is $\hat{c}$-strongly-convex, where $\hat{c} = \min_{1\le i\le m}\hat{c}_i$. Then, for a given vector $\widetilde{\gamma}^*$, we have

$$(\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*))^\top(\bar{\alpha}^* - \alpha^*) \ge \hat{c}\|\bar{\alpha}^* - \alpha^*\|. \tag{39}$$

Note that $\mathcal{A}$ is a bounded set, there exists a positive diameter δ such that $\|\alpha - \alpha^*\| \le \delta \le \infty$ holds for any two points α and α^* in $\mathcal{A}$. Then we have

$$\|\bar{\alpha}^* - \alpha^*\| \le \frac{\delta}{\hat{c}}\|\Lambda(\nabla\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\nabla\varphi(\alpha^*),\widetilde{\gamma}^*)\| \le \frac{\delta}{\hat{c}}\Gamma(K), \tag{40}$$

where the first inequality is due to the Cauchy-Schwarz inequality. Based on the strong-convexity of the lower-level function $f(\cdot,\alpha)$, for given α, we have $\|\omega^*(\alpha) - \omega_K(\alpha)\| \le (1-\mu\vartheta)^K\|\omega^0 - \omega^*(\alpha)\|$, where ω^0 is the initialization of ω in the inner loop. Then we have

$$\|\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\varphi_K(\alpha^*),\widetilde{\gamma}^*)\| \le \|\Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*) - \Lambda(\varphi(\alpha^*),\widetilde{\gamma}^*)\| + \|\Lambda(\varphi(\alpha^*),\widetilde{\gamma}^*) - \Lambda(\varphi_K(\alpha^*),\widetilde{\gamma}^*)\| \tag{41}$$

$$\le L\|\alpha^* - \bar{\alpha}^*\| + \|\varphi_K(\alpha^*) - \varphi(\alpha^*)\| \tag{42}$$

$$\le \frac{\delta L}{\hat{c}}\Gamma(K) + (1-\mu\vartheta)^K\hat{L}\|\omega^0 - \omega^*(\alpha_t)\|, \tag{43}$$

where the second inequality is due to the Cauchy-Schwarz inequality and the Lipschitz assumption of the function $\varphi(\alpha)$, and the third inequality is due to the Lipschitz assumption of the functions F_i w.r.t. ω. This finishes the proof. □

B.6. Proof of Theorem 6

Proof. For a given $\widetilde{\gamma}_t$, the function $\Lambda(\varphi_K(\alpha),\widetilde{\gamma}_t)$ is c-strongly-convex. Since $\Lambda(\nabla\varphi_K(\alpha^*),\widetilde{\gamma}_t)^\top(\alpha_t - \alpha^*) \ge 0$, we have

$$\Lambda(\nabla\varphi_K(\alpha_t),\widetilde{\gamma}_t)^\top(\alpha_t - \alpha^*) \le [\Lambda(\nabla\varphi_K(\alpha_t),\widetilde{\gamma}_t) - \Lambda(\nabla\varphi_K(\alpha^*),\widetilde{\gamma}_t)]^\top(\alpha_t - \alpha^*) \le c\|\alpha_t - \alpha^*\|^2. \tag{44}$$

Then, plugging inequalities (30) and (31) into (44), we have

$$\|A_{t+1}\|^2 \le (1-2\nu_t c)\|A_t\|^2 + \nu_t^2 L^2. \tag{45}$$

Therefore, with $\nu_t = \xi/t$ and $\xi \ge 1/2c$, we have $\|\alpha_t - \alpha^*\| \le \max\{2\xi L(2c\xi-1)^{-1}, L\|\alpha^0 - \alpha^*\|^2\}/t$. Therefore, for a given $\widetilde{\gamma}^*$, we have

$$\|\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}^*) - \Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\| \leq \|\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}^*) - \Lambda(\varphi_K(\alpha^*),\widetilde{\gamma}^*)\| + \|\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}^*) - \Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\| \tag{46}$$

$$\leq M\|\alpha_t - \alpha^*\| + \|\Lambda(\varphi_K(\alpha_t),\widetilde{\gamma}^*) - \Lambda(\varphi(\bar{\alpha}^*),\widetilde{\gamma}^*)\|\|, \tag{47}$$

where the first inequality is due to the triangle inequality and the second inequality is due to the Cauchy-Schwarz inequality and the Lipschitz assumption of the function $\varphi_K(\alpha)$. Thus, the proof is completed by using the upper bound of $\|\alpha_t - \alpha^*\|$ and the result in Theorem 5 directly. □

References

[1] R.G. Bartle, D.R. Sherbert, Introduction to Real Analysis, vol. 2, Wiley, New York, 2000.
[2] H. Cai, C. Gan, T. Wang, Z. Zhang, S. Han, Once-for-all: train one network and specialize it for efficient deployment, in: Proceedings of the 8th International Conference on Learning Representations, 2020.
[3] R. Caruana, Multitask learning, Mach. Learn. 28 (1997) 41–75.
[4] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the 14th European Conference on Computer Vision, 2018, pp. 833–851.
[5] X. Chen, A. Ghadirzadeh, M. Björkman, P. Jensfelt, Meta-learning for multi-objective reinforcement learning, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019, pp. 977–983.
[6] K. Deb, A. Sinha, Solving bilevel multi-objective optimization problems using evolutionary algorithms, in: Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2009, pp. 110–124.
[7] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.
[8] J.A. Désidéri, Multiple-gradient descent algorithm (MGDA) for multiobjective optimization, C. R. Math. 350 (2012) 313–318.
[9] J. Domke, Generic methods for optimization-based modeling, in: Proceedings of the 15th Artificial Intelligence and Statistics, 2012, pp. 318–326.
[10] G. Eichfelder, Twenty years of continuous multiobjective optimization in the twenty-first century, EURO J. Comput. Optim. 9 (2021) 100014.
[11] A. Fallah, A. Mokhtari, A. Ozdaglar, On the convergence theory of gradient-based model-agnostic meta-learning algorithms, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2020, pp. 1082–1092.
[12] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, C. Finn, Measuring and harnessing transference in multi-task learning, arXiv preprint, arXiv:2010.15413, 2020.
[13] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1126–1135.
[14] J. Fliege, A.I.F. Vaz, L.N. Vicente, Complexity of gradient descent for multiobjective optimization, Optim. Methods Softw. 34 (2019) 949–959.
[15] L. Franceschi, M. Donini, P. Frasconi, M. Pontil, Forward and reverse gradient-based hyperparameter optimization, in: International Conference on Machine Learning, 2017.
[16] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, M. Pontil, Bilevel programming for hyperparameter optimization and meta-learning, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 1568–1577.
[17] Y. Ganin, V.S. Lempitsky, Unsupervised domain adaptation by backpropagation, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1180–1189.
[18] I. Giagkiozis, R.C. Purshouse, P.J. Fleming, An overview of population-based algorithms for multi-objective optimisation, Int. J. Syst. Sci. 46 (2015) 1572–1599.
[19] Y. Grandvalet, Y. Bengio, Semi-supervised learning by entropy minimization, in: Advances in Neural Information Processing Systems, 2004, pp. 529–536.
[20] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al., Soft actor-critic algorithms and applications, arXiv preprint, arXiv:1812.05905, 2018.
[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[22] N. Hilliard, L. Phillips, S. Howland, A. Yankov, C.D. Corley, N.O. Hodas, Few-shot learning with metric-agnostic conditional embeddings, arXiv preprint, arXiv:1802.04376, 2018.
[23] T. Hospedales, A. Antoniou, P. Micaelli, A. Storkey, Meta-learning in neural networks: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2021) 5149–5169.
[24] M. Huisman, J.N. van Rijn, A. Plaat, A survey of deep meta-learning, Artif. Intell. Rev. 54 (2021) 4483–4541.
[25] K. Ji, J. Yang, Y. Liang, Bilevel optimization: convergence analysis and enhanced design, in: International Conference on Machine Learning, PMLR, 2021, pp. 4882–4892.
[26] Y. Ji, S. Qu, Z. Yu, A new method for solving multiobjective bilevel programs, Discrete Dyn. Nat. Soc. 2017 (2017).
[27] X. Jin, J. Wang, J. Slocum, M.H. Yang, S. Dai, S. Yan, J. Feng, Rc-darts: resource constrained differentiable architecture search, arXiv preprint, arXiv:1912.12814, 2019.
[28] T. Kim, C. Kim, Attract, perturb, and explore: learning a feature alignment network for semi-supervised domain adaptation, in: Proceedings of the 16th European Conference on Computer Vision, 2020, pp. 591–607.
[29] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, 2015.
[30] V. Konda, J. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, 1999.
[31] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[32] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, in: Proceedings of the 5th International Conference on Learning Representations, 2017.
[33] D. Li, T.M. Hospedales, Online meta-learning for multi-source and semi-supervised domain adaptation, in: Proceedings of the 16th European Conference on Computer Vision, 2020, pp. 382–403.
[34] K. Li, J. Malik, Learning to optimize, arXiv preprint, arXiv:1606.01885, 2016.
[35] B. Lin, F. Ye, Y. Zhang, I. Tsang, Reasonable effectiveness of random weighting: a litmus test for multi-task learning, Trans. Mach. Learn. Res. (2022).
[36] B. Lin, Y. Zhang, LibMTL: a Python library for multi-task learning, J. Mach. Learn. Res. 24 (2023) 1–7.
[37] B. Liu, X. Liu, X. Jin, P. Stone, Q. Liu, Conflict-averse gradient descent for multi-task learning, in: Advances in Neural Information Processing Systems, 2021, pp. 18878–18890.
[38] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: Proceedings of the 7th International Conference on Learning Representations, 2019.
[39] R. Liu, J. Gao, J. Zhang, D. Meng, Z. Lin, Investigating bi-level optimization for learning and vision from a unified perspective: a survey and beyond, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[40] R. Liu, P. Mu, X. Yuan, S. Zeng, J. Zhang, A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton, in: International Conference on Machine Learning, PMLR, 2020, pp. 6305–6315.
[41] S. Liu, E. Johns, A.J. Davison, End-to-end multi-task learning with attention, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1871–1880.

[42] M. Long, Z. Cao, J. Wang, M.I. Jordan, Conditional adversarial domain adaptation, in: Advances in Neural Information Processing Systems, 2018, pp. 1647–1657.
[43] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 2208–2217.
[44] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, V.N. Boddeti, NSGANetv2: evolutionary multi-objective surrogate-assisted neural architecture search, in: Proceedings of the 16th European Conference on Computer Vision, 2020, pp. 35–51.
[45] R. Lucchetti, Convexity and Well-Posed Problems, Springer Science & Business Media, 2006.
[46] R. Lucchetti, E. Miglierina, Stability for convex vector optimization problems, Optimization 53 (2004) 517–528.
[47] D. Mahapatra, V. Rajan, Multi-task learning with user preferences: gradient descent with controlled ascent in Pareto optimization, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 6597–6607.
[48] N. Milojkovic, D. Antognini, G. Bergamin, B. Faltings, C. Musat, Multi-gradient descent for multi-objective recommender systems, arXiv preprint, arXiv:2001.00846, 2019.
[49] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533.
[50] A. Nichol, J. Achiam, J. Schulman, On first-order meta-learning algorithms, arXiv preprint, arXiv:1803.02999, 2018.
[51] J. Oh, H. Yoo, C. Kim, S.Y. Yun, Boil: towards representation change for few-shot learning, in: Proceedings of the 9th International Conference on Learning Representations, 2021.
[52] F. Pedregosa, Hyperparameter optimization with approximate gradient, in: Proceedings of the 33rd International Conference on Machine Learning, PMLR, 2016, pp. 737–746.
[53] H. Pham, M.Y. Guan, B. Zoph, Q.V. Le, J. Dean, Efficient neural architecture search via parameter sharing, in: Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 4092–4101.
[54] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M.P. Reyes, M.L. Shyu, S.C. Chen, S. Iyengar, A survey on deep learning: algorithms, techniques, and applications, ACM Comput. Surv. 51 (2018) 1–36.
[55] S. Qu, M. Goh, F.T. Chan, Quasi-Newton methods for solving multiobjective optimization, Oper. Res. Lett. 39 (2011) 397–399.
[56] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning, in: Proceedings of the 5th International Conference on Learning Representations, 2017.
[57] S. Ruuska, K. Miettinen, Constructing evolutionary algorithms for bilevel multiobjective optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2012, pp. 1–7.
[58] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: Proceedings of the 6th European Conference on Computer Vision, 2010, pp. 213–226.
[59] K. Saito, D. Kim, S. Sclaroff, T. Darrell, K. Saenko, Semi-supervised domain adaptation via minimax entropy, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8050–8058.
[60] K. Saito, Y. Ushiku, T. Harada, K. Saenko, Adversarial dropout regularization, in: Proceedings of the 6th International Conference on Learning Representations, 2018, OpenReview.net.
[61] O. Sener, V. Koltun, Multi-task learning as multi-objective optimization, in: Advances in Neural Information Processing Systems, 2018, pp. 525–536.
[62] A. Shaban, C.A. Cheng, N. Hatch, B. Boots, Truncated back-propagation for bilevel optimization, in: Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, 2019, pp. 1723–1732.
[63] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: Proceedings of the 8th European Conference on Computer Vision, 2012, pp. 746–760.
[64] A. Sinha, Bilevel multi-objective optimization problem solving using progressively interactive emo, in: Proceedings of the 6th International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2011, pp. 269–284.
[65] A. Sinha, P. Malo, K. Deb, Towards understanding bilevel multi-objective optimization with deterministic lower level decisions, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2015, pp. 426–443.
[66] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.
[67] S. Sodhani, A. Zhang, J. Pineau, Multi-task reinforcement learning with context-based representations, in: International Conference on Machine Learning, PMLR, 2021, pp. 9767–9779.
[68] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H. Torr, T.M. Hospedales, Learning to compare: relation network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1199–1208.
[69] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, Q.V. Le, MnasNet: platform-aware neural architecture search for mobile, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2820–2828.
[70] H. Tanabe, E.H. Fukuda, N. Yamashita, Proximal gradient methods for multiobjective optimization and their applications, Comput. Optim. Appl. 72 (2019) 339–361.
[71] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: maximizing for domain invariance, arXiv preprint, arXiv:1412.3474, 2014.
[72] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.
[73] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, in: Advances in Neural Information Processing Systems, 2016, pp. 3630–3638.
[74] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200-2011 dataset, 2011.
[75] X. Wang, Y. Tsvetkov, G. Neubig, Balancing training for multilingual neural machine translation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 8526–8537.
[76] Y. Wang, Q. Yao, J.T. Kwok, L.M. Ni, Generalizing from a few examples: a survey on few-shot learning, ACM Comput. Surv. 53 (2020) 1–34.
[77] Z. Wang, Z.C. Lipton, Y. Tsvetkov, On negative interference in multilingual models: findings and A meta-learning treatment, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 4438–4450.
[78] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, K. Keutzer, Fbnet: hardware-aware efficient convnet design via differentiable neural architecture search, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 10734–10742.
[79] S. Xie, H. Zheng, C. Liu, L. Lin, SNAS: stochastic neural architecture search, in: Proceedings of the 7th International Conference on Learning Representations, 2019.
[80] Z. Xu, H. van Hasselt, D. Silver, Meta-gradient reinforcement learning, in: Advances in Neural Information Processing Systems, 2018, pp. 2402–2413.
[81] Q. Yang, Y. Zhang, W. Dai, S.J. Pan, Transfer Learning, Cambridge University Press, 2020.
[82] R. Yang, H. Xu, Y. Wu, X. Wang, Multi-task reinforcement learning with soft modularization, Adv. Neural Inf. Process. Syst. 33 (2020) 4767–4777.
[83] Y.Y. Yang, C. Rashtchian, H. Zhang, R. Salakhutdinov, K. Chaudhuri, A closer look at accuracy vs. robustness, in: Advances in Neural Information Processing Systems, 2020.
[84] F. Ye, B. Lin, Z. Yue, P. Guo, Q. Xiao, Y. Zhang, Multi-objective meta learning, in: Advances in Neural Information Processing Systems, 2021.
[85] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, C. Finn, Gradient surgery for multi-task learning, in: Advances in Neural Information Processing Systems, 2020.
[86] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, S. Levine, Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning, in: Conference on Robot Learning, PMLR, 2020, pp. 1094–1100.

[87] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Trans. Knowl. Data Eng. 34 (2022) 5586–5609.
[88] A. Zhou, B.Y. Qu, H. Li, S.Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (2011) 32–49.
[89] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, Q. He, Deep subdomain adaptation network for image classification, IEEE Trans. Neural Netw. Learn. Syst. 32 (2021) 1713–1722.