Memory-Efficient Gradient Unrolling for Large-Scale Bi-level Optimization

Qianli Shen[1]* **Yezhen Wang**[1] **Zhouhao Yang**[1] **Xiang Li**[1] **Haonan Wang**[1]
Yang Zhang[1] **Jonathan Scarlett**[1] **Zhanxing Zhu**[2] **Kenji Kawaguchi**[1]
[1]National University of Singapore [2]University of Southampton, UK

Abstract

Bi-level optimization (BO) has become a fundamental mathematical framework for addressing hierarchical machine learning problems. As deep learning models continue to grow in size, the demand for scalable bi-level optimization has become increasingly critical. Traditional gradient-based bi-level optimization algorithms, due to their inherent characteristics, are ill-suited to meet the demands of large-scale applications. In this paper, we introduce **F**orward **G**radient **U**nrolling with **F**orward **G**radient, abbreviated as **(FG)^{2}U**, which achieves an unbiased stochastic approximation of the meta gradient for bi-level optimization. (FG)^{2}U circumvents the memory and approximation issues associated with classical bi-level optimization approaches, and delivers significantly more accurate gradient estimates than existing large-scale bi-level optimization approaches. Additionally, (FG)^{2}U is inherently designed to support parallel computing, enabling it to effectively leverage large-scale distributed computing systems to achieve significant computational efficiency. In practice, (FG)^{2}U and other methods can be strategically placed at different stages of the training process to achieve a more cost-effective two-phase paradigm. Further, (FG)^{2}U is easy to implement within popular deep learning frameworks, and can be conveniently adapted to address more challenging black-box bi-level optimization scenarios. We provide a thorough convergence analysis and a comprehensive practical discussion for (FG)^{2}U, complemented by extensive empirical evaluations, showcasing its superior performance in diverse large-scale bi-level optimization tasks. Code is available at `https://github.com/ShenQianli/FG2U`.

1 Introduction

Bi-level optimization is a mathematical framework with a long history of research [10, 65, 73], dealing with hierarchical optimization problems where one problem is nested within the other. A bi-level optimization problem can be formulated as:

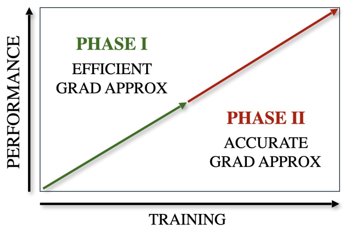
$$\min_{\boldsymbol{\phi}} \; f(\boldsymbol{\theta}^*(\boldsymbol{\phi}), \boldsymbol{\phi}) \quad s.t.\; \boldsymbol{\theta}^*(\boldsymbol{\phi}) \in \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}, \boldsymbol{\phi}), \tag{1}$$

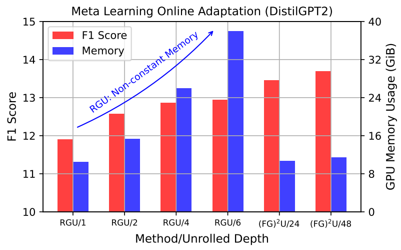
where $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^M$ denotes the inner parameter, $\boldsymbol{\phi} \in \Phi \subseteq \mathbb{R}^N$ denotes the meta parameter, and f, g are called the meta objective function and inner objective function, respectively.

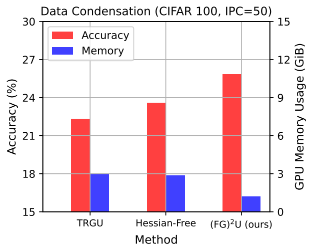
Recently, with the rise of deep learning, bi-level optimization has regained attention as a theoretical framework covering a wide range of machine learning problems, including hyperparameter optimization [46, 43, 17, 16, 45], neural architecture search [78, 38, 14], robust machine learning [79, 76, 71, 26], meta learning [15, 53, 49, 2], and physics-informed machine learning [23, 62]. In these scenarios, the inner problem often pertains to the optimization of neural networks, thereby

*Correspondence: `shenqianli@u.nus.edu`

Category	Method	Constant Memory	Hessian-Free	Stochastic Optimization	Grad. Approx. Preference
Classical	GU [17, 16]	✗	✓	✓	-
	IF [53, 19, 64]	✓	✗	✓	-
	VF [39, 37, 61, 33]	✓	✓	✗	-
Large-scale	TRGU [60]	✓	✓	✓	Efficiency
	Hessian-Free [76, 75, 9]	✓	✓	✓	Efficiency
	(FG)^{2}U (ours)	✓	✓	✓	Accuracy







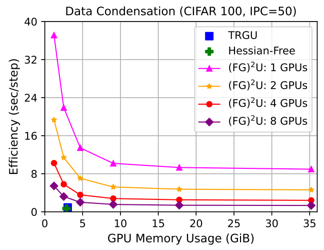


Figure 1: **Top Left**: A comparison of bi-level optimization methods. (FG)^{2}U circumvents the large-scale challenges inherent in classical bi-level optimization techniques. Within large-scale bi-level optimization, (FG)^{2}U prioritizes the accuracy of gradient approximation over efficiency. **Top Right**: An overview of the cost-effective two-phase paradigm. (FG)^{2}U is ideally positioned in Phase II to enhance performance after an approximate solution has been obtained using other efficient methods. **Bottom Left**: GPU Memory Usage and Performance on *Meta Learning Online Adaptation* experiment. (FG)^{2}U can effectively address the memory issue of RGU when both the inner model and the unrolled depth are large. **Bottom Center**: GPU Memory Usage and Performance on *Data Condensation* experiments. The performance of (FG)^{2}U surpasses that of other large-scale bi-level optimization methods, owing to its accurate gradient approximation, while demonstrating better memory efficiency. **Bottom Right**: Efficiency tradeoff of (FG)^{2}U on *Data Condensation* experiments. The efficiency of (FG)^{2}U can be well enhanced via intra/inter-GPU parallelism.

precipitating challenges associated with gradient-based bi-level optimization. Consequently, various gradient-based bi-level optimization algorithms have been developed [73]. These algorithms typically employ an iterative solution θ_T obtained by executing multiple inner optimization steps to approximate the meta gradient, and provide different tradeoffs between computational costs and performance for meta gradient approximation.

However, as the scale of deep learning models continues to expand, the requirements for scalability in bi-level optimization correspondingly increase. Existing gradient-based bi-level optimization algorithms, due to their inherent characteristics, are ill-suited to meet the demands of large-scale applications. Concretely, *gradient unrolling* (GU) methods [17, 16, 40, 60] are bottlenecked by the memory overhead associated with either the dimension of the inner parameter or the number of iterative steps for the inner problem. Implicit Function (IF) approaches [48, 19, 64, 76] are compromised by approximation errors, which stem from the iterative estimation of inner solutions and computations that involve the Hessian matrix. *Value Function* (VF) based strategies [39, 37, 61, 33], although exhibit commendable theoretical properties [8] for deterministic bi-level optimization, have yet to gain traction in practical applications, predominantly due to their limitations in addressing large-scale stochastic challenges [73]. Recent advancements in algorithms [60, 9] have been specifically tailored for large-scale bi-level optimization. Although these methodologies facilitate efficient gradient approximation by compromising accuracy, they may result in significantly suboptimal performance due to biased gradient approximations. Additionally, these methods struggle in more complex scenarios, such as when inner problems are addressed through black-box optimization.

In this paper, we propose a novel method called **F**orward **G**radient **U**nrolling with **F**orward **G**radient, abbreviated as **(FG)^{2}U**, which achieves an unbiased stochastic approximation of the meta gradient for bi-level optimization. (FG)^{2}U circumvents the memory issues associated with GU-based approaches and approximation issues associated with IF-based approaches. Compared to recently developed large-scale bi-level optimization approaches, (FG)^{2}U delivers significantly more accurate gradient estimates. Additionally, (FG)^{2}U is inherently designed to support parallel computing, enabling it to effectively leverage large-scale distributed computing systems to achieve significant computational efficiency. In practice, a cost-effective two-phase paradigm can be achieved by strategically placing

(FG)^{2}U and other methods at different stages of the training process to balance efficiency and performance. Further, (FG)^{2}U is easy to implement within popular deep learning frameworks, and can be conveniently adapted to address more challenging zeroth-order bi-level optimization scenarios.

We provide an overview of (FG)^{2}U in Figure 1 to illustrate its strengths and role in large-scale bi-level optimization. The rest of the paper is organized as follows. Firstly, in Section 2, we provide summaries of existing bi-level optimization algorithms and discuss their limitations in large-scale contexts. Next, in Section 3, we introduce the proposed method, (FG)^{2}U, followed by a convergence analysis in Section 3.1 and a detailed discussion of the practical considerations in Section 3.2. Further, in Section 4, we conduct extensive empirical studies covering large-scale bi-level optimization in computer vision, natural language processing, and physics-informed machine learning to demonstrate the efficacy of (FG)^{2}U in large-scale bi-level optimization scenarios.

2 Background

Gradient-based Bi-level Optimization. Within deep learning applications, the model concerned with optimizing over $\boldsymbol{\theta}$ as presented in (1) typically constitutes deep neural networks. The optimal parameters of such networks are not explicitly accessible and are estimated through iterative procedures. Consequently, the primal problem of bi-level optimization in (1) is approximately reformulated as follows:

$$\min_{\boldsymbol{\phi}\in\Phi} h(\boldsymbol{\phi}) := f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi}), \tag{2}$$
$$\text{where } \boldsymbol{\theta}_0(\boldsymbol{\phi}) = \boldsymbol{\Omega}_0(\boldsymbol{\phi}),\ \ \boldsymbol{\theta}_t(\boldsymbol{\phi}) = \boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi}) \in \Theta,\ \ t = 1, \ldots, T,$$

where $\Phi \subseteq \mathbb{R}^N$, $\Theta \subseteq \mathbb{R}^M$ are the parameter spaces; T, commonly called the unrolled depth, denotes the number of inner optimization steps for approximating $\boldsymbol{\theta}^*(\boldsymbol{\phi})$; $\boldsymbol{\Omega}_0 : \mathbb{R}^N \to \mathbb{R}^M$ specifies the initialization of the inner optimization, and $\boldsymbol{\Omega}_t : \Theta \times \Phi \to \Phi$ delineates the transition dynamics of the inner optimization at timestep t. In particular, for gradient descent, $\boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi}) = \boldsymbol{\theta}_{t-1} - \eta_t \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi})$, where η_t denotes the step size at timestep t.

To optimize $\boldsymbol{\phi}$ using a first-order method, it is necessary to estimate the meta gradient $\nabla_{\boldsymbol{\phi}} h$, which can be further decomposed according to the chain rule:

$$\underbrace{\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi})}_{\text{meta gradient}} = \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T} \frac{d\boldsymbol{\theta}_T(\boldsymbol{\phi})}{d\boldsymbol{\phi}}}_{\text{implicit gradient}} + \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}}_{\text{explicit gradient}}. \tag{3}$$

The computation of meta-gradient poses a significant challenge, primarily due to the need for efficient approximation of the implicit gradient. This task is complicated by the recursive dependency of $\boldsymbol{\theta}_T$ on $\boldsymbol{\phi}$. To surmount this challenge, a variety of gradient-based bi-level optimization algorithms have been developed, as extensively reviewed recently in [73]. These algorithms can be fundamentally categorized into three types based on their approach to meta-gradient approximation: *Gradient Unrolling* (GU), *Implicit Function* (IF), and *Value Function* (VF). Recent innovations such as truncated RGU (TRGU) [60] and Hessian-Free approaches [76, 75, 9], which are predicated on GU and IF methodologies respectively, have introduced significant biases in their approximations to accommodate the computational constraints of large-scale scenarios. In the subsequent paragraph, we furnish a concise overview of GU-based approaches, addressing their non-constant memory issues in large-scale applications. Extended discussions on the remaining methods are reserved for Appendix B.

Gradient Unrolling. The core idea behind GU [17, 16, 40, 60] entails unrolling the inner optimization into an expansive computational graph, followed by the employment of automatic differentiation (AD) techniques for the iterative computation of gradients.

Forward Gradient Unrolling (FGU) [17, 16] computes the meta gradient using the following forward recursive formula, starting from $\boldsymbol{Z}_0 = \frac{d\boldsymbol{\Omega}_0(\boldsymbol{\phi})}{d\boldsymbol{\phi}}$:

$$\underbrace{\frac{d\boldsymbol{\theta}_t(\boldsymbol{\phi})}{d\boldsymbol{\phi}}}_{\boldsymbol{Z}_t} = \underbrace{\frac{\partial \boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_{t-1}}}_{\boldsymbol{A}_t} \underbrace{\frac{d\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi})}{d\boldsymbol{\phi}}}_{\boldsymbol{Z}_{t-1}} + \underbrace{\frac{\partial \boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}}_{\boldsymbol{B}_t},\ t = 1, \ldots, T, \tag{4}$$

Reverse Gradient Unrolling (RGU) [46, 16], instead of the employment of explict reccursive formulas of $\boldsymbol{Z}_T$, focuses on the implicit reccursive formulas of $\nabla_{\boldsymbol{\phi}} h$:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}} h(\boldsymbol{\phi}) &= \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T}}_{\boldsymbol{d}_T} \underbrace{\frac{d\boldsymbol{\theta}_T(\boldsymbol{\phi})}{d\boldsymbol{\phi}}}_{\boldsymbol{Z}_T} + \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}}_{\boldsymbol{c}_T} \\ &= \boldsymbol{d}_T \boldsymbol{Z}_T + \boldsymbol{c}_T \overset{(4)}{=} \underbrace{\boldsymbol{d}_T \boldsymbol{A}_T}_{\boldsymbol{d}_{T-1}} \boldsymbol{Z}_{T-1} + \underbrace{\boldsymbol{d}_T \boldsymbol{B}_T + \boldsymbol{c}_T}_{\boldsymbol{c}_{T-1}} = \cdots = \boldsymbol{d}_0 \boldsymbol{Z}_0 + \boldsymbol{c}_0.\end{aligned} \tag{5}$$

The corresponding reverse recursive formulas can thus be summarized as

$$\boldsymbol{c}_{t-1} = \boldsymbol{c}_t + \boldsymbol{d}_t \boldsymbol{B}_t, \quad \boldsymbol{d}_{t-1} = \boldsymbol{d}_t \boldsymbol{A}_t, \quad t = T, \ldots, 1. \tag{6}$$

Weakness (GU): Non-Constant Memory. Both GU approaches exhibit a non-constant memory overhead, which constrains their utility in large-scale scenarios. The forward reccursive formulas in (4) revolve around the Jacobian matrix product, demanding $\mathcal{O}(MN)$ space consumption. The reverse recursive formulas in (6) necessitate the storage of the entire trajectory of the inner optimization $\boldsymbol{\theta}_{0:T}$ for backward computation, thereby imposing a memory requirement of $\mathcal{O}(TM)$. These requirements are often impractical for large-scale bi-level optimization, when $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are of high dimension and a significant unrolled depth is required.

Forward Gradient. Forward-mode automatic differentiation (forward-mode AD) has been applied to a variety of research fields, including the training of recurrent neural networks [70], the computation of Hessian vector products [50], etc. However, the computation of the true gradient via forward-mode AD requires the full Jacobian, which is typically too costly to compute.

To solve this, forward gradient learning [69, 4, 63, 4, 56], built upon forward-mode AD, was proposed. Forward gradient methods update parameters based on the directional gradient along a random perturbation direction for backpropagation-free training. More formally, given a differentiable function $h : \mathbb{R}^N \to \mathbb{R}$, the gradient for a given input $\boldsymbol{\phi} \in \mathbb{R}^N$ can be approximated as

$$\hat{\nabla} h(\boldsymbol{\phi}) = \nabla h(\boldsymbol{\phi}) \boldsymbol{v}\boldsymbol{v}^T, \tag{7}$$

where $\boldsymbol{v} \sim p(\boldsymbol{v})$ is a N-dimensional multivariate random variable, satisfying $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^T] = \mathbf{I}$. Common choices of the distribution of $\boldsymbol{v}$ include Rademacher $\boldsymbol{v} \sim \text{Unif}(\{-1, 1\}^N)$, Gaussian $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and uniform distribution over a set of normalized orthogonal coordinates $\boldsymbol{v} \sim \text{Unif}(\{\sqrt{N}\boldsymbol{e}_i\}_{1:N})$. For any given $\boldsymbol{\phi}$, $\hat{\nabla} h(\boldsymbol{\phi})$ is an unbiased estimator of $\nabla h(\boldsymbol{\phi})$, as $\mathbb{E}[\hat{\nabla} h(\boldsymbol{\phi})] = \mathbb{E}[\nabla h(\boldsymbol{\phi}) \boldsymbol{v}\boldsymbol{v}^T] = \nabla h(\boldsymbol{\phi}) \mathbb{E}[\boldsymbol{v}\boldsymbol{v}^T] = \nabla h(\boldsymbol{\phi}) \mathbf{I} = \nabla h(\boldsymbol{\phi})$. Despite the unbiasedness of $\hat{\nabla} h$, the dimension-dependent variance of the estimated gradient with a single direction impedes the scaling-up to high-dimensional problems. In practice, Monte Carlo gradient estimation can be used via averaged forward gradients over multiple random directions to reduce the variance.

3 (FG)^{2}U: Forward Gradient Unrolling with Forward Gradient

We aim to circumvent the memory overhead issues associated with forward gradient unrolling (FGU) as discussed in Section 2. We begin by examining the forward gradient of h at $\boldsymbol{\phi}$,

$$\hat{\nabla} h(\boldsymbol{\phi}) = \nabla h(\boldsymbol{\phi}) \boldsymbol{v}\boldsymbol{v}^T \overset{(5)}{=} (\boldsymbol{d}_T \boldsymbol{Z}_T \boldsymbol{v} + \boldsymbol{c}_T \boldsymbol{v}) \boldsymbol{v}^T, \tag{8}$$

where $\boldsymbol{v} \sim p(\boldsymbol{v})$ is a N-dimensional multivariate random variable, satisfying $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^T] = \mathbf{I}$. We follow the idea of FGU introduced in Section 2 to compute $\boldsymbol{Z}_T \boldsymbol{v}$. By multiplying both sides of (4) by $\boldsymbol{v}$ on the right, we can obtain the recursive formulas for $\boldsymbol{Z}_t \boldsymbol{v}$ as

$$\boldsymbol{Z}_0 \boldsymbol{v} = \boldsymbol{B}_0 \boldsymbol{v}; \quad \boldsymbol{Z}_t \boldsymbol{v} = \boldsymbol{A}_t \boldsymbol{Z}_{t-1} \boldsymbol{v} + \boldsymbol{B}_t \boldsymbol{v}, \;\; t = 1, \ldots, T. \tag{9}$$

The revised recursive formulas in (9) facilitate the tracking of a M-dimensional vector $\boldsymbol{Z}_t \boldsymbol{v}$, rather than full Jacobian $\boldsymbol{Z}_t$ of size $M \times N$, throughout the forward pass. The stochastic estimation in (8) is unbiased, adhering to the properties of forward gradient methods. To reduce the variance, we use

Monte Carlo estimate via averaged forward gradients over b *i.i.d.* random directions:

$$\hat{\nabla} h(\boldsymbol{\phi}) = \frac{1}{b}\sum_{i=1}^{b} \nabla h(\boldsymbol{\phi}) \boldsymbol{v}_i {\boldsymbol{v}_i}^T = \frac{1}{b}\sum_{i=1}^{b} (\boldsymbol{d}_T \boldsymbol{Z}_T \boldsymbol{v}_i + \boldsymbol{c}_T \boldsymbol{v}_i){\boldsymbol{v}_i}^T. \tag{10}$$

We call this algorithm **(FG)$^{\mathbf{2}}$U**, as an abbreviation of **F**orward **G**radient **U**nrolling with **F**orward **G**radient. The algorithm is summarized in Appendix A as Algorithm 1.

Compared to GU-based methods, as discussed in Section 2, (FG)^{2}U eliminates the dependency on the meta parameter dimension N and the depth of unrolling T without introducing bias, significantly enhancing memory efficiency. Unlike IF-based methods, as discussed in Appendix B.2, (FG)^{2}U overcomes the approximation issues associated with them while maintaining a constant memory overhead, thus providing superior gradient approximation. Compared to TRGU and Hessian-Free methods, which compromise approximation accuracy for efficiency, (FG)^{2}U consistently delivers accurate gradient approximations. The computational efficiency of (FG)^{2}U can be further enhanced by leveraging large-scale distributed computing resources, capitalizing on its inherently parallelizable formulation as presented in (10). In practice, a more cost-effective two-phase paradigm can be achieved by strategically placing (FG)^{2}U and other methods at different stages of the training process, as we will discuss in Section 3.2. For an illustration of the role of (FG)^{2}U in large-scale bi-level optimization, please refer to Figure 1.

3.1 Convergence

In this section, we provide a convergence analysis for (FG)^{2}U. The proofs can be found in Appendix C.

First, we establish a bound on the variance of the estimated gradient, when employing random vectors whose entries follow the Rademacher distribution.

Lemma 3.1. *For any $\boldsymbol{\phi} \in \Phi$, if $\boldsymbol{v}_i \sim \mathrm{Unif}(\{-1,1\}^N)$, the gradient estimation in* (10)*, satisfies*

$$\mathbb{E}\|\hat{\nabla} h(\boldsymbol{\phi}) - \nabla h(\boldsymbol{\phi})\|^2 = \frac{1}{\rho}\|\nabla h(\boldsymbol{\phi})\|^2,$$

where $\rho := \frac{b}{N-1} \in (0,1]$ as the sample size b is selected from $1, \cdots, N-1$.

The resultant error is bounded by $O\left(\frac{N-1}{b}\right)$, where b represents the sample size used for computing the forward gradient, and N is the dimensionality of the gradient itself. This bound demonstrates how the error scales inversely with the sample size while also being influenced by the gradient's dimensionality.

Next, we lay down the following assumptions, on which our main theorems are based. Let $\boldsymbol{\psi} = (\boldsymbol{\theta}, \boldsymbol{\phi}) \in \Theta \times \Phi$ denote the combination of the lower-level parameter $\boldsymbol{\theta}$ and the meta parameter $\boldsymbol{\phi}$. Following existing papers on the theory of bilevel optimization [45, 60, 28], in Assumption 3.2, we adopt some standard assumptions over the smoothness of the objective functions f and g.

Assumption 3.2. *The meta objective function $f(\boldsymbol{\psi})$ and the lower-level objective function $g(\boldsymbol{\psi})$ are both C-Lipschitz and L-smooth,* i.e.*, for any $\boldsymbol{\psi}, \boldsymbol{\psi}' \in \Theta \times \Phi$,*

$$|f(\boldsymbol{\psi}) - f(\boldsymbol{\psi}')| \le C\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|, \quad \|\nabla f(\boldsymbol{\psi}) - \nabla f(\boldsymbol{\psi}')\| \le L\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|, \tag{11}$$

$$|g(\boldsymbol{\psi}) - g(\boldsymbol{\psi}')| \le C\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|, \quad \|\nabla g(\boldsymbol{\psi}) - \nabla g(\boldsymbol{\psi}')\| \le L\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|. \tag{12}$$

The next assumption regulates that the transition functions $\boldsymbol{\Omega}$ satisfy similar smoothness conditions.

Assumption 3.3. *The transition functions $\boldsymbol{\Omega}_{0:T}$ are $C_{\boldsymbol{\Omega}}$-Lipschitz and $L_{\boldsymbol{\Omega}}$-smooth,* i.e.*, for any $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$,*

$$\|\boldsymbol{\Omega}_0(\boldsymbol{\phi}) - \boldsymbol{\Omega}_0(\boldsymbol{\phi}')\| \le C_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|, \ \|\nabla\boldsymbol{\Omega}_0(\boldsymbol{\phi}) - \nabla\boldsymbol{\Omega}_0(\boldsymbol{\phi}')\| \le L_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|. \tag{13}$$

For any $\boldsymbol{\psi}, \boldsymbol{\psi}' \in \Theta \times \Phi$, $t = 1, \ldots, T$,

$$\|\boldsymbol{\Omega}_t(\boldsymbol{\psi}) - \boldsymbol{\Omega}_t(\boldsymbol{\psi}')\| \le C_{\boldsymbol{\Omega}}\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|, \ \|\nabla\boldsymbol{\Omega}_t(\boldsymbol{\psi}) - \nabla\boldsymbol{\Omega}_t(\boldsymbol{\psi}')\| \le L_{\boldsymbol{\Omega}}\|\boldsymbol{\psi} - \boldsymbol{\psi}'\|. \tag{14}$$

Assumption 3.3 is made to ensure the generality of our analysis over different optimizers. Note that $\boldsymbol{\Omega}$ is scheme-dependent w.r.t. the gradient-based optimizer we adopt for lower-level problems. In many cases, such as gradient descent where $\boldsymbol{\Omega}_t(\boldsymbol{\psi}_{t-1}) = \boldsymbol{\theta}_{t-1} - \eta_t \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\psi}_{t-1})$, Assumption 3.3 is a direct consequence of Assumption 3.2.

We propose the following theorem and remark for convergence analysis of (FG)^{2}U on problem (2). Notice the convergence result can be extended to the primal BO problem (1) with some further assumptions. We place a proof scratch and some discussions in Appendix C.3.

Theorem 3.4 (Convergence)**.** *Suppose that Asumption 3.2 and Assumption 3.3 hold. Setting the learning rate $\beta = \frac{\rho}{(\rho+1)L_h}$ for gradient descent over the hyperparameter $\boldsymbol{\phi}$, then there exists a constant L_h (depending on C, L, C_Ω, L_Ω, and T, and defined formally in the proof) such that*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla h(\boldsymbol{\phi}_k)\|^2\right] \leq \frac{4L_h\left(\mathbb{E}[h(\boldsymbol{\phi}_0)] - \min_{\boldsymbol{\phi}} h(\boldsymbol{\phi})\right)}{\rho K}. \tag{15}$$

Remark 3.5. *Theorem 3.4 shows that Algorithm 1 converges to an ϵ-accurate stationary point with a convergence rate of $= \mathcal{O}(\epsilon^{-1}\rho^{-1})$.*

Recall that $\rho = \frac{b}{N-1}$, which indicates that the convergence rate is linearly dependent on N, which poses a significant challenge when managing high-dimensional meta-parameters $\boldsymbol{\phi}$. However, it is important to note that the dimension-dependent convergence rate represents an upper bound, and scalability has been found to be feasible with several practical considerations, as discussed in the following subsection.

3.2 Practical Considerations

Choice of b. According to the convergence analysis in Section 3.1, a sample size of $b = \mathcal{O}(N)$ is required to achieve a convergence rate of $\mathcal{O}(\epsilon^{-1})$. However, it has been widely observed that forward gradient and zeroth-order optimization, despite having dimension-dependent convergence rates, work well empirically with $b = \mathcal{O}(1)$ in large-scale scenarios, such as in LLM fine-tuning [47, 74]. In this paper, we select the largest possible b that does not exceed the GPU memory limit for our empirical study. Additionally, gradient accumulation is utilized to further control variance and stabilize the training process.

Cost-Effective Two-Phase Paradigm. It is important to note that the upper bound delineated in (15) linearly depends on the performance discrepancy between the initialized meta parameter $\boldsymbol{\phi}_0$ and the optimal. This dependence motivates the adoption of a more cost-effective two-phase paradigm for large-scale bi-level optimization. In the initial phase, we utilize efficient yet less accurate gradient approximation methods, such as TRGU [60] and Hessian-Free [9], to efficiently establish an initial $\boldsymbol{\phi}_0$ that surpasses random initialization, while keeping computational overhead manageable. Subsequently, in the second phase, (FG)^{2}U is utilized for a more accurate, albeit less efficient, gradient approximation to further elevate the performance, leveraging extensive computational resources.

Implementation. The technique employed in computing $\nabla h(\boldsymbol{\phi})\boldsymbol{v}$ is identified as forward-mode automatic differentiation (forward-mode AD). In advanced automatic differentiation libraries, such as JAX [5] and PyTorch [3], forward-mode AD is efficiently implemented as Jacobian-vector product (`jvp`), without the necessity of explicitly computing the Jacobian matrix. The FLOP cost of `jvp` is approximately three times that of a standard forward pass, while the memory overhead is doubled. In practice, it is only necessary to define the forward computational graph of inner optimization and invoke forward-mode AD, which simplifies the implementation process significantly. Regarding distributed training, JAX offers the `vmap` interface for efficient intra-GPU parallelism and the `pmap` interface for effective inter-GPU parallelism.

Zeroth-order Bi-level optimization. In certain applications of bi-level optimization, the inner problem is approached as a black box, where the gradient of $\boldsymbol{\Omega}$ is inaccessible, rendering the analytical gradient unrolling unfeasible. For example, in PDE-constrained optimization [23, 62], in which the inner problem entails solving a Partial Differential Equation (PDE) using a non-differentiable solver. In such scenarios, rather than employing forward-mode Automatic Differentiation (AD), one can resort to Finite Difference methods to approximate the directional gradient $\nabla h(\boldsymbol{\phi})\boldsymbol{v}$ by

$$\nabla h(\boldsymbol{\phi})\boldsymbol{v} = \lim_{\mu\to 0}\frac{h(\boldsymbol{\phi}+\mu\boldsymbol{v}) - h(\boldsymbol{\phi})}{\mu} \approx \frac{h(\boldsymbol{\phi}+\bar{\mu}\boldsymbol{v}) - h(\boldsymbol{\phi})}{\bar{\mu}} \tag{16}$$

with sufficiently small positive $\bar{\mu} > 0$. We refer to this zeroth-order variant of (FG)^{2}U as (FG)^{2}U-ZO, noting that the computation solely encompasses two forward passes and does not involve the utilization of any first-order information. The memory complexity is the same as forward-mode AD and the actual computation time will be slightly less than forward-mode AD, at the cost of introducing an approximation bias. We give a more detailed discussion within the context of *zeroth-order optimization* [41] in Appendix D, and empirically study a corresponding case in Section 4.

4 Experiments

We conduct experiments across various contexts, as detailed in the respective subsections. Initially, we engage in an image data condensation task, where we focus on a comprehensive performance comparison between (FG)^{2}U and both classical and large-scale bi-level optimization algorithms. Subsequently, we investigate meta-learning for the online adaptation of language models, employing a GPT model as the inner model, to illustrate how (FG)^{2}U effectively circumvents the non-constant memory issue associated with RGU. Finally, we address a physics-informed bi-level optimization problem, where gradient-based inner solvers are ineffective, to demonstrate the efficacy of combining (FG)^{2}U-ZO, the zeroth-order variant of (FG)^{2}U discussed in Section 3.2, with non-differentiable numerical solvers.

Data Condensation. To overcome the challenges posed by large-scale datasets, a line of works known as data condensation [68, 72] has been proposed. The main idea is to generate a compact, synthesized dataset, designed to elicit similar behaviors in machine learning models as those trained with the original, massive dataset. The objective of the mainstream principles [72] designed for data condensation can be naturally formulated as a bi-level optimization problem. We focus on the best-known principle *performance matching* [72] on classification tasks, which can be formulated as

$$\min_{\mathcal{D}_c} \mathcal{L}(\theta_T; \mathcal{D}_o), \quad \text{where } \theta_t = \theta_{t-1} - \eta \nabla \mathcal{L}(\theta_{t-1}; \mathcal{D}_c), \;\; t = 1, \ldots, T, \tag{17}$$

where $\mathcal{D}_o$, $\mathcal{D}_c$ respectively denote the original and condensed dataset, θ denotes the model parameter, $\mathcal{L}$ denotes the cross-entropy loss function, and η represents the step-size for inner optimization.

Dataset	IPC	Ratio (%)	Approaches				For Reference	
			TRGU	Hessian-Free	Neumann	**(FG)^{2}U**	RGU	WHOLE
MNIST	1	0.017	73.76±1.68	65.98±1.38	68.37±1.44	**82.44**±0.68	92.32±0.33	
	10	0.17	94.05±0.33	94.97±0.34	95.75±0.24	**96.12**±0.28	96.79±0.29	99.6±0.00
	50	0.83	96.63±0.41	96.34±0.31	96.78±0.22	**97.01**±0.19	97.72±0.23	
CIFAR-10	1	0.02	20.78±1.07	19.72±1.28	21.33±0.90	**29.37**±0.75	34.08±0.55	
	10	0.2	44.01±0.57	45.32±1.02	47.67±0.87	**50.10**±0.56	53.15±0.53	84.8±0.10
	50	1	49.22±0.45	48.73±0.78	50.02±0.69	**51.98**±0.44	56.37±0.37	
CIFAR-100	1	0.2	3.96±0.68	3.14±0.41	4.52±0.56	**8.22**±0.45	15.61±0.32	
	10	2	20.20±0.66	19.01±0.84	20.87±0.82	**23.38**±0.33	25.42±0.45	56.2±0.30
	50	10	22.33±0.93	23.59±0.71	24.52±0.77	**25.84**±0.31	28.52±0.53	

Table 1: The performance (testing accuracy %) comparison among various bilevel optimization methods on the data condensation task over three datasets. All the datasets are condensed using a 3-layer ConvNet. IPC: image(s) per class. Ratio (%): the ratio of condensed examples to the whole training set.

We conducted our experiments following the standard data condensation setting established by [68, 77, 67]. A more detailed task description is given in Appendix E.1 and implementation details are given in Appendix F.1.

The condensed datasets are evaluated using 3-layer convolutional networks with randomly initialized parameters, and the average accuracies on test datasets are summarized in Table 1. Compared to large-scale bi-level optimization methods like TRGU and Hessian-Free, which prioritize efficiency at the expense of approximation accuracy, (FG)^{2}U exhibits significantly better performance, due to more accurate gradient approximation as explained in Appendix B. Additionally, we assessed Neumann Series (denoted as Neumann in Table 1), an IF-based method that mitigates gradient approximation errors through extended computations, as introduced in Appendix B.2. While it demonstrates performance enhancements over the Hessian-Free method, Neumann still yields suboptimal performance compared to (FG)^{2}U, owing to the inherent bias of the IF-based method. Further discussions and supporting evidence are available in Appendix B.2.

The results of RGU, which represent the upper performance bound for both TRGU and (FG)^{2}U, are provided for reference, along with the results from training on the entire dataset (denoted as WHOLE in Table 1), representing the upper performance bound for all approaches. However, it is crucial to acknowledge that RGU is not practical in large-scale bi-level optimization scenarios due to its non-constant memory requirements, as discussed in Section 2. This limitation will be further exemplified in the subsequent, where the inner model is significantly larger. In principle, the performance of (FG)^{2}U can be further improved to approach that of RGU by increasing the number of random directions for gradient approximation.

The memory and computational efficiencies of TRGU, Hessian-Free, and (FG)^{2}U in the most challenging case (CIFAR-100, IPC=50) are reported in Figure 1 (Bottom Right), demonstrating that the efficiency of (FG)^{2}U can be significantly enhanced through intra/inter-GPU parallelism.

Meta Learning Online Adaptation of Language Models. The online adaptation of language models (LM) has been studied recently to keep the knowledge of LM current [34, 27]. However, trivial auto-regressive fine-tuning the LM, which applies uniform weights to all tokens, often results in suboptimal performance in downstream tasks. This issue stems from the default average negative log-likelihood (NLL) loss, which fails to capture the significance of tokens [25]. To overcome this limitation, [25] proposed Context-aware Meta-learned Loss Scaling (CaMeLS), a strategy that employs meta-learning to adjust token weights for more effective online adaptation. Specifically, they meta train a weight model to reweight the auto-regressive loss during online fine-tuning, aiming to enhance LM performance on downstream question-answering tasks. A comprehensive task description and the mathematical formulation of the objectives are detailed in Appendix E.2.

The trained weight model is subsequently fine-tuned on unseen online documents and evaluated on corresponding question-answering tasks. In [25], RGU is utilized for meta gradient approximation. To mitigate the non-constant memory issue associated with RGU, a DistilGPT2 model [59] is chosen as the surrogate base model for training the weight model, instead of larger models typically employed for online adaptation. Additionally, a very limited unrolled depth of 6 is utilized within a 40 GiB GPU memory budget. In our experiments, since (FG)^{2}U has circumvented the non-constant memory issue associated with RGU, we are able to increase the unrolled depth and upscale the base model for training the weight model. Empirical evaluations are conducted on two datasets, StreamingQA [36] and SQuAD-Seq [54].

Model (# params)	**Method**	**StreamingQA** EM (↑)	**StreamingQA** F1 (↑)	**SQuAD-Seq** EM (↑)	**SQuAD-Seq** F1 (↑)
DistilGPT2 (82M)	CaMeLS + RGU [25, 66]	1.62	5.79	1.45	3.08
	CaMeLS + RGU (impl.)	2.04	5.53	1.52	3.16
	CaMeLS + **(FG)^{2}U** (ours)	**2.22**	**6.37**	**1.72**	**3.50**
GPT2-Large (774M)	CaMeLS + RGU [25, 66]	5.35	10.60	4.97	8.63
	CaMeLS + RGU (impl.)	7.02	12.19	4.86	8.57
	CaMeLS + **(FG)^{2}U** (ours)	**7.21**	**12.50**	**5.56**	**8.99**
GPT2-XL (1.5B)	CaMeLS + RGU [25, 66]	6.55	11.67	6.70	10.15
	CaMeLS + RGU (impl.)	7.93	12.94	6.71	9.65
	CaMeLS + **(FG)^{2}U** (ours)	**8.89**	**14.42**	**7.37**	**10.37**

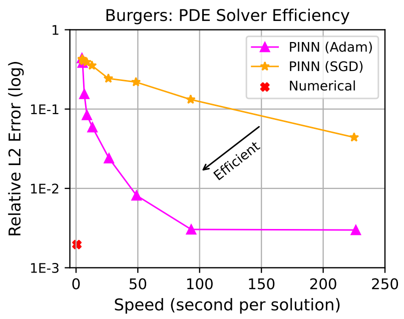
Table 2: Comparison of the online adaptation performance. The reported evaluation metrics include the exact match (EM) and F1 scores. For vanilla CaMeLS [25], RGU is conducted with unrolled depth 6, using DistilGPT2 as the base model. We present both the results reported by [66] and those from our implementation (denoted as impl.). For CaMeLS + (FG)^{2}U, we select unrolled depths from $\{24, 48\}$, and the base model from {DistilGPT2, GPT2}. We report the results for the combination that yields the best F1 score. Additional details and ablation studies are documented in Appendix G.1.

Firstly, we increased the unrolled depth while maintaining the base model as a DistilGPT2. We plotted the F1 scores and GPU memory usages for RGU with unrolled depths of $\{1, 2, 4, 6\}$ and (FG)^{2}U with unrolled depths of $\{24, 48\}$ on StreamingQA in Figure 1 (Bottom Left). The performance of the weight model is positively correlated with the unrolled depth, substantiating the benefits of training with larger unrolled depths. The non-constant memory issue associated with RGU can be observed when the unrolled depth increases, while (FG)^{2}U maintains constant memory even with large unrolled depth. Subsequently, we endeavored to upscale the base model to GPT2 to reduce the disparity between training and evaluation. The performances are summarized in Table 2, with detailed ablation studies on unrolled depths and base model variants documented in Table G.1 and Table G.2.

Data-driven Discovery of Partial Differential Equations (PDEs). Let us consider the following general forms of parametrized and nonlinear PDEs:

$$u_t + \mathcal{N}[u; \boldsymbol{\phi}] = 0,\ x \in \Psi, t \in [0, T], \tag{18}$$

where x denotes the space-time coordinate, Ψ denotes a bounded domain with boundary, $u : [0, T] \times \Psi \to \mathbb{R}$ denotes the latent solution, u_t represents the first-order derivative of u with respect to t, and $\mathcal{N}$ is a general differential operator parameterized by $\boldsymbol{\phi}$, acting on Ψ. This setup encompasses a broad spectrum of problems in physics. For example, the one-dimensional Burgers' equation is defined by $\mathcal{N}[u; \boldsymbol{\phi}] = \mu u u_x - \nu u_{xx}$, where $\boldsymbol{\phi} = (\mu, \nu) \in \mathbb{R}^2$, and u_x, u_{xx} represent the first and second-order derivatives of u with respect to x, respectively.



Method		(FG)^{2}U	(FG)^{2}U-ZO
Inner solver		PINN [52]	Numerical
Burgers	$\boldsymbol{\epsilon_\phi}$(E-2, ↓)	2143.58$\pm$855.26	**0.97**$\pm$0.45
	$\boldsymbol{\epsilon}_u$(E-3, ↓)	336.06$\pm$46.91	**0.63**$\pm$0.33
Allen-Cahn	$\boldsymbol{\epsilon_\phi}$(E-2, ↓)	438.13$\pm$101.77	**2.34**$\pm$0.64
	$\boldsymbol{\epsilon}_u$(E-3, ↓)	133.61$\pm$35.93	**0.97**$\pm$0.54
KdV	$\boldsymbol{\epsilon_\phi}$(E-2, ↓)	94.40$\pm$4.31	**0.72**$\pm$0.57
	$\boldsymbol{\epsilon}_u$(E-3, ↓)	832.81$\pm$67.01	**2.72**$\pm$1.55

Figure 2: **Left**: Comparison of efficiency between the PINN solver and the numerical solver. We evaluated Adam [29] and SGD as the inner optimizers for the PINN solver, with steps ranging from 100 to 50,000. The results demonstrate that the numerical solver is significantly more efficient. **Right**: Comparison of relative L2 errors in the prediction of $\boldsymbol{\phi}$ and u. $\boldsymbol{\epsilon_\phi} = \|\boldsymbol{\phi}_{pred} - \boldsymbol{\phi}\|_2 / \|\boldsymbol{\phi}\|_2$, $\boldsymbol{\epsilon}_u = \|u_{pred} - u\|_2 / \|u\|_2$.

The problem of data-driven discovery of PDEs [52] can be framed as follows: given a set of scattered observations of the latent solution $u(x)$, what are the parameters most accurately describing the observed data? The problem can be formulated as a PDE-constrained optimization problem (PDECO):

$$\min_{\boldsymbol{\phi}}\ \mathbb{E}_{x,u\sim\mathcal{D}}\ |u(x; \boldsymbol{\phi}) - u|^2 \quad s.t. \quad u_t + \mathcal{N}[u(\cdot; \boldsymbol{\phi}); \boldsymbol{\phi}] = 0, x \in \Psi, \tag{19}$$

where $\mathcal{D} = \{(x_i, u_i)\}_{1:k}$ denotes the observed data. In cases where the closed-form solutions of the nonlinear PDEs are intractable, parametric solutions $u_{\boldsymbol{\theta}}$ are used to approximate the latent solution u for given $\boldsymbol{\phi}$. The PDECO in (19) is then reformulated into a bi-level optimization problem:

$$\min_{\boldsymbol{\phi}}\ \mathbb{E}_{x,u\sim\mathcal{D}}\ |u_{\boldsymbol{\theta}_S(\boldsymbol{\phi})}(x; \boldsymbol{\phi}) - u|^2 \quad s.t. \quad \boldsymbol{\theta}_s(\boldsymbol{\phi}) = \boldsymbol{\Omega}_s(\boldsymbol{\theta}_{s-1}, \boldsymbol{\phi}), s = 1, \ldots, S. \tag{20}$$

Employing gradient-based PDE solvers, such as physics-informed neural networks (PINN) [52], facilitates the direct application of (FG)^{2}U. However, as demonstrated in Figure 2 (Left), the accuracy and efficiency of PINNs fall short of the rigorous demands of scientific computing. This limitation has prompted us to integrate faster and more accurate traditional solvers like the spectral method [1] (see also Appendix E.3.4) to tackle the inner problem. Given these solvers are non-differentiable, we employ (FG)^{2}U-ZO, the zeroth-order variant of (FG)^{2}U introduced in Section 3.2, to solve the problem.

We conduct experiments on three non-linear PDEs: Burgers, Allen-Cahn, and KdV, with a more detailed task description available in Appendix E.3. The results are summarized in Figure 2 (Right). We can observe that the combination of (FG)^{2}U-ZO and the numerical solver significantly outperforms (FG)^{2}U and the PINN solver, in terms of both the prediction on $\boldsymbol{\phi}$ and u. The implementation details are documented in Appendix F.3.

5 Conclusion

In this work, we propose a novel algorithm **F**orward **G**radient **U**nrolling with **F**orward **G**radient, abbreviated as **(FG)^{2}U**, designed to tackle the challenges associated with large-scale bi-level optimization. We conduct a convergence analysis of (FG)^{2}U, perform extensive comparisons with existing methods, and provide detailed discussions on its practical applications. Additionally, we undertake an empirical evaluation across a series of large-scale bi-level optimization tasks. Our findings indicate that (FG)^{2}U effectively complements existing bi-level optimization algorithms, addressing gaps in large-scale bi-level optimization scenarios.

Limitations and future works. The experiments conducted in this paper are of relatively small scale, with the largest inner model being a GPT-2 model. We look forward to validating its effectiveness on larger-scale bi-level optimization tasks. Additionally, the application of black-box bi-level optimization and the potential of (FG)^{2}U-ZO remain underexplored, considering the prevalent black-box interaction between users and models today. We hope our work will inspire further development of large-scale bi-level optimization algorithms and their application in corresponding scenarios. Furthermore, we have not specifically addressed the efficiency issues inherited by (FG)^{2}U from the forward gradient method. Enhancing the efficiency of (FG)^{2}U while maintaining its gradient estimation accuracy will be an important direction for future research.

6 Acknowledgements

This research is supported by the National Research Foundation Singapore under the AI Singapore Programme (AISG Award No: AISG2-TC-2023-010-SGIL) and the Singapore Ministry of Education Academic Research Fund Tier 1 (Award No: T1 251RES2207, T1 251RES2218). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

[1] William F Ames. *Numerical methods for partial differential equations*. Academic press, 2014.

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[3] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

[4] Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

[5] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, and Thomas A Zang. *Spectral methods: fundamentals in single domains*. Springer Science & Business Media, 2007.

[8] Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal fully first-order algorithms for finding stationary points in bilevel optimization. *arXiv preprint arXiv:2306.14853*, 2023.

[9] Sang Choe, Sanket Vaibhav Mehta, Hwijeen Ahn, Willie Neiswanger, Pengtao Xie, Emma Strubell, and Eric Xing. Making scalable meta learning practical. *Advances in neural information processing systems*, 36, 2024.

[10] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022.

[13] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[16] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pages 1165–1173. PMLR, 2017.

[17] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pages 1568–1577. PMLR, 2018.

[18] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[19] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[20] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[21] David Gottlieb and Steven A Orszag. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

[22] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

[23] Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, Jian Song, and Ze Cheng. Bi-level physics-informed neural networks for pde constrained optimization using broyden's hypergradients. *arXiv preprint arXiv:2209.07075*, 2022.

[24] Ryuichiro Hataya and Makoto Yamada. Nyström method for accurate and scalable implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 4643–4654. PMLR, 2023.

[25] Nathan Hu, Eric Mitchell, Christopher D Manning, and Chelsea Finn. Meta-learning online adaptation of language models. *arXiv preprint arXiv:2305.15076*, 2023.

[26] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems*, 33:12080–12091, 2020.

[27] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. *arXiv preprint arXiv:2110.03215*, 2021.

[28] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *International conference on machine learning*, pages 4882–4892. PMLR, 2021.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Steven George Krantz and Harold R Parks. *The implicit function theorem: History, theory, and applications*. Springer Science & Business Media, 2002.

[31] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[32] Harshat Kumar, Dionysios S Kalogerias, George J Pappas, and Alejandro Ribeiro. Zeroth-order deterministic policy gradient. *arXiv preprint arXiv:2006.07314*, 2020.

[33] Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D Nowak. A fully first-order method for stochastic bilevel optimization. In *International Conference on Machine Learning*, pages 18083–18113. PMLR, 2023.

[34] Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

[35] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[36] Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, D'Autume Cyprien De Masson, Tim Scholtes, Manzil Zaheer, Susannah Young, et al. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. In *International Conference on Machine Learning*, pages 13604–13622. PMLR, 2022.

[37] Bo Liu, Mao Ye, Stephen Wright, Peter Stone, and Qiang Liu. Bome! bilevel optimization made easy: A simple first-order approach. *Advances in neural information processing systems*, 35:17248–17262, 2022.

[38] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[39] Risheng Liu, Xuan Liu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A value-function-based interior-point method for non-convex bi-level optimization. In *International conference on machine learning*, pages 6882–6892. PMLR, 2021.

[40] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *International conference on machine learning*, pages 6305–6315. PMLR, 2020.

[41] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O Hero III, and Pramod K Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

[42] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[43] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pages 1540–1552. PMLR, 2020.

[44] Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[45] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pages 2952–2960. PMLR, 2016.

[46] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[47] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[48] John L Nazareth. Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):348–353, 2009.

[49] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[50] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160, 1994.

[51] George F Pinder. *Numerical methods for solving partial differential equations: a comprehensive introduction for scientists and engineers*. John Wiley & Sons, 2018.

[52] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[53] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

[54] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[55] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[56] Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

[57] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

[58] Levent Sagun, Utku Evci, V. Ugur Güney, Yann N. Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*, 2018.

[59] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[60] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *The International Conference on Artificial Intelligence and Statistics*, pages 1723–1732. PMLR, 2019.

[61] Han Shen and Tianyi Chen. On penalty-based bilevel gradient descent method. In *International Conference on Machine Learning*, pages 30992–31015. PMLR, 2023.

[62] Qianli Shen, Wai Hoh Tang, Zhun Deng, Apostolos Psaros, and Kenji Kawaguchi. Picprop: Physics-informed confidence propagation for uncertainty quantification. *Advances in Neural Information Processing Systems*, 36, 2024.

[63] David Silver, Anirudh Goyal, Ivo Danihelka, Matteo Hessel, and Hado van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

[64] Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression. *Advances in Neural Information Processing Systems*, 33:18098–18109, 2020.

[65] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[66] Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, and Jonathan Richard Schwarz. Online adaptation of language models with a memory of amortized contexts. *arXiv preprint arXiv:2403.04317*, 2024.

[67] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12196–12205, 2022.

[68] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

[69] R. E. Wengert. A simple automatic derivative evaluation program. *Commun. ACM*, 7(8):463–464, 1964.

[70] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.*, 1(2):270–280, 1989.

[71] Peng Yang, Yingjie Lao, and Ping Li. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14841–14850, 2021.

[72] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[73] Yihua Zhang, Prashant Khanduri, Ioannis Tsaknakis, Yuguang Yao, Mingyi Hong, and Sijia Liu. An introduction to bi-level optimization: Foundations and applications in signal processing and machine learning. *arXiv preprint arXiv:2308.00788*, 2023.

[74] Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. *arXiv preprint arXiv:2402.11592*, 2024.

[75] Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. Advancing model pruning via bi-level optimization. *Advances in Neural Information Processing Systems*, 35:18309–18326, 2022.

[76] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *International Conference on Machine Learning*, pages 26693–26712. PMLR, 2022.

[77] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

[78] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[79] Simiao Zuo, Chen Liang, Haoming Jiang, Xiaodong Liu, Pengcheng He, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. Adversarial regularization as stackelberg game: An unrolled optimization approach. *arXiv preprint arXiv:2104.04886*, 2021.

A Algorithm

Algorithm 1 (FG)^{2}U: Forward Gradient Unrolling with Forward Gradient

```
Require: Initial inner parameters $\boldsymbol{\theta}_0$, initial meta parameter $\boldsymbol{\phi}_0$, random direction distribution $\boldsymbol{p}$,
    number of random directions $b$, total meta steps $K$, meta update mappings $\boldsymbol{\Psi}_{1:K}$.
 1: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$, $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}_0$
 2: for $k = 1, \ldots, K$ do
 3:     for $i = 1, \ldots, b$ do
 4:         Sample $\boldsymbol{v}_i \sim \boldsymbol{p}(\cdot)$ and initialize $\boldsymbol{y}_i \leftarrow \frac{\partial \boldsymbol{\Omega}_0(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \boldsymbol{v}_i$
 5:     end for
 6:     for $t = 1, \ldots, T$ do
 7:         $\boldsymbol{\theta} \leftarrow \boldsymbol{\Omega}_t(\boldsymbol{\theta}, \boldsymbol{\phi})$, $\boldsymbol{A} \leftarrow \frac{\partial \boldsymbol{\Omega}_t(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\theta}}$, $\boldsymbol{B} \leftarrow \frac{\partial \boldsymbol{\Omega}_t(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}$
 8:         for $i = 1, \ldots, b$ do
 9:             $\boldsymbol{y}_i \leftarrow \boldsymbol{A}\boldsymbol{y}_i + \boldsymbol{B}\boldsymbol{v}_i$
10:         end for
11:     end for
12:     for $i = 1, \ldots, b$ do
13:         $w_i \leftarrow \frac{\partial f(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\theta}} \boldsymbol{y}_i + \frac{\partial f(\boldsymbol{\theta},\boldsymbol{\phi})}{\partial \boldsymbol{\phi}} \boldsymbol{v}_i$
14:     end for
15:     $\boldsymbol{\phi} \leftarrow \boldsymbol{\Psi}_k(\boldsymbol{\phi}, \frac{1}{b}\sum_{i=1}^{b} w_i \boldsymbol{v}_i^T)$
16: end for
17: return $\boldsymbol{\phi}$
```

B Extended Discussion on Bi-level Optimization

B.1 Truncated Reverse Gradient Unrolling (TRGU)

To address the memory issue of GU methods, truncated Reverse Gradient Unrolling (TRGU) [60] is proposed to reduce the memory usage by preserving only the last K steps of the inner optimization trajectory. However, this introduces a significant bias in large-scale scenarios, particularly when the permissible K is small.

Recall (5) and (6), where the conventional RGU method computes the hypergradient by fully unrolling the T-step inner optimization into a computational graph. Instead, TRGU performs s-step truncated back-propagation and approximates the gradient with the intermediate term $\boldsymbol{c}_{T-s}$:

$$\boldsymbol{c}_{T-s} = \boldsymbol{c}_T + \sum_{t=T-s+1}^{T} \boldsymbol{B}_t \boldsymbol{A}_{t+1} \cdots \boldsymbol{A}_T d_T. \tag{21}$$

According to Proposition 3.1 in [60], if the inner-level objective function g is L-smooth, twice-differentiable and globally α-strongly convex, and the gradient update rule writes $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi})$, then the bias of s-step TRGU would be bounded by

$$\|\nabla_{\boldsymbol{\phi}} h - \boldsymbol{c}_{T-s}\| \leq \frac{(1-\eta\alpha)^s}{\eta\alpha} \|\boldsymbol{d}_T\| \max_{t \in 0,\ldots,T-s} \|\boldsymbol{B}_t\|. \tag{22}$$

The bound (22) demonstrates an exponentially decaying rate in s over the bias of s-step TRGU. However, when s gets smaller, which means that we truncate the computational graph heavier in pursuit of lower memory cost, the bias would grow exponentially. This would result in an inaccurate calculation of the hypergradient. Contrastively, our **(FG)^{2}U** is an unbiased estimator of the hypergradient, while still keeping high memory efficiency with a small sample size of forward gradient as in (10).

B.2 Implicit Function (IF)

Another idea for computing the implicit gradient is to utilize the implicit function theorem (IFT) [30]. Suppose that the inner optimality $\nabla_{\boldsymbol{\theta}} g(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi}) \approx 0$ is approximately achieved by sufficient inner optimization steps. If g is second-order differentiable, by applying the implicit function theorem and taking the first-order derivative of $\boldsymbol{\phi}$,

$$\frac{\partial^2 g(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T^2} \frac{d\boldsymbol{\theta}_T(\boldsymbol{\phi})}{d\boldsymbol{\phi}} + \frac{\partial^2 g(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T \partial \boldsymbol{\phi}} \approx 0. \tag{23}$$

Then, if the Hessian is further assumed to be invertible, the meta gradient can be approximated as

$$\nabla h(\boldsymbol{\phi}) \approx -\underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T}}_{\boldsymbol{d}} \underbrace{\left(\frac{\partial^2 g(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T^2}\right)^{-1}}_{\boldsymbol{H}^{-1}} \underbrace{\frac{\partial^2 g(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T \partial \boldsymbol{\phi}}}_{\boldsymbol{Y}} + \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}}_{\boldsymbol{c}}. \tag{24}$$

The main challenge lies in the computation of the inverse Hessian matrix $\boldsymbol{H}^{-1}$, which is intractable when $\boldsymbol{\theta}$ is of high dimensionality. Fortunately, several iterative inverse Hessian vector product (`ihvp`) approximators requiring only Hessian vector product (`hvp`) and $\mathcal{O}(M)$ space can be employed to produce $\widehat{\boldsymbol{d}\boldsymbol{H}^{-1}}$ for approximating $\boldsymbol{d}\boldsymbol{H}^{-1}$, based on Conjugate Gradient [48, 53], Neumann Series [19, 28] and low-rank approximation [64, 24].

Neumann Series. The inverse Hessian vector product can be approximated with a truncated sum of Neumann series [19, 28],

$$\widehat{\boldsymbol{d}\boldsymbol{H}^{-1}} = \alpha \sum_{k=0}^{K} \boldsymbol{d}(\boldsymbol{I} - \alpha \boldsymbol{H})^k = \boldsymbol{d}\boldsymbol{H}^{-1} - \alpha \sum_{k=K+1}^{\infty} \boldsymbol{d}(\boldsymbol{I} - \alpha \boldsymbol{H})^k, \tag{25}$$

where α is a hyperparameter to ensure the convergence, and K is the number of truncated steps. Compared to other IF-based methods, the Neumann Series has demonstrated good empirical performance and stability [19], and its stochastic variant has been well studied [28].

Weakness (IF): Approximation Errors. The errors of IF emanate from two distinct sources Firstly, IFT presupposes that the Karush-Kuhn-Tucker (KKT) conditions of the inner problem are satisfied, leading to an approximation error in (23) when iterative approximations of the inner solutions are used. Secondly, the singular nature of the Hessian within neural network training [57] leads to costly and unstable inverse Hessian approximation in practical applications, with a heavy reliance on engineering efforts [9]. More formally, recall

$$\nabla h(\boldsymbol{\phi}) = \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\theta}_T}}_{\boldsymbol{d}} \underbrace{\frac{d\boldsymbol{\theta}_T(\boldsymbol{\phi})}{d\boldsymbol{\phi}}}_{\boldsymbol{Z}} + \underbrace{\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}}}_{\boldsymbol{c}}. \tag{26}$$

The approximation error can be decomposed into

$$\underbrace{\nabla h(\boldsymbol{\phi}) - \hat{\nabla} h(\boldsymbol{\phi})}_{\boldsymbol{\epsilon}} = \underbrace{\boldsymbol{d}(\boldsymbol{Z} + \boldsymbol{H}^{-1}\boldsymbol{Y})}_{\boldsymbol{\epsilon}_{\text{if}}} + \underbrace{(\widehat{\boldsymbol{d}\boldsymbol{H}^{-1}} - \boldsymbol{d}\boldsymbol{H}^{-1})\boldsymbol{Y}}_{\boldsymbol{\epsilon}_{\text{inv}}}. \tag{27}$$

To reduce the computational cost, Hessian-free approaches [76, 75, 9] propose approximating the Hessian as an identity matrix, incorporating additional assumptions about the inner model and objective.

$$\widehat{\boldsymbol{d}\boldsymbol{H}^{-1}} = \alpha \boldsymbol{d}\boldsymbol{I}, \tag{28}$$

where $\alpha > 0$ is a hyperparameter to control the magnitude. However, these numerous assumptions often diverge from practical scenarios, resulting in significant approximation errors and consequently inducing suboptimal outcomes.

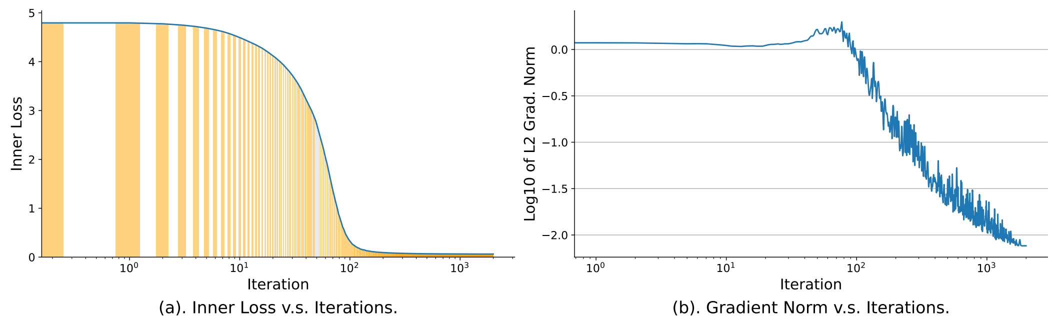


Figure B.1: **CIFAR100, IPC=50**: Inner Loss and gradient norm for Neumann

In Figure B.1, it is evident that the inner optimization has not converged by the unrolled step 100, as indicated by both inner loss and gradient norm. This observation implies that the Karush-Kuhn-Tucker (KKT) conditions are not satisfied, leading to the conclusion that the approximation used in (23) introduces a bias.

B.3 Value Function (VF)

The VF-based methodology [39, 37, 61, 33] considers an equivalent reformulation of the original optimization problem as outlined in (2):

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} f(\boldsymbol{\theta},\boldsymbol{\phi}) \quad s.t.\ g(\boldsymbol{\theta},\boldsymbol{\phi}) \le g(\boldsymbol{\theta}_T(\boldsymbol{\phi}),\boldsymbol{\phi}). \tag{29}$$

This reformulation casts the standard bi-level optimization challenge into a constrained single-level optimization framework. VF-based methods circumvent the need for second-order computations and have demonstrated near-optimal complexity, comparable to second-order methodologies in deterministic settings, as reported in [8].

Weakness (VF): stochastic optimization. VF-based strategies have yet to gain widespread acceptance in practical ML applications. This limited adoption is primarily attributed to the challenges these methods face in addressing large-scale stochastic problems, where the complexity significantly impedes their performance [73].

C Proofs of Theoretical Results

In this section, we detail the proofs of Lemma 3.1 and Theorem 3.4. Our approach to proving Theorem 3.4 follows a similar high-level approach as [19, 28, 60] with some important distinctions. Initially, in Lemma B.2, we extend the smoothness properties of the objective functions f and g, and the transition functions Ω, to the T-th iteration lower-level parameter $\boldsymbol{\theta}_T$. Following this, Lemma B.3 establishes the smoothness of the meta-learning objective $f(\boldsymbol{\theta}_T,\boldsymbol{\phi})$, incorporating results from the inner-loop computations. Building on the demonstrated smoothness of the meta objective function and the variance of the forward gradient method (as shown in Lemma 3.1), we then validate the convergence properties of Algorithm 1.

The novelty in our proof of Theorem 3.4 lies in two primary aspects. Firstly, our analysis does not presume that the lower-level optimization yields an optimal solution $\boldsymbol{\theta}^*$; instead, it more realistically assumes the use of $\boldsymbol{\theta}_T$, which is derived from a finite number of iterations. This assumption aligns more closely with the computational constraints encountered in real-world scenarios. Secondly, our convergence analysis explicitly accounts for the variance of our unbiased gradient estimator, achieving a convergence rate of $\mathcal{O}(\epsilon^{-1}\rho^{-1})$. This demonstrates that utilizing the forward gradient method, while significantly reducing memory requirements, does not adversely affect the algorithm's convergence rate, underscoring the practical viability and efficiency of our approach even with memory constraints.

In Appendix C.3, we discuss how to extend the convergence of optimization problem (2) into (1), with additional assumptions.

C.1 Proof of Lemma 3.1

For convenience, the lemma is restated as follows.

Lemma 3.1. *For any $\boldsymbol{\phi} \in \Phi$, the gradient estimation with forward gradient method:*

$$\hat{\nabla} h(\boldsymbol{\phi}) = \frac{1}{b}\sum_{i=1}^{b} \nabla h(\boldsymbol{\phi})\boldsymbol{v}_i\boldsymbol{v}_i^\top,$$

where $\boldsymbol{v}_i \sim \mathrm{Unif}(\{-1,1\}^N)$, and b denotes the sample size, satisfies

$$\mathbb{E}\|\hat{\nabla} h(\boldsymbol{\phi}) - \nabla h(\boldsymbol{\phi})\|^2 = \frac{1}{\rho}\|\nabla h(\boldsymbol{\phi})\|^2,$$

where $\rho := \frac{b}{N-1} \in (0,1]$ as b is selected from $1,\ldots,N-1$.

Proof. We start by computing the variance of one-sample estimation, $\hat{\nabla} h(\boldsymbol{\phi}) = \nabla h(\boldsymbol{\phi})\boldsymbol{v}\boldsymbol{v}^\top$. Since $\mathbb{E}[\boldsymbol{v}\boldsymbol{v}^\top] = \mathbf{I}$, we know that $\mathbb{E}[\hat{\nabla} h(\boldsymbol{\phi})] = \nabla h(\boldsymbol{\phi})$. Consequently,

$$\begin{aligned} &\mathbb{E}\|\hat{\nabla} h(\boldsymbol{\phi}) - \mathbb{E}\hat{\nabla} h(\boldsymbol{\phi})\|^2 = \mathbb{E}\|\nabla h(\boldsymbol{\phi})(\boldsymbol{v}\boldsymbol{v}^\top - \mathbf{I})\|^2 \\ &= \mathbb{E}[\nabla h(\boldsymbol{\phi})^\top(\boldsymbol{v}\boldsymbol{v}^\top - \mathbf{I})^\top(\boldsymbol{v}\boldsymbol{v}^\top - \mathbf{I})(\nabla h(\boldsymbol{\phi}))] \\ &= \mathbb{E}[(\nabla h(\boldsymbol{\phi}))^\top(\boldsymbol{v}\boldsymbol{v}^\top)^\top\boldsymbol{v}\boldsymbol{v}^\top\nabla h(\boldsymbol{\phi}) - 2(\nabla h(\boldsymbol{\phi}))^\top(\boldsymbol{v}\boldsymbol{v}^\top)^\top\nabla h(\boldsymbol{\phi}) + (\nabla h(\boldsymbol{\phi}))^\top\nabla h(\boldsymbol{\phi})] \\ &= \mathbb{E}\|\nabla h(\boldsymbol{\phi})\boldsymbol{v}\boldsymbol{v}^\top\|^2 - 2\mathbb{E}\|\nabla h(\boldsymbol{\phi})\boldsymbol{v}\|^2 + \|\nabla h(\boldsymbol{\phi})\|^2. \end{aligned} \tag{30}$$

Since $\boldsymbol{v}$ is an N-dimensional Rademacher random variable, we have $\mathbb{E}\|\nabla h(\boldsymbol{\phi})\boldsymbol{v}\|^2 = \|\nabla h(\boldsymbol{\phi})\|^2$ and $\mathbb{E}\|\boldsymbol{v}\boldsymbol{v}^\top\|^2 = N$. Then,

$$\begin{aligned} (30) &= \mathbb{E}\|\nabla h(\boldsymbol{\phi})\boldsymbol{v}\boldsymbol{v}^\top\|^2 - \|\nabla h(\boldsymbol{\phi})\|^2 \\ &= (N-1)\|\nabla h(\boldsymbol{\phi})\|^2. \end{aligned}$$

For the multi-sample estimation $\hat{\nabla} h(\boldsymbol{\phi}) = \frac{1}{b}\sum_{i=1}^{b} \nabla h(\boldsymbol{\phi})\boldsymbol{v}_i{\boldsymbol{v}_i}^\top$. Since$\boldsymbol{v}_i$ are i.i.d. sampled, and $\mathbb{E}[\hat{\nabla} h(\boldsymbol{\phi})] = \nabla h(\boldsymbol{\phi})$, we have

$$\begin{aligned}\mathbb{E}\|\hat{\nabla} h(\boldsymbol{\phi}) - \mathbb{E}\hat{\nabla} h(\boldsymbol{\phi})\|^2 &= \mathbb{E}\left\|\nabla h(\boldsymbol{\phi})\frac{1}{b}\sum_{i=1}^{b}(\boldsymbol{v}_i{\boldsymbol{v}_i}^\top - \mathbf{I})\right\|^2 \\ &= \frac{1}{b^2}\sum_{i=1}^{b}\mathbb{E}\|\nabla h(\boldsymbol{\phi})(\boldsymbol{v}_i{\boldsymbol{v}_i}^\top - \mathbf{I})\|^2 \\ &= \frac{N-1}{b}\|\nabla h(\boldsymbol{\phi})\|^2.\end{aligned}$$

□

C.2 Proof of Theorem 3.4

To prove our main result (Theorem 3.4), we first establish useful smoothness properties of the hyperparameter learned from solving the lower-level optimization problem. Subsequently, we establish the smoothness of the meta objective function examined at the approximated lower-level parameters in Lemma B.3.

Regarding the lower-level parameter $\boldsymbol{\theta}(\boldsymbol{\phi})$, we present the following lemma, which is based on Assumption 3.3, and establishes that $\boldsymbol{\theta}(\boldsymbol{\phi})$ inherits similar Lipschitz continuity and smoothness properties as $\boldsymbol{\Omega}_t$.

Lemma B.2. *Under Assumptions 3.2 and 3.3, $\boldsymbol{\theta}_T(\boldsymbol{\phi})$ is $C_{\boldsymbol{Z}}$-Lipchitz and $L_{\boldsymbol{Z}}$-smooth,* i.e.*, for any $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$,*

$$\|\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \boldsymbol{\theta}_T(\boldsymbol{\phi}')\| \le C_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|, \quad \|\nabla\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \nabla\boldsymbol{\theta}_T(\boldsymbol{\phi}')\| \le L_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,$$

where $C_{\boldsymbol{Z}} = \frac{C_{\boldsymbol{\Omega}}^{T+2} - C_{\boldsymbol{\Omega}}}{C_{\boldsymbol{\Omega}} - 1}$ and $L_{\boldsymbol{Z}} = L_{\boldsymbol{\Omega}}\left[C_{\boldsymbol{\Omega}}^T + \frac{C_{\boldsymbol{\Omega}}^{T+2}T}{C_{\boldsymbol{\Omega}}-1} - \frac{C_{\boldsymbol{\Omega}}^T - 1}{(C_{\boldsymbol{\Omega}}-1)^2}\right]$.

Proof. We start with the proof of Lipshitz continuity. For any pair of $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$, using (2) and Assumption 3.3, we have

$$\begin{aligned}\|\boldsymbol{\theta}_s(\boldsymbol{\phi}) - \boldsymbol{\theta}_s(\boldsymbol{\phi}')\| &= \|\boldsymbol{\Omega}(\boldsymbol{\theta}_{s-1}(\boldsymbol{\phi}), \boldsymbol{\phi}) - \boldsymbol{\Omega}(\boldsymbol{\theta}_{s-1}(\boldsymbol{\phi}'), \boldsymbol{\phi}')\| \\ &\le C_{\boldsymbol{\Omega}}\|\boldsymbol{\theta}_{s-1}(\boldsymbol{\phi}) - \boldsymbol{\theta}_{s-1}(\boldsymbol{\phi}')\| + C_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|.\end{aligned} \tag{31}$$

Applying (31) recursively over $s = 1, \dots, t$ gives

$$\begin{aligned}\|\boldsymbol{\theta}_t(\boldsymbol{\phi}) - \boldsymbol{\theta}_t(\boldsymbol{\phi}')\| &\le C_{\boldsymbol{\Omega}}^t\|\boldsymbol{\theta}_0(\boldsymbol{\phi}) - \boldsymbol{\theta}_0(\boldsymbol{\phi}')\| + \sum_{s=1}^{t} C_{\boldsymbol{\Omega}}^s\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\ &\le \sum_{s=1}^{t+1} C_{\boldsymbol{\Omega}}^s\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| = \frac{C_{\boldsymbol{\Omega}}^{t+2} - C_{\boldsymbol{\Omega}}}{C_{\boldsymbol{\Omega}} - 1}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,\end{aligned} \tag{32}$$

where the last inequality holds from the fact that $\boldsymbol{\theta}_0 = \boldsymbol{\Omega}_0$ as well as Assumption 3.3, and the subsequent equality follows from the geometric series summation formula.

Therefore, $\boldsymbol{\theta}_t(\boldsymbol{\phi})$ is $C_{\boldsymbol{Z}}(t)$-Lipchitz, where $C_{\boldsymbol{Z}}(t) := \frac{C_{\boldsymbol{\Omega}}^{t+2} - C_{\boldsymbol{\Omega}}}{C_{\boldsymbol{\Omega}} - 1}$. Substituting $t = T$, we get $C_{\boldsymbol{Z}} = C_{\boldsymbol{Z}}(T) = \frac{C_{\boldsymbol{\Omega}}^{T+2} - C_{\boldsymbol{\Omega}}}{C_{\boldsymbol{\Omega}} - 1}$.

We now proceed with the proof of $L_{\boldsymbol{Z}}$-smoothness. For simplicity of notation, we follow (4) and denote

$$\boldsymbol{Z}_t(\boldsymbol{\phi}) = \nabla\boldsymbol{\theta}_t(\boldsymbol{\phi}); \quad \boldsymbol{A}_t(\boldsymbol{\phi}) = \frac{\partial\boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\theta}_{t-1}}; \quad \boldsymbol{B}_t(\boldsymbol{\phi}) = \frac{\partial\boldsymbol{\Omega}_t(\boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\phi}}.$$

Subsequently, considering the update rule $\boldsymbol{Z}_t(\boldsymbol{\phi}) = \boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) + \boldsymbol{B}_t(\boldsymbol{\phi})$ of Forward Gradient Unrolling (4), we have

$$\begin{aligned}&\|\nabla\boldsymbol{\theta}_t(\boldsymbol{\phi}) - \nabla\boldsymbol{\theta}_t(\boldsymbol{\phi}')\| \\ &= \|\boldsymbol{Z}_t(\boldsymbol{\phi}) - \boldsymbol{Z}_t(\boldsymbol{\phi}')\| \\ &\overset{(4)}{=} \|\boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) + \boldsymbol{B}_t(\boldsymbol{\phi}) - \left[\boldsymbol{A}_t(\boldsymbol{\phi}')\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}') + \boldsymbol{B}_t(\boldsymbol{\phi}')\right]\| \\ &\le \|\boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) - \boldsymbol{A}_t(\boldsymbol{\phi}')\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + \|\boldsymbol{B}_t(\boldsymbol{\phi}) - \boldsymbol{B}_t(\boldsymbol{\phi}')\| \\ &= \|\boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) - \boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}') + \boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}') - \boldsymbol{A}_t(\boldsymbol{\phi}')\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + \|\boldsymbol{B}_t(\boldsymbol{\phi}) - \boldsymbol{B}_t(\boldsymbol{\phi}')\| \\ &\le \|\boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) - \boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + \|\boldsymbol{A}_t(\boldsymbol{\phi})\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}') - \boldsymbol{A}_t(\boldsymbol{\phi}')\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| \\ &\quad + \|\boldsymbol{B}_t(\boldsymbol{\phi}) - \boldsymbol{B}_t(\boldsymbol{\phi}')\| \\ &\le \|\boldsymbol{A}_t(\boldsymbol{\phi})\| \cdot \|\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) - \boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + \|\boldsymbol{A}_t(\boldsymbol{\phi}) - \boldsymbol{A}_t(\boldsymbol{\phi}')\| \cdot \|\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + \|\boldsymbol{B}_t(\boldsymbol{\phi}) - \boldsymbol{B}_t(\boldsymbol{\phi}')\| \\ &\le C_{\boldsymbol{\Omega}}\|\boldsymbol{Z}_{t-1}(\boldsymbol{\phi}) - \boldsymbol{Z}_{t-1}(\boldsymbol{\phi}')\| + (C_{\boldsymbol{Z}}(t) + 1)L_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,\end{aligned} \tag{33}$$

where the last inequality follows from Assumption 3.3 that $\boldsymbol{\Omega}_0(\boldsymbol{\phi})$ and $\boldsymbol{\Omega}_{1:T}(\boldsymbol{\psi})$ are $C_{\boldsymbol{\Omega}}$-Lipshitz and $L_{\boldsymbol{\Omega}}$-smooth, and the previously proved result that $\boldsymbol{\theta}_t(\boldsymbol{\phi})$ is $C_{\boldsymbol{Z}}(t)$-Lipchitz.

Noting that $\boldsymbol{Z}_0(\boldsymbol{\phi}) = \nabla\boldsymbol{\theta}_0(\boldsymbol{\phi}) = \nabla\boldsymbol{\Omega}_0(\boldsymbol{\phi})$, applying (33) recursively over $t = 1, \dots, T$ gives

$$\begin{aligned}
\|\nabla\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \nabla\boldsymbol{\theta}_T(\boldsymbol{\phi}')\| &\le C_{\boldsymbol{\Omega}}^T\|\boldsymbol{Z}_0(\boldsymbol{\phi}) - \boldsymbol{Z}_0(\boldsymbol{\phi}`)\| + \sum_{t=1}^{T} C_{\boldsymbol{\Omega}}^{T-t}(C_{\boldsymbol{Z}}(t)+1)L_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&= C_{\boldsymbol{\Omega}}^T\|\nabla\boldsymbol{\Omega}_0(\boldsymbol{\phi}) - \nabla\boldsymbol{\Omega}_0(\boldsymbol{\phi}')\| + C_{\boldsymbol{\Omega}}^T\sum_{t=1}^{T}\frac{C_{\boldsymbol{Z}}(t)+1}{C_{\boldsymbol{\Omega}}^t}L_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&\le L_{\boldsymbol{\Omega}}C_{\boldsymbol{\Omega}}^T\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + C_{\boldsymbol{\Omega}}^T\sum_{t=1}^{T}\frac{C_{\boldsymbol{\Omega}}^{t+2}-1}{C_{\boldsymbol{\Omega}}^t(C_{\boldsymbol{\Omega}}-1)}L_{\boldsymbol{\Omega}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&= L_{\boldsymbol{\Omega}}\left[C_{\boldsymbol{\Omega}}^T + C_{\boldsymbol{\Omega}}^T\sum_{t=1}^{T}\frac{C_{\boldsymbol{\Omega}}^2}{C_{\boldsymbol{\Omega}}-1} - \frac{C_{\boldsymbol{\Omega}}^T}{C_{\boldsymbol{\Omega}}-1}\sum_{t=1}^{T}\frac{1}{C_{\boldsymbol{\Omega}}^t}\right]\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&= L_{\boldsymbol{\Omega}}\left[C_{\boldsymbol{\Omega}}^T + \frac{C_{\boldsymbol{\Omega}}^{T+2}T}{C_{\boldsymbol{\Omega}}-1} - \frac{C_{\boldsymbol{\Omega}}^T}{C_{\boldsymbol{\Omega}}-1}\frac{\frac{1}{C_{\boldsymbol{\Omega}}}(1-\frac{1}{C_{\boldsymbol{\Omega}}^T})}{1-\frac{1}{C_{\boldsymbol{\Omega}}}}\right]\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&= L_{\boldsymbol{\Omega}}\left[C_{\boldsymbol{\Omega}}^T + \frac{C_{\boldsymbol{\Omega}}^{T+2}T}{C_{\boldsymbol{\Omega}}-1} - \frac{C_{\boldsymbol{\Omega}}^T-1}{(C_{\boldsymbol{\Omega}}-1)^2}\right]\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,
\end{aligned}$$

where the third line follows from Assumption 3.3 that $\boldsymbol{\Omega}_0(\boldsymbol{\phi})$ is $L_{\boldsymbol{\Omega}}$-smooth and the choice $C_{\boldsymbol{Z}}(t) = \frac{C_{\boldsymbol{\Omega}}^{t+2}-C_{\boldsymbol{\Omega}}}{C_{\boldsymbol{\Omega}}-1}$ (which gives $C_{\boldsymbol{Z}}(t)+1 = \frac{C_{\boldsymbol{\Omega}}^{t+2}-1}{C_{\boldsymbol{\Omega}}-1}$), and the fifth line again uses the geometric series summation formula.

Hence, $\boldsymbol{\theta}_T(\boldsymbol{\phi})$ is $L_{\boldsymbol{Z}}$-smooth with $L_{\boldsymbol{Z}} = L_{\boldsymbol{\Omega}}\left[C_{\boldsymbol{\Omega}}^T + \frac{C_{\boldsymbol{\Omega}}^{T+2}T}{C_{\boldsymbol{\Omega}}-1} - \frac{C_{\boldsymbol{\Omega}}^T-1}{(C_{\boldsymbol{\Omega}}-1)^2}\right]$. □

Next, we provide a lemma establishing that the upper-level objective f, evaluated at the learned parameter $(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})$, also adheres to certain smoothness properties.

Lemma B.3. *Define $h(\boldsymbol{\phi}) := f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})$. Under Assumptions 3.2 and 3.3, $h(\boldsymbol{\phi})$ is L_h-smooth, i.e., for any $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$,*

$$\|\nabla h(\boldsymbol{\phi}) - \nabla h(\boldsymbol{\phi}')\| \le L_h\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,$$

where $L_h = (C_{\boldsymbol{Z}}+1)^2L + CL_{\boldsymbol{Z}}$, with $C_{\boldsymbol{Z}}$ and $L_{\boldsymbol{Z}}$ defined in Lemma B.2.

Proof. For simplicity of notation, we follow (5) and denote

$$\boldsymbol{Z}_t(\boldsymbol{\phi}) = \nabla\boldsymbol{\theta}_t(\boldsymbol{\phi}); \quad \boldsymbol{c}_T(\boldsymbol{\phi}) = \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\phi}}; \quad \boldsymbol{d}_T(\boldsymbol{\phi}) = \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\theta}_T}.$$

For any $\boldsymbol{\phi}, \boldsymbol{\phi}' \in \Phi$, following a similar proof as Lemma B.2, we have

$$\begin{aligned}
&\|\nabla h(\boldsymbol{\phi}) - \nabla h(\boldsymbol{\phi}')\| \\
&\overset{(5)}{=} \|\boldsymbol{d}_T(\boldsymbol{\phi})\boldsymbol{Z}_T(\boldsymbol{\phi}) + \boldsymbol{c}(\boldsymbol{\phi}) - (\boldsymbol{d}_T(\boldsymbol{\phi}')\boldsymbol{Z}_T(\boldsymbol{\phi}') + \boldsymbol{c}(\boldsymbol{\phi}'))\| \\
&\le \|\boldsymbol{d}_T(\boldsymbol{\phi})\boldsymbol{Z}_T(\boldsymbol{\phi}) - \boldsymbol{d}_T(\boldsymbol{\phi}')\boldsymbol{Z}_T(\boldsymbol{\phi}')\| + \|\boldsymbol{c}_T(\boldsymbol{\phi}) - \boldsymbol{c}_T(\boldsymbol{\phi}')\| \qquad (34)\\
&= \|\boldsymbol{d}_T(\boldsymbol{\phi})\boldsymbol{Z}_T(\boldsymbol{\phi}) - \boldsymbol{d}_T(\boldsymbol{\phi}')\boldsymbol{Z}_T(\boldsymbol{\phi}) + \boldsymbol{d}_T(\boldsymbol{\phi}')\boldsymbol{Z}_T(\boldsymbol{\phi}) - \boldsymbol{d}_T(\boldsymbol{\phi}')\boldsymbol{Z}_T(\boldsymbol{\phi}')\| + \|\boldsymbol{c}_T(\boldsymbol{\phi}) - \boldsymbol{c}_T(\boldsymbol{\phi}')\| \\
&\le \|\boldsymbol{d}_T(\boldsymbol{\phi}) - \boldsymbol{d}_T(\boldsymbol{\phi}')\| \cdot \|\boldsymbol{Z}_T(\boldsymbol{\phi})\| + \|\boldsymbol{d}_T(\boldsymbol{\phi}')\| \cdot \|\boldsymbol{Z}_T(\boldsymbol{\phi}) - \boldsymbol{Z}_T(\boldsymbol{\phi}')\| + \|\boldsymbol{c}_T(\boldsymbol{\phi}) - \boldsymbol{c}_T(\boldsymbol{\phi}')\|
\end{aligned}$$

Subsequently, we deduce that

$$\begin{aligned}
(34) &\le C_{\boldsymbol{Z}}\|\boldsymbol{d}_T(\boldsymbol{\phi}) - \boldsymbol{d}_T(\boldsymbol{\phi}')\| + CL_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + \|\boldsymbol{c}_T(\boldsymbol{\phi}) - \boldsymbol{c}_T(\boldsymbol{\phi}')\| \\
&\le C_{\boldsymbol{Z}}\left(\left\|\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\theta}_T} - \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi})}{\partial\boldsymbol{\theta}_T}\right\| + \left\|\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi})}{\partial\boldsymbol{\theta}_T} - \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi}')}{\partial\boldsymbol{\theta}_T}\right\|\right) \\
&\quad + \left(\left\|\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})}{\partial\boldsymbol{\phi}} - \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi})}{\partial\boldsymbol{\phi}}\right\| + \left\|\frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi})}{\partial\boldsymbol{\phi}} - \frac{\partial f(\boldsymbol{\theta}_T(\boldsymbol{\phi}'), \boldsymbol{\phi}')}{\partial\boldsymbol{\phi}'}\right\|\right) \\
&\quad + CL_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&\le C_{\boldsymbol{Z}}L\|\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \boldsymbol{\theta}_T(\boldsymbol{\phi}')\| + C_{\boldsymbol{Z}}L\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + L\|\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \boldsymbol{\theta}_T(\boldsymbol{\phi}')\| + L\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&\quad + CL_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&\le C_{\boldsymbol{Z}}LC_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + C_{\boldsymbol{Z}}L\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + LC_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + L\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| + CL_{\boldsymbol{Z}}\|\boldsymbol{\phi} - \boldsymbol{\phi}'\| \\
&= [(C_{\boldsymbol{Z}}+1)^2L + CL_{\boldsymbol{Z}}]\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|,
\end{aligned}$$

where the first, third, and fourth lines all follow directly from Lemma B.2 and Assumption 3.3 (recall that the latter states that f is L-Lipschitz and C-smooth).

Therefore, $h(\boldsymbol{\phi})$ is L_h-smooth with $L_h = (C_{\boldsymbol{Z}} + 1)^2 L + CL_{\boldsymbol{Z}}$. □

Now based on the aforementioned lemmas, we put forward the proof of our main theorem: the convergence analysis for our bilevel optimization method $\mathbf{(FG)^2U}$.

Theorem 3.4 (Convergence)**.** *Suppose that Asumption 3.2 and Assumption 3.3 hold. Setting the learning rate* $\beta = \frac{\rho}{(\rho+1)L_h}$ *for gradient descent over the hyperparameter* $\boldsymbol{\phi}$*, we have*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\|\nabla h(\boldsymbol{\phi}_k)\|^2\right] \leq \frac{4L_h\left(\mathbb{E}[h(\boldsymbol{\phi}_0)] - \min_{\boldsymbol{\phi}} h(\boldsymbol{\phi})\right)}{\rho K}. \tag{35}$$

Proof. We have

$$\begin{aligned}
&h(\boldsymbol{\phi}_{k+1}) - h(\boldsymbol{\phi}_k) \\
&\leq \left\langle \nabla h(\boldsymbol{\phi}_k), \boldsymbol{\phi}_{k+1} - \boldsymbol{\phi}_k \right\rangle + \frac{L_h}{2}\|\boldsymbol{\phi}_{k+1} - \boldsymbol{\phi}_k\|^2 \\
&= -\beta\langle \nabla h(\boldsymbol{\phi}_k), \hat{\nabla} h(\boldsymbol{\phi}_k)\rangle + \frac{\beta^2 L_h}{2}\|\hat{\nabla} h(\boldsymbol{\phi}_k)\|^2 \\
&= -\beta\langle \nabla h(\boldsymbol{\phi}_k), \hat{\nabla} h(\boldsymbol{\phi}_k)\rangle + \frac{\beta^2 L_h}{2}\|\nabla h(\boldsymbol{\phi}_k) + \hat{\nabla} h(\boldsymbol{\phi}_k) - \nabla h(\boldsymbol{\phi}_k)\|^2 \\
&= -\frac{\beta^2 L_h}{2}\|\nabla h(\boldsymbol{\phi}_k)\|^2 + (\beta^2 L_h - \beta)\langle \nabla h(\boldsymbol{\phi}_k), \hat{\nabla} h(\boldsymbol{\phi}_k)\rangle + \frac{\beta^2 L_h}{2}\|\nabla h(\boldsymbol{\phi}_k) - \hat{\nabla} h(\boldsymbol{\phi}_k)\|^2,
\end{aligned} \tag{36}$$

where the second line is a well-known inequality for smooth functions with the L_h-smoothness itself following from Lemma B.3, and the third line uses the gradient descent rule $\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_k - \beta\hat{\nabla} h(\boldsymbol{\phi}_k)$.

By Lemma 3.1 and the fact that $\hat{\nabla} h$ is unbiased (see the proof of Lemma 3.1), we know that

$$\mathbb{E}[\langle \nabla h(\boldsymbol{\phi}_k), \hat{\nabla} h(\boldsymbol{\phi}_k)\rangle | \boldsymbol{\phi}_k] = \|\nabla h(\boldsymbol{\phi}_k)\|^2;$$
$$\mathbb{E}[\|\nabla h(\boldsymbol{\phi}_k) - \hat{\nabla} h(\boldsymbol{\phi}_k)\|^2 | \boldsymbol{\phi}_k] = \frac{1}{\rho}\|\nabla h(\boldsymbol{\phi}_k)\|^2.$$

Therefore, taking the conditional expectation $\mathbb{E}[\cdot \mid \boldsymbol{\phi}_k]$ over (36) gives

$$\mathbb{E}[h(\boldsymbol{\phi}_{k+1})|\boldsymbol{\phi}_k] - h(\boldsymbol{\phi}_k) \leq -\left[\beta - \left(1 + \frac{1}{\rho}\right)\frac{\beta^2 L_h}{2}\right]\|\nabla h(\boldsymbol{\phi}_k)\|^2. \tag{37}$$

Furthermore, taking the full expectation and telescoping (37) over k form 0 to $K-1$ yields

$$\begin{aligned}
\frac{1}{K}\sum_{k=0}^{K-1}\left[\beta - \frac{(\rho+1)L_h}{2\rho}\beta^2\right]\mathbb{E}\left[\|\nabla h(\boldsymbol{\phi}_k)\|^2\right] &\leq \frac{\mathbb{E}[h(\boldsymbol{\phi}_0)] - \mathbb{E}[h(\boldsymbol{\phi}_K)]}{K} \\
&\leq \frac{\mathbb{E}[h(\boldsymbol{\phi}_0)] - \min_{\boldsymbol{\phi}} h(\boldsymbol{\phi})}{K}.
\end{aligned} \tag{38}$$

Choosing $\beta = \frac{\rho}{(\rho+1)L_h}$, we have

$$\begin{aligned}
\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla h(\boldsymbol{\phi}_k)\|^2\right] &\leq \frac{2(\rho+1)L_h\left(\mathbb{E}[h(\boldsymbol{\phi}_0)] - \min_{\boldsymbol{\phi}} h(\boldsymbol{\phi})\right)}{\rho K} \\
&\leq \frac{4L_h\left(\mathbb{E}[h(\boldsymbol{\phi}_0)] - \min_{\boldsymbol{\phi}} h(\boldsymbol{\phi})\right)}{\rho K}.
\end{aligned} \tag{39}$$

Hence, Algorithm 1 requires $\mathcal{O}(\epsilon^{-1}\rho^{-1})$ steps to attain an ϵ-accurate stationary point. □

C.3 Extended Discussions

Convergence of Problem (1). To extend the convergence of optimization problem (2) into (1), we need to assume that the lower-level objective function g is strongly convex w.r.t. $\boldsymbol{\theta}$ as commonly done by previous works [60, 28]. From the strong convexity and first-order smoothness (Assumption 3.2) of g, we have 1) the zeroth and first-order smoothness of $\boldsymbol{\theta}^*(\boldsymbol{\phi})$; 2) $\|\boldsymbol{\theta}_T(\boldsymbol{\phi}') - \boldsymbol{\theta}^*(\boldsymbol{\phi}')\| \to 0$ as $T \to +\infty$. Then the inequality

$$\|\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \boldsymbol{\theta}_T(\boldsymbol{\phi}')\| \leq \|\boldsymbol{\theta}_T(\boldsymbol{\phi}) - \boldsymbol{\theta}^*(\boldsymbol{\phi})\| + \|\boldsymbol{\theta}_T(\boldsymbol{\phi}') - \boldsymbol{\theta}^*(\boldsymbol{\phi}')\| + \|\boldsymbol{\theta}^*(\boldsymbol{\phi}) - \boldsymbol{\theta}^*(\boldsymbol{\phi}')\| \tag{40}$$

implies the the zeroth and first-order smoothness of $\boldsymbol{\theta}_T(\boldsymbol{\phi})$. Following the same line of proof as presented in our paper, we derive the smoothness of $f(\boldsymbol{\theta}^*(\boldsymbol{\phi}), \boldsymbol{\phi})$ and $f(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi})$, and subsequently the convergence of either problem (1) or (2).

However, as discussed in Section 2, given that the scope of this paper is large-scale BO, the inner optimization typically involves deep neural networks. Therefore, the optimal parameters are not explicitly accessible and can only be estimated through iterative procedures. Most related works [9, 60, 28] are implicitly or explicitly solving (2) instead of (1). Additionally, it is important to acknowledge that achieving strong convexity is often unfeasible in practical applications. Consequently, we focus on (2), aiming to present a more practical convergence theory that proves the effectiveness of our method.

D Zeroth-Order Derivative Estimator

In this section, we give a more detailed introduction to zeroth-order (ZO) derivative estimators. These estimators are pivotal in scenarios where the computation of exact derivatives is either infeasible due to memory constraints or computationally prohibitive. Apart from the forward gradient method employed in $\mathbf{(FG)^2U}$, randomized smoothing (RS) is another widely-used derivative estimator, both in Reinforcement Learning [22, 32] and Large Language Models [18, 47, 74].

For a function $F : \mathbb{R}^n \to \mathbb{R}$, gradient estimation via RS can be mathematically formulated as:

$$\nabla_{\boldsymbol{x}} F \approx \mathbb{E}_{v\sim\mathcal{N}(0,I)}\left[\frac{F(\boldsymbol{x}+\epsilon v)-F(\boldsymbol{x})}{\epsilon}v^\top\right] \approx \frac{1}{b}\sum_{i=1}^{b}\frac{F(\boldsymbol{x}+\epsilon v_i)-F(\boldsymbol{x})}{\epsilon}v_i^\top, \tag{41}$$

where b is the number of random samples, ϵ is the smoothing parameter, v_i are samples drawn from a standard Gaussian distribution. Regarding the accuracy of estimation, it has been shown in [13, 42] that the variance of RS is roughly in the order of $O(N/b)$, which is the same as FG as proved in Lemma 3.1.

RS stands out particularly in its ability to estimate gradients of functions evaluated through black-box systems, where internal operations are inaccessible or highly complex. This characteristic makes RS exceptionally valuable in practical applications such as adversarial robustness and black-box optimization, where obtaining direct gradients might not be possible. Another advantage of RS is its robustness against noise and discontinuities in the function landscape. Unlike deterministic methods, the stochastic nature of RS allows it to approximate the gradient over a smoothed version of the function, providing stability in scenarios where slight perturbations can lead to substantial changes in the output.

While RS provides robust gradient estimates across various scenarios, it is critical to recognize that RS inherently introduces bias if the expectation is not computed during inference. In many CV and NLP applications, the computational expense of Monte Carlo sampling at the evaluation stage is prohibitive, leading to a biased estimation when using RS. However, in the context of inverse PDE problems, where the inner-loop solvers are non-differentiable numerical solvers, we employ RS as a zeroth-order derivative estimator.

E Detailed Task Description

E.1 Data Condensation

In the era of rapid advancement in machine learning, a multitude of foundation models [11, 6, 55] has benefited from training on large-scale datasets, exhibiting formidable performance that models trained on small-scale data cannot match. However, the exponential growth of data also presents challenges: (1) Models updated with only new data are prone to catastrophic forgetting [20] while retaining all historical data for subsequent training imposes significant storage and computational burdens. (2) Applications within the realm of meta-learning, such as hyperparameter tuning [46, 43] and neural architecture search [78, 38], necessitate multiple training iterations over datasets. The computational cost of these operations scales dramatically with the size of the datasets, posing a bottleneck for efficiency and scalability. (3) The widespread dissemination and utilization of datasets have raised significant concerns regarding privacy and copyright [12].

To overcome the challenges posed by large-scale datasets, a line of work known as data condensation [68, 72] has been proposed, with the idea to generate a compact, synthesized dataset, designed to elicit similar behaviors in machine learning models as those trained with the original, massive dataset. The objectives of the mainstream principles [72] designed for data condensation can be naturally formulated as a bi-level optimization problem. We focus on the best-known principle *performance matching* [72] on classification task, which can be formulated as,

$$\min_{\mathcal{D}_o} \mathcal{L}(\theta_T; \mathcal{D}_o), \quad \text{where } \theta_t = \theta_{t-1} - \eta\nabla\mathcal{L}(\theta_{t-1}; \mathcal{D}_c), \ \ t = 1, \dots, T, \tag{42}$$

We conduct our experiments to condense the following image datasets:

- **MNIST** [35]: a handwritten digits dataset containing $60,000$ training images and $10,000$ testing images with the size of 28×28 from 10 categories.
- **CIFAR 10/100** [31]: colored natural images datasets contraining $50,000$ training images and $10,000$ testing images from $10/100$ categories, respectively.

The scale of the condensed dataset will fundamentally impact the results. Therefore, we consider different scales for each dataset, with images per class set to 1, 10, and 50. The condensed dataset will be used to train random initialized models, and evaluated on a test dataset.

E.2 Meta Learning Online Adaptation of Language Models

The online adaptation of language models (LM) [34, 27] has been studied recently to keep the knowledge of LM updated to date. However, trivial auto-regressive fine-tuning the LM with uniform weights for all tokens results in poor performance in downstream tasks, as the default average negative log-likelihood (NLL) loss does not accurately reflect the importance of tokens [25]. To address the issue, [25] proposed Context-aware Meta-learned Loss Scaling (CaMeLS) to meta-learning the weights of tokens for effective online adaption. More formally, let $\boldsymbol{\theta}$ denote the parameter of the base model for adaptation, $\boldsymbol{\phi}$ denote the parameter of a parametric weight model to assign weights for each token, the meta-learning online adaption of LM can be formulated as the following bi-level optimization,

$$\min_{\boldsymbol{\phi}} \mathcal{L}_{meta}(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi}) \quad s.t.\ \boldsymbol{\theta}_t(\boldsymbol{\phi}) = \boldsymbol{\theta}_{t-1}(\boldsymbol{\phi}) - \eta \nabla_\theta \mathcal{L}_{train}(\boldsymbol{\theta}_{t-1}, w_{\boldsymbol{\phi}}),\ \ t = 1, \ldots, T. \tag{43}$$

We follow the setting studied by [25], where the downstream task is question-answering. The meta-objective consists of a question-answering term measuring the performance gained from adaptation, and a locality term that prevents the updated base model parameters from excessively changing the base model's behavior. Let $\mathcal{D}_{QA}$ denotes the question-answering dataset, $\mathcal{D}_{loc}$ denotes the locality dataset, and $c \in \mathbb{R}^+$ denotes the weight of the locality term, then the meta objective is formally defined as

$$\mathcal{L}_{meta}(\boldsymbol{\theta}_T(\boldsymbol{\phi}), \boldsymbol{\phi}) := \mathbb{E}_{q,a\sim\mathcal{D}_{QA}} - \log p_{\boldsymbol{\theta}_T}(a|q) + c\mathbb{E}_{x\sim\mathcal{D}_{loc}} \sum_i \mathrm{KL}(p_{\boldsymbol{\theta}_T}(\cdot|x_{:i}) \parallel p_{\boldsymbol{\theta}_0}(\cdot|x_{:i})). \tag{44}$$

The inner objective is defined as a weighted NLL loss, where the weights are determined by the weight model $w_{\boldsymbol{\phi}}$,

$$\mathcal{L}_{train}(\boldsymbol{\theta}, w_{\boldsymbol{\phi}}) := \mathbb{E}_{x\sim\mathcal{D}_{train}} \sum_i -w_{\boldsymbol{\phi}}(x_i, x) \log p_{\boldsymbol{\theta}}(x_i|x_{:i}). \tag{45}$$

The trained weight model is then fine-tuned on unseen online documents and evaluated on corresponding question-answering tasks.

E.3 Data-driven Discovery of Partial Differential Equations (PDEs)

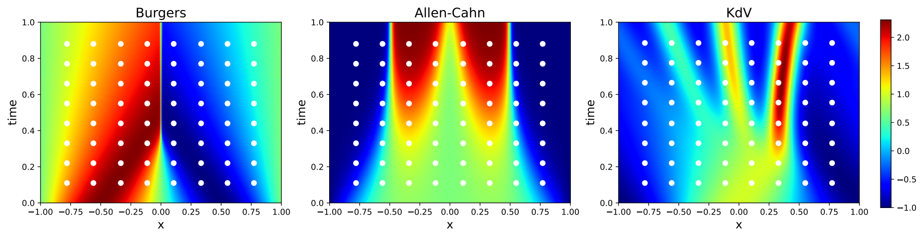


Figure E.1: Visualization of the 2D latent solutions for the Burgers, Allen-Cahn, and KdV equations. The observed data are sampled on an 8×8 grid, denoted by white points.

We conducted experiments on three non-linear PDEs, with the latent solutions visualized in Figure E.1. The PDE structures (46) (47) (48) are assumed to be known while the PDE parameters ν in (46) (47) (48) are assumed to be unknown. For each equation, 64 observed data points are sampled on an 8×8 grid. The objective is to predict the unknown PDE parameters using the observed data. The predicted PDE parameters are evaluated by comparing the error with the ground truth, as well as the error in the corresponding prediction of the latent solution.

E.3.1 Burgers Equation

The nonlinear viscous Burgers equation is a pivotal partial differential equation arising in diverse domains of applied mathematics such as fluid mechanics, nonlinear acoustics, and traffic flow. This equation can be deduced from the Navier-Stokes equations for the velocity field by omitting the pressure gradient term. In our experiment, the equation along with Dirichlet boundary conditions, is expressed as follows:

$$\begin{aligned}
&u_t + uu_x - \nu u_{xx} = 0,\ x \in [-1,1], t \in [0,1], \nu > 0, \\
&u(0,x) = -\sin(\pi x), \\
&u(t,-1) = u(t,1) = 0,
\end{aligned} \tag{46}$$

with actual viscosity $\nu = \frac{0.01}{\pi} \approx 0.0031831$. For PINN, following [44], we enforced the initial condition into the output by choosing a surrogate model of the solution as

$$\hat{u}(x) = (1 - \exp(-t))\,\mathrm{NN}(x;\boldsymbol{\theta}) - \sin(\pi x),$$

where $\mathrm{NN}(x;\boldsymbol{\theta})$ is a neural network.

E.3.2 Allen-Cahn Equation

The Allen-Cahn equation is a reaction-diffusion equation of mathematical physics describing the process of phase separation in multi-component alloy systems, including order-disorder transitions. In our experiment, it is expressed as follows:

$$\begin{aligned}
&u_t - \nu u_{xx} = 5(u - u^3), x \in [-1,1], t \in [0,1], \nu > 0, \\
&u(0,x) = x^2 \cos(\pi x), \\
&u(t,-1) = u(t,1) = -1,
\end{aligned} \tag{47}$$

with the actual diffusion coefficient $\nu = 0.001$. For PINN, we enforced the initial condition into the output by choosing a surrogate model of the solution as

$$\hat{u}(x) = (1 - \exp(-t))\,\mathrm{NN}(x;\boldsymbol{\theta}) + x^2 \cos(\pi x),$$

where $\mathrm{NN}(x;\boldsymbol{\theta})$ is a neural network.

E.3.3 Korteweg–De Vries (KdV) Equation

The Korteweg–de Vries (KdV) equation serves as a mathematical model for waves on shallow water surfaces. This equation is distinguished as a prototypical example of an integrable PDE. It is characterized by features typical of integrable systems, including a plethora of explicit solutions, notably soliton solutions, and an infinite number of conserved quantities. These properties are particularly noteworthy given the inherent nonlinearity of the equation, which generally complicates the solvability of PDEs. In specific, we consider:

$$\begin{aligned}
&u_t + uu_x + \nu u_{xxx} = 0, \\
&x \in [-1,1], t \in [0,1], \nu \neq 0, \\
&u(0,x) = \cos(\pi x),
\end{aligned} \tag{48}$$

with the actual coefficient of dispersion ν equal to 0.0025. For PINN, we enforced the initial condition into the output by choosing a surrogate model of the solution as

$$\hat{u}(x) = (1 - \exp(-t))\,\mathrm{NN}(x;\boldsymbol{\theta}) + \cos(\pi x),$$

where $\mathrm{NN}(x;\boldsymbol{\theta})$ is a neural network.

E.3.4 Numerical PDE solver

Numerical solvers play a critical role in the study and application of PDEs, enabling the simulation and analysis of complex physical phenomena that cannot be addressed analytically [51]. These solvers convert PDEs into a form that can be handled computationally, typically by discretizing the domain into a finite set of points or elements and approximating the derivatives. Conventional numerical methods include finite difference methods, finite element methods, and spectral methods [1].

Among the various numerical methods for solving PDEs, spectral methods stand out for their ability to deliver highly accurate solutions, particularly for problems with smooth solutions [21, 7]. Spectral methods involve representing the solution to a PDE as a sum of basis functions, such as trigonometric polynomials, which are globally defined over the domain. This approach contrasts with finite difference or finite element methods, where the solution is localized to the grid points or elements. In this paper, we mainly adopt spectral methods, as we focus on the Burgers, Allen-Cahn, and KdV equations. All these three equations can be efficiently and accurately resolved by spectral techniques.

F Implementation Details

F.1 Data Condensation

We conducted our experiments following the standard data condensation setting established by [68, 77, 67]. The condensation and evaluation are both performed on a depth-3 convolutional neural network [58]. The hyperparameters we used for $(FG)^2U$ are summarized in Appendix F.1. All experiments are conducted on NVIDIA-L40S (40G).

Datasets	**MNIST**			**CIFAR10**			**CIFAR100**		
IPC	1	10	50	1	10	50	1	10	50
Unrolled Depth	100	100	100	100	100	100	100	100	200
# Random Directions	32	32	32	32	32	32	32	32	32
# Inner Batch Size	Full	Full	Full	Full	Full	Full	Full	Full	100
Hessian-Free Pretraining	✗	✓	✓	✗	✓	✓	✗	✓	✓
Gradient Accumulate	16	32	32	32	32	32	64	64	64
Outer Steps	10000	10000	10000	10000	10000	10000	10000	10000	10000
Outer Step Size	1e-2	5e-4	5e-4	1e-2	5e-4	5e-4	1e-2	5e-4	5e-4
Evaluation Steps	1000	10000	10000	1000	10000	10000	1000	10000	10000

Table F.1: $(FG)^2U$ hyperparameters for data condensation experiments.

F.2 Meta Learning Online Adaptation of Language Models

We adhered to the standard settings of CaMeLS [25] and adapted their official code for our implementation. The only modification made was replacing the meta gradient approximation module with $(FG)^2U$. It is important to note that the base models used for meta-learning were initially pre-trained on a split QA-paired set. While the official codebase provided the script for pretraining, it did not include the exact base model (weights) they used. We executed the official script to generate the pre-trained base models and observed that meta-learning performance is sensitive to the choice of base models. For a fair comparison, we reported both the results from [66] (where CaMeLS [25] presented performance improvements over baselines using bar plots without specific metric values) and the results with our best custom pre-trained base models. Following the two-phase training paradigm introduced in Section 3.2, we performed training of $(FG)^2U$ on RGU (DistilGPT2, unrolled depth 6) results. The hyperparameters we used for $(FG)^2U$ are summarized in Appendix F.2, while all remaining hyperparameters were kept the same as in [25]. All experiments are conducted on one NVIDIA A100 GPU (80G).

base model	DistilGPT2	GPT2
Unrolled Depth	24/48	24/48
# of Random Directions	12	8
Gradient Accumulate	32	32
Outer Optimizer	Adam	Adam
Outer Step Size	2.5E-6	2.5E-6

Table F.2: $(FG)^2U$ hyperparameters for CaMeLS experiments.

F.3 Data-driven Discovery of Partial Differential Equations (PDEs)

The hyperparameters we used for this experiment are summarized in Appendix F.3. All experiments are conducted on NVIDIA-L40S (40G). The structure for PINN is a depth-9 and width-20 MLP with tanh activations.

G Additional Experimental Results

G.1 Meta Learning Online Adaptation of Language Models

Ablation results on the unrolled depth and the base model are summarized in Table G.1 and Table G.2.

PDEs	**Burgers**		**AllenCahn**		**KdV**	
Inner Solvers	PINN	Numerical	PINN	Numerical	PINN	Numerical
Directional Grad. Calculation	FAD	ZO	FAD	ZO	FAD	ZO
# Random Directions	1	1	1	1	1	1
Outer Steps	5000	5000	5000	5000	5000	5000
Unrolling Depth	1000	-	1000	-	1000	-
Grid Size for Numerical Method	-	256×512	-	256×512	-	256×512
Range of initial ν	(0, 1e1]	(0, 1e1]	(0, 1e-1]	(0, 1e-1]	(0, 1e-2]	(0, 1e-2]
Outer Optimizer	Adam	Adam	Adam	Adam	Adam	Adam
Outer Step Size	1e-2	1e-2	1e-2	1e-2	1e-3	1e-3
Inner Optimizer	SGD	-	SGD	-	SGD	-
Inner Batch Size	5000	-	5000	-	5000	-
Inner Step Size	1e-3	-	1e-3	-	1e-3	-
μ for Finite Difference	-	1e-4	-	1e-4	-	1e-4

Table F.3: (FG)^{2}U hyperparameters for discovery of PDEs experiments. ν denotes the unknown PDE parameters.

Model (# params)	**Method**	**Unrolled Steps (#)**	DistilGPT2		GPT2	
			EM (↑)	F1 (↑)	EM (↑)	F1 (↑)
DistilGPT2 (82M)	RGU (impl.)	6	2.04	5.53	OOM	
	(FG)^{2}U (ours)	24	2.10	5.59	**2.22**	**6.37**
		48	2.10	6.25	2.16	6.32
GPT2-Large (774M)	RGU (impl.)	6	7.02	12.19	OOM	
	(FG)^{2}U (ours)	24	6.91	12.12	7.21	**12.50**
		48	7.03	12.31	**7.27**	12.45
GPT2-XL (1.5B)	RGU (impl.)	6	7.93	12.94	OOM	
	(FG)^{2}U (ours)	24	8.34	13.46	**8.89**	**14.42**
		48	8.23	13.70	8.65	13.91

Table G.1: **StreamingQA**: Ablation results on the unrolled depth and the base model.

Model (# params)	**Method**	**Unrolled Steps (#)**	DistilGPT2		GPT2	
			EM (↑)	F1 (↑)	EM (↑)	F1 (↑)
DistilGPT2 (82M)	RGU (impl.)	6	1.52	3.16	OOM	
	(FG)^{2}U (ours)	24	1.72	3.49	1.72	**3.50**
		48	**1.75**	3.47	1.73	3.49
GPT2-Large (774M)	RGU (impl.)	6	4.86	8.57	OOM	
	(FG)^{2}U (ours)	24	5.49	8.88	**5.56**	**8.99**
		48	5.45	8.90	5.32	8.97
GPT2-XL (1.5B)	RGU (impl.)	6	6.71	9.65	OOM	
	(FG)^{2}U (ours)	24	7.00	10.13	7.27	10.33
		48	**7.37**	**10.37**	7.25	10.32

Table G.2: **SQuAD**: Ablation results on the unrolled depth and the base model.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The assumptions are included in Section 3.1, and the complete proof is placed in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: To the best of our knowledge, all the information needed to reproduce the main experimental results are included in Section 4, Appendix E, and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The data used in this paper is public. We will release the code on the acceptance of the paper.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: To the best of our knowledge, all necessary information has been included in Section 4, Appendix E, and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: We have reported the error bars in *Data Condensation* experiments and *Data-driven Discovery of PDE* experiments. For *Meta Learning Online Adaptation of LM* experiments, we adhere to the standard evaluation protocol as employed in [25, 66], specifically, using the identical evaluation script with the same random seed, to ensure a fair comparison, hence, without error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the information of computer resources in Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: The research presented in this paper focuses on fundamental algorithms rather than specific applications. Consequently, it is challenging to predict the potential social impacts of this work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not have plans to release any new data or models in conjunction with this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All usages of the related assets are accompanied by formal citations and comply with the respective licensing terms and conditions.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not have plans to release any new assets or models in conjunction with this work. The code, which will be released upon acceptance, will be well documented, and the documentation will also be made publicly available.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.