Debiasing a First-order Heuristic for Approximate Bi-level Optimization

Valerii Likhosherstov [*1] **Xingyou Song** [*2] **Krzysztof Choromanski** [2 3] **Jared Davis** [4 5] **Adrian Weller** [1 6]

Abstract

Approximate bi-level optimization (ABLO) consists of (outer-level) optimization problems, involving numerical (inner-level) optimization loops. While ABLO has many applications across deep learning, it suffers from time and memory complexity proportional to the length r of its inner optimization loop. To address this complexity, an earlier *first-order method* (FOM) was proposed as a heuristic that omits second derivative terms, yielding significant speed gains and requiring only constant memory. Despite FOM's popularity, there is a lack of theoretical understanding of its convergence properties. We contribute by theoretically characterizing FOM's gradient bias under mild assumptions. We further demonstrate a rich family of examples where FOM-based SGD does not converge to a stationary point of the ABLO objective. We address this concern by proposing an unbiased FOM (UFOM) enjoying constant memory complexity as a function of r. We characterize the introduced time-variance tradeoff, demonstrate convergence bounds, and find an optimal UFOM for a given ABLO problem. Finally, we propose an efficient adaptive UFOM scheme.

1 Introduction

Bi-level optimization (BLO) is a popular technique, which defines an outer-level objective function using the result of another (inner-level) optimization. Approximate BLO (ABLO), where the inner-level optimization is approximated by an iterative solver, has multiple applications in various subtopics of deep learning: hyperparameter optimization (Franceschi et al., 2018; Lorraine et al., 2019; Shaban et al., 2018; Maclaurin et al., 2015), neural architecture search (Liu et al., 2019), adversarial robustness (Chen et al., 2018; Zügner & Günnemann, 2019) and model-agnostic meta-learning (Finn et al., 2017; Shaban et al., 2018).

In deep learning applications of ABLO, the outer-level optimization is conducted via stochastic gradient descent (SGD), where gradients are obtained by auto-differentiation (Griewank & Walther, 2008) through the r steps of the inner gradient descent (GD). This leads to storing r intermediate inner steps, which requires a prohibitive amount of memory.

To address this, several methods for ABLO have been proposed. Among them is truncated back-propagation (Shaban et al., 2018) which stores a fixed amount of intermediate steps. Another is implicit differentiation (Pedregosa, 2016; Gould et al., 2016; Rajeswaran et al., 2019; Lorraine et al., 2019), which takes advantage of the Implicit Function Theorem, but requires multiple restrictions on the problem (Rajeswaran et al., 2019). Designing the optimizer's iteration as reversible dynamics (Maclaurin et al., 2015) allows a constant reduction in memory complexity, as only the bits lost in fixed-precision arithmetic operations are saved in memory. Forward differentiation (Franceschi et al., 2017) can be employed when outer-level gradients require a small number of parameters. Checkpointing (Griewank, 1992; Hascoet & Araya-Polo, 2006) is a generic solution for memory reduction by a factor of $\sqrt{r}$. Unbiased stochastic truncation (Beatson & Adams, 2019) can save time during stochastic optimization, but its maximum memory usage can still grow to $O(r)$. Evolution Strategies, or ES (Maheswaranathan et al., 2019), provides unbiased gradient estimates in $O(1)$ memory, but its variance depends linearly on the number of parameters, which can be prohibitively large in deep-learning applications.

One practical and simple heuristic is the *first-order method* (FOM), which omits second-order computations, and does not store inner-loop rollout states in memory. FOM is widely used in the context of neural architecture search (Liu et al., 2019), adversarial robustness (Chen et al., 2018; Zügner & Günnemann, 2019), and meta-learning (Finn et al., 2017). FOM is a limiting case of truncated back-propagation, where no states are cached in memory. We argue however, that there is a high cost to FOM's computational simplicity: FOM suffers from biased ABLO gradients and can fail to converge to a stationary point of the ABLO objective. Consequently, it would be highly desirable to

[*]Equal contribution [1]University of Cambridge [2]Google Research, Brain Team [3]Columbia University [4]Deepmind [5]Stanford University [6]The Alan Turing Institute. Correspondence to: Valerii Likhosherstov <vl304@cam.ac.uk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

modify FOM to achieve better convergence properties, without harming its $O(1)$ memory complexity.

We propose a solution: a FOM-related algorithm with unbiased gradient estimation. To achieve this, we first propose an algorithm to compute ABLO gradients in only $O(1)$ memory. Then we use this algorithm, computed "rarely" at random with a probability q, to debias FOM and yield convergent training. We call our approach *Unbiased First-Order Method* (UFOM) and highlight the following benefits:

1. UFOM has $O(1)$ memory and $O(r + qr^2)$ expected time complexity per outer-loop iteration.

2. Under mild assumptions, we show that UFOM-based SGD converges to a stationary point of the ABLO objective (Th. 1). Confirming the need for UFOM, we prove that there is a rich family of ABLO problems where **FOM ends up arbitrarily far away from a stationary point** (Th. 2).

3. We analyze the convergence rate of UFOM as a function of wall-clock time. As a result, we derive a condition on the FOM's bias $\mathbb{D}^2$ and the exact estimator's variance $\mathbb{V}^2$ that allows the optimal q to be less than one, meaning that under the constant memory restriction, **the exact ABLO has a slower convergence rate**. We then give an expression for the optimal $q^* < 1$.

4. Since the expression for the optimal q^* can be intractable in real applications, we propose a theory-inspired heuristic for choosing q^* adaptively during optimization.

We demonstrate the utility of UFOM in a synthetic experiment, data hypercleaning on MNIST (LeCun et al., 2010), and few-shot-learning on CIFAR100 (Krizhevsky et al., 2009) as well as Omniglot (Lake et al., 2011). Full proofs are provided in Appendix E in the Supplement. We provide additional related work discussions in Appendix A.

2 Preliminaries

We begin by formulating the bi-level optimization (BLO), approximate BLO problems and the first-order method.

2.1 Bi-level Optimization

We present the bi-level optimization (BLO) problem in a form which is compatible with prominent large-scale deep-learning applications. Namely, let $\Omega_{\mathcal{T}}$ be a non-empty set of *tasks* $\mathcal{T} \in \Omega_{\mathcal{T}}$ and let $p(\mathcal{T})$ be a probabilistic distribution defined on $\Omega_{\mathcal{T}}$. In practice, the procedure of sampling a task $\mathcal{T} \sim p(\mathcal{T})$ is usually resource-cheap and consists of retrieving a task from disk using either a random index (for a dataset of a fixed size), a stream, or a simulator. Define two functions as the inner and outer loss respectively: $\mathcal{L}^{in}, \mathcal{L}^{out} : \mathbb{R}^s \times \mathbb{R}^p \times \Omega_{\mathcal{T}} \to \mathbb{R}$. In addition, define $U : \mathbb{R}^s \times \Omega_{\mathcal{T}} \to \mathbb{R}^p$ so that $U(\theta, \mathcal{T})$ is a global solution of the inner (task-level) $\mathcal{L}^{in}(\theta, \cdot, \mathcal{T})$ optimization problem. Namely, for all $\theta \in \mathbb{R}^s, \mathcal{T} \in \Omega_{\mathcal{T}}$: $U(\theta, \mathcal{T}) = \arg\min_{\phi \in \mathbb{R}^p} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$. Then, the *bi-level optimization* is defined as an optimization over θ to find the optimal average value of $\mathcal{L}^{out}(\theta, U(\theta, \mathcal{T}), \mathcal{T})$:

$$\min_{\theta} \left(\mathcal{M}(\theta) = \mathbb{E}_{p(\mathcal{T})} \left[\mathcal{L}^{out}(\theta, U(\theta, \mathcal{T}), \mathcal{T}) \right] \right). \quad (1)$$

In practice, however, the global optimum $U(\theta, \mathcal{T})$ is usually intractable and needs to be approximated with a few gradient descent (GD) iterations:

$$\forall r \in \mathbb{N} : U^{(r)}(\theta, \mathcal{T}) = \phi_r, \quad \phi_0 = V(\theta, \mathcal{T}), \quad (2)$$

$$\forall 1 \leq j \leq r : \phi_j = \phi_{j-1} - \alpha_j \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}), \quad (3)$$

where further in the text, $\frac{\partial}{\partial \square}$ denotes a partial derivative of a function with respect to the corresponding argument, while $\nabla_{\square}$ denotes a full gradient. $V : \mathbb{R}^s \times \Omega_{\mathcal{T}} \to \mathbb{R}^p$ defines *a starting point* of the GD, r is a number of GD iterations and $\{\alpha_j > 0\}_{j=1}^{\infty}$ is a sequence of step sizes. Then the *approximate BLO (ABLO) with r GD steps* is defined as finding an optimal average $\mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$:

$$\min_{\theta} \left(\mathcal{M}^{(r)}(\theta) = \mathbb{E}_{p(\mathcal{T})} \left[\mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T}) \right] \right). \quad (4)$$

The formulations (2-4) unify various application scenarios. We can think of ϕ as model parameters and (3) as a training loop, while θ are hyperparameters optimized in the outer loop – a scenario matching the hyperparameter optimization paradigm (Franceschi et al., 2018; Lorraine et al., 2019; Shaban et al., 2018; Maclaurin et al., 2015). Alternatively, θ may act as parameters encoding a neural-network's topology which corresponds to differentiable neural architecture search (Liu et al., 2019). Here $\mathcal{T}$ encodes a minibatch drawn from a dataset. The adversarial robustness problem fits into (3-4) by assuming that θ are learned model parameters and ϕ is an input perturbation optimized through (3) to decrease the model's performance. *Model-Agnostic Meta-Learning* (MAML) (Finn et al., 2017) is an approach to few-shot learning (adaptation to a few training examples) by learning a generalizable initialization $\phi_0 = V(\theta, \mathcal{T}) = \theta$, where $\mathcal{L}^{in}$ and $\mathcal{L}^{out}$ correspond to an empirical risk on a small set of train and validation shots respectively.

We consider stochastic gradient descent (SGD) (Bottou et al., 2018) as a solver for (4) – see Algorithm 1, where $\mathcal{G}(\theta, \mathcal{T})$ is the exact gradient $\nabla_{\theta} \mathcal{L}^{out}(U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$ or its approximation and $\{\gamma_k > 0\}_{k=1}^{\infty}$ is a sequence of outer-loop step sizes satisfying

$$\sum_{k=1}^{\infty} \gamma_k = \infty, \quad \lim_{k \to \infty} \gamma_k = 0. \quad (5)$$

Algorithm 1 Outer SGD.

Input: $\theta_0 \in \mathbb{R}^s$, # iterations τ.
for $k = 1$ **to** τ **do**
  Draw $\mathcal{T} \sim p(\mathcal{T})$;
  Set $\theta_k := \theta_{k-1} - \gamma_k \mathcal{G}(\theta_{k-1}, \mathcal{T})$;
end for

Assuming that the procedure for sampling tasks $\mathcal{T} \sim p(\mathcal{T})$ takes negligible resources, the time and memory requirement of Algorithm 1 is dominated by the part spent for computing $\mathcal{G}(\theta, \mathcal{T})$. Therefore, in our subsequent derivations, we analyse the computational complexity of finding the gradient estimate $\mathcal{G}(\theta, \mathcal{T})$.

2.2 Exact ABLO gradients

Outer SGD requires the computation or approximation of a gradient $\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$. We apply the full derivative rule with respect to θ to the inner GD (3) and deduce that

$$\nabla_\theta \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}) = \frac{\partial}{\partial \theta} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}) + \sum_{j=1}^{r} D_j \quad (6)$$

$$+ \left(\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}) \right)^\top \nabla_{\phi_0} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}),$$

$$D_j = \left(\frac{\partial}{\partial \theta} (\phi_{j-1} - \alpha_j \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T})) \right)^\top \times \nabla_{\phi_j} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T})$$

$$= -\alpha_j \left(\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}) \right)^\top \nabla_{\phi_j} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}), \quad (7)$$

where $\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}) \in \mathbb{R}^{s \times p}$ is a second-order partial derivative matrix. Next, we observe that

$$\nabla_{\phi_r} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}) = \frac{\partial}{\partial \phi} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T})$$

and apply the chain rule to a sequence of states $\phi_1, \ldots, \phi_r$ to deduce that for all $1 \leq j \leq r$,

$$\nabla_{\phi_{j-1}} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}) = \left(\frac{\partial}{\partial \phi_{j-1}} (\phi_{j-1} \right. \quad (8)$$

$$\left. - \alpha_j \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T})) \right)^\top \nabla_{\phi_j} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T})$$

$$= \left(I - \alpha_j \frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}) \right)^\top \nabla_{\phi_j} \mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}), \quad (9)$$

where $\frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi_{r-1}, \mathcal{T}) \in \mathbb{R}^{p \times p}$ is a second-order partial derivative matrix. Based on (6-9), Algorithm 2 illustrates inner loop for the exact ABLO gradient, which, together with Algorithm 1, outlines the training procedure.

Hessian-vector products $(\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$ and $(\frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$ can be computed efficiently at

Algorithm 2 Inner GD (exact).

Input: $\theta \in \mathbb{R}^s$, $r \in \mathbb{N}$, $\mathcal{T}$.
Output: $\mathcal{G}(\theta, \mathcal{T}) = \nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$.
New array Arr$[1..r]$;
Set $\phi := V(\theta, \mathcal{T})$;
for $j = 1$ **to** r **do**
  Set Arr$[j] := \phi$;
  $\phi := \phi - \alpha_j \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$;
end for
Set $b_1 := \frac{\partial}{\partial \theta} \mathcal{L}^{out}(\theta, \phi, \mathcal{T}), b_2 := \frac{\partial}{\partial \phi} \mathcal{L}^{out}(\theta, \phi, \mathcal{T})$;
for $j = r$ **to** 1 **do**
  Set $\phi :=$ Arr$[j]$;
  Set $b_1 := b_1 - \alpha_j (\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$;
  Set $b_2 := b_2 - \alpha_j (\frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$;
end for
return $b_1 + (\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}))^\top b_2$;

Algorithm 3 Inner GD (FOM).

Input: $\theta \in \mathbb{R}^s$, $r \in \mathbb{N}$, $\mathcal{T}$.
Output: $\mathcal{G}_{FO}(\theta, \mathcal{T})$.
Set $\phi := V(\theta, \mathcal{T})$;
for $j = 1$ **to** r **do**
  $\phi := \phi - \alpha_j \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$;
end for
Set $b_1 := \frac{\partial}{\partial \theta} \mathcal{L}^{out}(\theta, \phi, \mathcal{T}), b_2 := \frac{\partial}{\partial \phi} \mathcal{L}^{out}(\theta, \phi, \mathcal{T})$;
return $b_1 + (\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}))^\top b_2$;

once using the reverse accumulation technique (Christianson, 1992) without explicitly constructing Hessian matrices $\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$ and $\frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$. This technique consists of evaluation and automatic differentiation (Griewank & Walther, 2008) of a functional $H(\theta, \phi) : \mathbb{R}^s \times \mathbb{R}^p \to \mathbb{R}$, $H(\theta, \phi) = \frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})^\top b_2$, where $\frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$ is produced by another automatic differentiation. The result of differentiation is precisely $(\frac{\partial^2}{\partial \theta \partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$ and $(\frac{\partial^2}{\partial \phi^2} \mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$. Similarly, the Jacobian-vector product $(\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}))^\top b_2$ can be computed by auto-differentiating through $V(\theta, \mathcal{T})^\top b_2$. According to *the Cheap Gradient Principle* (Griewank & Walther, 2008), the time complexities of the Hessian-vector product and the Jacobian-vector product are the same as the time complexities respectively, of computations $\frac{\partial}{\partial \phi} \mathcal{L}^{in}(\theta, \phi, \mathcal{T})$ and $V(\theta, \mathcal{T})$.

2.3 First-order heuristic for ABLO

One limitation, which can significantly complicate application of ABLO in real-life scenarios when both the number of parameters p and the number of gradient descent iterations r are large, is the requirement to allocate an array Arr$[1..r]$ for all intermediate states $\phi_1, \ldots, \phi_r \in \mathbb{R}^p$ (Algorithm 2).

A simple improvement which sometimes performs well in practice is the *first-order method* (FOM) (Finn et al., 2017; Liu et al., 2019; Chen et al., 2018; Zügner & Günnemann, 2019); this proposes to approximate $\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$ with

$$\mathcal{G}_{FO}(\theta, \mathcal{T}) = \frac{\partial}{\partial \theta}\mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}) + \left(\frac{\partial}{\partial \theta} V(\theta, \mathcal{T})\right)^\top \quad (10)$$

$$\times \frac{\partial}{\partial \phi}\mathcal{L}^{out}(\theta, \phi_r, \mathcal{T}), \quad \phi_r = U^{(r)}(\theta, \mathcal{T}), \quad (11)$$

which corresponds to zeroing out Hessians $(\frac{\partial^2}{\partial \theta \partial \phi}\mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}))^\top b_2$ and $(\frac{\partial^2}{\partial \phi^2}\mathcal{L}^{in}(\theta, \phi_{j-1}, \mathcal{T}))^\top b_2$ in Equations (6-9). FOM (Algorithm 3) does not store any intermediate states, giving $O(1)$ memory complexity and $O(r)$ time complexity. Although FOM enjoys better memory complexity, it can fail to converge to a stationary point of the ABLO objective (4), see Section 4. In the next section we propose a solution – modified FOM with the desired convergence properties due to unbiased gradients.

3 Unbiased First-Order Method (UFOM)

An alternative, memory-efficient method for finding $\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$ is to recompute each ϕ_j from scratch instead of retrieving it from the array Arr during the backward loop of Algorithm 2. However, this would result in $O(r^2)$ complexity, as re-computation requires an $O(r)$-long nested loop inside each $j = r, \dots, 1$ iteration. Therefore, we propose an unbiased stochastic combination of this slow but memory-efficient exact algorithm, and a fast and cheap, but biased FOM-based estimator.

Let $\xi \in \{0, 1\}$ be a Bernoulli random variable with $\mathbb{P}(\xi = 1) = q$, which we write as $\xi \sim \text{Bernoulli}(q)$, where $q \in (0, 1]$. Recall that $\mathcal{G}_{FO}$ is the first-order gradient from Algorithm 3. We consider the following stochastic approximation to $\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$:

$$\mathcal{G}_{UFO}(\theta, \mathcal{T}) = \mathcal{G}_{FO}(\theta, \mathcal{T}) \quad (12)$$

$$+\frac{\xi}{q}(\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T}) - \mathcal{G}_{FO}(\theta, \mathcal{T})). \quad (13)$$

In fact, (12-13) is an unbiased estimate of $\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T})$. Indeed, since $\mathbb{E}_\xi[\xi] = q$:

$$\mathbb{E}_\xi[\mathcal{G}_{UFO}(\theta, \mathcal{T})] = (1 - \frac{q}{q})\mathcal{G}_{FO}(\theta, \mathcal{T})$$

$$+\frac{q}{q}\nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T}) = \nabla_\theta \mathcal{L}^{out}(\theta, U^{(r)}(\theta, \mathcal{T}), \mathcal{T}).$$

For this reason we call the estimate (12-13) an *unbiased first-order method* (UFOM). Algorithm 4 illustrates randomized computation of UFOM. It combines FOM and memory-efficient computation of the exact gradient. Hence, UFOM has constant memory complexity and $O(r + qr^2)$ expected time complexity:

Algorithm 4 Inner GD (UFOM).

Input: $\theta \in \mathbb{R}^s$, $r \in \mathbb{N}$, $\mathcal{T}$, $q \in (0, 1]$.
Output: $\mathcal{G}_{UFO}(\theta, \mathcal{T})$.
Set $\phi := V(\theta, \mathcal{T})$;
for $j = 1$ **to** r **do**
  $\phi := \phi - \alpha_j \frac{\partial}{\partial \phi}\mathcal{L}^{in}(\theta, \phi, \mathcal{T})$;
end for
Set $b_1 := \frac{\partial}{\partial \theta}\mathcal{L}^{out}(\theta, \phi, \mathcal{T}), b_2 := \frac{\partial}{\partial \phi}\mathcal{L}^{out}(\theta, \phi, \mathcal{T})$;
Set $b_{FO} := b_1 + (\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}))^\top b_2$;
Draw $\xi \sim \text{Bernoulli}(q)$;
if $\xi = 0$ **then return** b_{FO} **end if**
for $j = r$ **to** 1 **do**
  Set $\phi = V(\theta, \mathcal{T})$;
  for $j_1 = 1$ **to** $j - 1$ **do**
    Set $\phi := \phi - \alpha_j \frac{\partial}{\partial \phi}\mathcal{L}^{in}(\theta, \phi, \mathcal{T})$;
  end for
  Set $b_1 := b_1 - \alpha_j (\frac{\partial^2}{\partial \theta \partial \phi}\mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$;
  Set $b_2 := b_2 - \alpha_j (\frac{\partial^2}{\partial \phi^2}\mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2$;
end for
Set $b_{Exact} = b_1 + (\frac{\partial}{\partial \theta} V(\theta, \mathcal{T}))^\top b_2$;
return $b_{FO} + \frac{1}{q}(b_{Exact} - b_{FO})$;

Proposition 1. *Algorithm 4 requires* $r + 1 + qr(r-1)/2$ *expected evaluations of the gradient* $\frac{\partial}{\partial \phi}\mathcal{L}^{\square}(\theta, \phi, \mathcal{T})$, $\square \in \{in, out\}$ *and* qr *expected evaluations of the Hessian-vector product* $\left[(\frac{\partial^2}{\partial \theta \partial \phi}\mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2, (\frac{\partial^2}{\partial \phi^2}\mathcal{L}^{in}(\theta, \phi, \mathcal{T}))^\top b_2\right]$.

Observe that, when $q = 1$, Algorithm 4 is reduced to the *memory-efficient exact gradient estimator*, discussed above.

4 Theoretical Results

We perform a theoretical analysis of the FOM and UFOM. First, we establish assumptions on the problem. Then, we characterize bias of the FOM approximation and variance of the UFOM. Next, we prove the convergence rate bounds for UFOM and show divergence of the FOM. Finally, we use these obtained bounds to select the optimal value of q and propose a heuristical adaptive scheme.

4.1 Assumptions

We first formulate Assumptions 1 and 2 used for proofs. The first assumption requires uniformly bounded, uniformly Lipschitz-continuous gradients and Hessians of $\mathcal{L}^{in}$, $\mathcal{L}^{out}$.

Assumption 1. *For any* $\mathcal{T} \in \Omega_{\mathcal{T}}$, $\mathcal{L}^{in}(\theta, \phi, \mathcal{T})$, $\mathcal{L}^{out}(\theta, \phi, \mathcal{T})$ *are twice differentiable as functions of* θ, ϕ *and* $V(\theta, \mathcal{T})$ *is differentiable as a function of* θ. *There exist* $M_1, M_2, L_1, L_2, L_3 > 0$ *such that for any* $\mathcal{T} \in \Omega_{\mathcal{T}}, \theta', \theta'' \in \mathbb{R}^s, \phi', \phi'' \in \mathbb{R}^p, \square \in \{in, out\}$

$$\|\frac{\partial}{\partial \theta} V(\theta', \mathcal{T})\|_2 \leq M_1,$$

$$\|\frac{\partial}{\partial\theta}V(\theta',\mathcal{T})-\frac{\partial}{\partial\theta}V(\theta'',\mathcal{T})\|_2\le M_2\|\theta'-\theta''\|_2,$$
$$\max(\|\frac{\partial}{\partial\theta}\mathcal{L}^{\square}(\theta',\phi',\mathcal{T})\|_2,\|\frac{\partial}{\partial\phi}\mathcal{L}^{\square}(\theta',\phi',\mathcal{T})\|_2)\le L_1,$$
$$\max(\|\frac{\partial}{\partial\theta}\mathcal{L}^{\square}(\theta',\phi',\mathcal{T})-\frac{\partial}{\partial\theta}\mathcal{L}^{\square}(\theta'',\phi'',\mathcal{T})\|_2,$$
$$\|\frac{\partial}{\partial\phi}\mathcal{L}^{\square}(\theta',\phi',\mathcal{T})-\frac{\partial}{\partial\phi}\mathcal{L}^{\square}(\theta'',\phi'',\mathcal{T})\|_2)$$
$$\le L_2\|\theta'-\theta''\|_2+L_2\|\phi'-\phi''\|_2,$$
$$\max(\|\frac{\partial^2}{\partial\theta\partial\phi}\mathcal{L}^{in}(\theta',\phi',\mathcal{T})-\frac{\partial^2}{\partial\theta\partial\phi}\mathcal{L}^{in}(\theta'',\phi'',\mathcal{T})\|_2,$$
$$\|\frac{\partial^2}{\partial\phi^2}\mathcal{L}^{in}(\theta',\phi',\mathcal{T})-\frac{\partial^2}{\partial\phi^2}\mathcal{L}^{in}(\theta'',\phi'',\mathcal{T})\|_2)$$
$$\le L_3\|\theta'-\theta''\|_2+L_3\|\phi'-\phi''\|_2.$$

The next assumption is satisfied in particular when $p(\mathcal{T})$ is defined on a finite set of tasks $\mathcal{T}$ and $\mathcal{L}^{out}$ is lower-bounded.

Assumption 2 (Regularity of $\mathcal{M}^{(r)}$)**.** *For each $\theta\in\mathbb{R}^s$, the terms $\mathcal{M}^{(r)}(\theta)$, $\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta)$, $\mathbb{E}_{p(\mathcal{T})}\left[\nabla_\theta\mathcal{L}^{out}(\theta,U^{(r)}(\theta,\mathcal{T}),\mathcal{T})\right]$ are well-defined and $\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta)=\mathbb{E}_{p(\mathcal{T})}\left[\nabla_\theta\mathcal{L}^{out}(\theta,U^{(r)}(\theta,\mathcal{T}),\mathcal{T})\right]$. Let $\mathcal{M}^{(r)}_*=\inf_{\theta\in\mathbb{R}^s}\mathcal{M}^{(r)}(\theta)$, then $\mathcal{M}^{(r)}_*>-\infty$.*

4.2 Bias of FOM and Variance of UFOM

FOM provides a cheap but biased estimate of the exact gradient $\nabla_\theta\mathcal{L}^{out}$. To characterize this bias, and its effect on the convergence of UFOM, we define $\mathbb{D}^2$ as:

$$\sup_{\theta\in\mathbb{R}^s}\mathbb{E}_{\mathcal{T}}[\|\mathcal{G}_{FO}(\theta,\mathcal{T})-\nabla_\theta\mathcal{L}^{out}(\theta,U^{(r)}(\theta,\mathcal{T}),\mathcal{T})\|_2^2]. \quad (14)$$

Intuitively, $\mathbb{D}^2$ is the maximal average bias over all θ values. Under Assumptions 1 and 2, Lemma 1 in Appendix E establishes the bound $\mathbb{D}^2\le\mathbb{D}^2_{bound}$, where

$$\mathbb{D}_{bound}=(1+M_1)L_1L_2\sum_{j=1}^{r}\alpha_j\prod_{j'=j}^{r}(1+\alpha_{j'}L_2). \quad (15)$$

The ability to derive the bound (15) and memory-efficiency **justify the use of FOM gradient proxies in the debiasing scheme** (Algorithm 4), as opposed to any other gradient approximations. Intuitively, when $\mathbb{D}^2$ is small – or sufficiently, its tractable proxy $\mathbb{D}^2_{bound}$ is small – the variance of the UFOM-based gradient estimate is reduced. This agrees with our theoretical findings: in Lemma 1 we show that

$$\sup_{\theta\in\mathbb{R}^s}\mathbb{E}_{p(\mathcal{T})}\left[\|\mathcal{G}_{UFO}(\theta,\mathcal{T})\|_2^2\right]\le(\frac{1}{q}-1)\mathbb{D}^2+\mathbb{V}^2,$$

where the left-hand side characterizes the variance of UFOM and $\mathbb{V}^2$ is the maximal variance of the exact ABLO gradient:

$$\mathbb{V}^2=\sup_{\theta\in\mathbb{R}^s}\mathbb{E}_{p(\mathcal{T})}\left[\|\nabla_\theta\mathcal{L}^{out}(\theta,U^{(r)}(\theta,\mathcal{T}),\mathcal{T})\|_2^2\right]. \quad (16)$$

The upper bound on the variance of UFOM approaches $\mathbb{V}^2$ if either $\mathbb{D}^2$ approaches 0, or q approaches 1.

4.3 Convergence of UFOM and Divergence of FOM

We analyze FOM and UFOM as algorithms which seek a stationary point of the ABLO objective (4), i.e. a point $\theta^*\in\mathbb{R}^s$ such that $\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta^*)=\mathbf{0}_s$ (a vector of s zeros). Motivated by searching for θ^*, we prove a standard result for stochastic optimization of nonconvex functions (Bottou et al., 2018, Section 4.3):

$$\liminf_{k\to\infty}\mathbb{E}\left[\|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_k)\|_2^2\right]=0, \quad (17)$$

where θ_k are iterates of SGD with UFOM gradient estimation. Intuitively, equation (17) implies that there exist iterates of UFOM which approach some stationary point θ^* up to any level of proximity. Proofs are in Appendix E.

Theorem 1 (Convergence of UFOM)**.** *Let $p,r,s\in\mathbb{N}$, $\{\alpha_j>0\}_{j=1}^\infty$ and $\{\gamma_k>0\}_{k=1}^\infty$ be any sequences, $q\in(0,1]$, $\theta_0\in\mathbb{R}^s$, $p(\mathcal{T})$ be a distribution on a nonempty set $\Omega_\mathcal{T}$, $V:\mathbb{R}^s\times\Omega_\mathcal{T}\to\mathbb{R}^p$, $\mathcal{L}^{in},\mathcal{L}^{out}:\mathbb{R}^s\times\mathbb{R}^p\times\Omega_\mathcal{T}\to\mathbb{R}$ be functions satisfying Assumption 1, and let $U^{(r)}:\mathbb{R}^s\times\Omega_\mathcal{T}\to\mathbb{R}^p$ be defined according to (2-3), $\mathcal{M}^{(r)}:\mathbb{R}^p\to\mathbb{R}$ be defined according to (4) and satisfy Assumption 2. Define $\mathcal{G}_{FO}:\mathbb{R}^s\times\Omega_\mathcal{T}\to\mathbb{R}^s$ as in (10-11) and $\mathcal{G}:\mathbb{R}^s\times\Omega_\mathcal{T}\times\{0,1\}\to\mathbb{R}^s$ as*

$$\mathcal{G}(\theta,\mathcal{T},x)=\mathcal{G}_{FO}(\theta,\mathcal{T})+\frac{x}{q}(\nabla_\theta\mathcal{L}^{out}(\theta,U^{(r)}(\theta,\mathcal{T}),\mathcal{T})-\mathcal{G}_{FO}(\theta,\mathcal{T})).$$

Let $\{\mathcal{T}_k\}_{k=1}^\infty,\{\xi_k\}_{k=1}^\infty$ be sequences of i.i.d. samples from $p(\mathcal{T})$ and $\mathrm{Bernoulli}(q)$ respectively, such that σ-algebras populated by both sequences are independent. Let $\{\theta_k\in\mathbb{R}^s\}_{k=0}^\infty$ be a sequence where for all $k\in\mathbb{N}$ $\theta_k=\theta_{k-1}-\gamma_k\mathcal{G}(\theta_{k-1},\mathcal{T}_k,\xi_k)$. Then it holds that

1) if $\{\gamma_k\}_{k=1}^\infty$ satisfies (5) and $\sum_{k=1}^\infty\gamma_k^2<\infty$, then $\liminf_{k\to\infty}\mathbb{E}\left[\|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_k)\|_2^2\right]=0$;

2) if $\forall k\in\mathbb{N}:\gamma_k=k^{-0.5}$, then

$$\min_{0\le u<k}\mathbb{E}\left[\|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_u)\|_2^2\right]=\left((1/q-1)\mathbb{D}^2+\mathbb{V}^2\right)\cdot O(k^{-0.5+\epsilon}), \quad (18)$$

where ϵ is any positive number and the constant, hidden in $O(k^{-0.5+\epsilon})$, does not depend on q. $\mathbb{D}^2,\mathbb{V}^2$ are defined by (14), (16) respectively.

Our second contribution is a proof that Equation (17) does not hold for FOM under the same assumptions. More specifically, we show that for any $D>0$, there exists a problem of type (4) such that $\liminf_{k\to\infty}\mathbb{E}\left[\|\nabla_{\theta_k}\mathcal{M}^{(r)}(\theta_k)\|_2^2\right]>D$

where $\{\theta_k\}$ are iterates of FOM. The intuition behind this result is that **FOM cannot find a solution with gradient norm lower than** D.

Theorem 2 (Divergence of FOM)**.** *Let $p = s, r \in \mathbb{N}, \alpha > 0, \theta_0 \in \mathbb{R}^s$, $\{\gamma_k > 0\}_{k=1}^{\infty}$ be any sequence satisfying (5), and D be any positive number. Then there exists a set $\Omega_{\mathcal{T}}$ with a distribution $p(\mathcal{T})$ on it and functions $V : \mathbb{R}^s \times \Omega_{\mathcal{T}} \to \mathbb{R}^p, \mathcal{L}^{in}, \mathcal{L}^{out} : \mathbb{R}^s \times \mathbb{R}^p \times \Omega_{\mathcal{D}} \to \mathbb{R}$ satisfying Assumption 1, such that for $U^{(r)} : \mathbb{R}^p \times \mathcal{T} \to \mathbb{R}^p$ defined according to (2-3), where $\forall j : \alpha_j = \alpha$, $\mathcal{M}^{(r)} : \mathbb{R}^p \to \mathbb{R}$ defined according to (4) and satisfying Assumption 2, the following holds: define $\mathcal{G}_{FO} : \mathbb{R}^s \times \Omega_{\mathcal{T}} \to \mathbb{R}^s$ as in (10-11). Let $\{\mathcal{T}_k\}_{k=1}^{\infty}$ be a sequence of i.i.d. samples from $p(\mathcal{T})$. Let $\{\theta_k \in \mathbb{R}^p\}_{k=0}^{\infty}$ be a sequence where for all $k \in \mathbb{N}$, $\theta_k = \theta_{k-1} - \gamma_k \mathcal{G}_{FO}(\theta_{k-1}, \mathcal{T}_k)$. Then $\liminf_{k\to\infty} \mathbb{E}\left[\|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_k)\|_2^2\right] > D$.*

4.4 Optimal Choice of q

Note that as a special case of Theorem 1, we obtain a convergence proof for ABLO with exact gradients ($q = 1$). The case $q \to 0$, on the other hand, corresponds to FOM without corrections. We analyze the bound (18) in order to develop an intuition about the choice of q, leading to the fastest convergence as a function of wall-clock time. Let C_1, C_2 denote the time required to compute $\frac{\partial}{\partial\phi}\mathcal{L}^{\square}(\theta, \phi, \mathcal{T})$, $\square \in \{in, out\}$ and $\left[(\frac{\partial^2}{\partial\theta\partial\phi}\mathcal{L}^{in}(\theta,\phi,\mathcal{T}))^\top b_2, (\frac{\partial^2}{\partial\phi^2}\mathcal{L}^{in}(\theta,\phi,\mathcal{T}))^\top b_2\right]$ respectively. Fix $0 < \epsilon$. The smaller is ϵ, the tighter is the bound (18), so in addition we assume that $\epsilon < 0.5$. Let $\mathcal{H}(\epsilon)$ be a hidden constant in O-notation of (18). Then for δ small enough and , it holds that one needs

$$\left(\mathcal{H}(\epsilon)^{-1} \cdot \delta \cdot ((1/q-1)\mathbb{D}^2 + \mathbb{V}^2)^{-1}\right)^{\frac{1}{-0.5+\epsilon}}$$

iterations to achieve $\min_{0\le u<k} \mathbb{E}\left[\|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_u)\|_2^2\right] \le \delta$. According to Proposition 1, the expected amount of time to get such precision is proportional to

$$\left((1/q-1)\mathbb{D}^2 + \mathbb{V}^2\right)^{\frac{2}{1-2\epsilon}} (C_{det} + C_{rnd} \cdot q), \qquad (19)$$

where $C_{det} = C_1(r+1), C_{rnd} = (C_1(r-1)/2 + C_2)r$, and we ignore the negligible time needed for the vector sums and scalar products in Algorithm 4. The convergence estimate as a function of time is tighter for UFOM than for the exact memory-efficient gradient, in the case when the q^*, minimizing (19), is smaller than 1. In Appendix F, we show that **this is equivalent to the condition**

$$\mathbb{D}^2 < \frac{C_{rnd}}{\frac{2}{1-2\epsilon}(C_{det} + C_{rnd})}\mathbb{V}^2. \qquad (20)$$

We also prove that in case of (20), the optimum q^* lies in $(0, 1)$ as the root of the quadratic equation

$$C_{rnd}(\mathbb{V}^2 - \mathbb{D}^2)q^2 + \frac{2\epsilon+1}{2\epsilon-1}\mathbb{D}^2 C_{rnd} q + \frac{2}{2\epsilon-1}\mathbb{D}^2 C_{det} = 0. \qquad (21)$$

4.5 Adaptive UFOM

For simplicity, in Assumption 1 we require global bounds M_1, M_2, L_1, L_2, L_3 on norms and Lipschitz constants of $V, \mathcal{L}^{in}, \mathcal{L}^{out}$ gradients. However, it can be deduced from the proof of Theorem 1, that this assumption can be relaxed by only assuming bounded and Lipschitz-continuous gradients in the open set containing the trajectory $[\theta_0, \theta_1] \cup [\theta_1, \theta_2] \cup \dots$ of SGD. Similarly, (14,16) can be relaxed by taking supremum over this open set rather than the whole $\mathbb{R}^s$. With this observation in mind, we propose a heuristic algorithm for choosing q adaptively during training (*Adaptive UFOM*). According to this algorithm, we maintain and update approximate estimates $\overline{\mathbb{D}_k^2}, \overline{\mathbb{V}_k^2}$ of $\mathbb{D}^2, \mathbb{V}^2$ respectively, and, during each iteration of SGD, compute q^* as described in Section 4.4 by substituting these approximations. C_1, C_2, ϵ are assumed to be known and set by the user. We define

$$\overline{\mathbb{D}_k^2} = \mathbb{D}_{sm,k}^2/(1-\beta^{k_{upd}}), \overline{\mathbb{V}_k^2} = \mathbb{V}_{sm,k}^2/(1-\beta^{k_{upd}}), \quad (22)$$

where $0 < \beta < 1$ is a user-defined exponential smoothing constant and $\mathbb{D}_{sm,k}^2, \mathbb{V}_{sm,k}^2$ are exponentially smoothed $\|b_{FO} - b_{Exact}\|_2^2, \|b_{Exact}\|_2^2$ from Algorithm 4. That is, initially $\mathbb{D}_{sm,0}^2 = \mathbb{V}_{sm,0}^2 = 0$. During the kth step of SGD, if $\xi = 0$ in Algorithm 4, then $\mathbb{D}_{sm,k}^2 = \mathbb{D}_{sm,k-1}^2, \mathbb{V}_{sm,k}^2 = \mathbb{V}_{sm,k-1}^2$. If $\xi = 1$, then we use the update rule

$$\mathbb{D}_{sm,k}^2 = \beta\mathbb{D}_{sm,k-1}^2 + (1-\beta)\|b_{FO} - b_{Exact}\|_2^2, \quad (23)$$

$$\mathbb{V}_{sm,k}^2 = \beta\mathbb{V}_{sm,k-1}^2 + (1-\beta)\|b_{Exact}\|_2^2. \quad (24)$$

k_{upd} is the number of updates (23-24) performed before the kth iteration. This way, $\overline{\mathbb{D}_k^2}, \overline{\mathbb{V}_k^2}$ resemble "local" estimates of (14,16) at the current region of SGD trajectory.

To perform updates (23-24) frequently enough, we set $q = \max(q^*, q_{min})$, where q_{min} is a small constant to guarantee that ξ is nonzero at least sometimes when running Algorithm 4. The proposed Adaptive UFOM doesn't result in any additional evaluations of $\mathcal{L}^{in}, \mathcal{L}^{out}$ or its first or second derivatives. As for the additional memory, Adaptive UFOM only requires to store and update two values $\mathbb{D}_{sm,k}^2, \mathbb{V}_{sm,k}^2$.

5 Experiments

We illustrate our theoretical findings on a synthetic experiment and then evaluate Adaptive UFOM on data hypercleaning and few-shot learning. As baselines, we compare with the memory-efficient exact estimator (equivalent to UFOM with $q = 1$) and FOM (equivalent to $q \to 0$). We demonstrate that, under the same memory constraints, UFOM can significantly speed up convergence of the exact estimator. As a measure of algorithm speed, we use the total number

of gradient and Hessian-vector product calls, which we refer to as *function calls*. This is equivalent to the "wall-clock time" from Section F, if we set $C_1 = C_2 = 1$. Since ϵ is any small positive number, we further set it to 0 for simplicity.

5.1 Synthetic Experiment

Theorem 2 is proven by explicitly constructing the following counterexample: $\Omega_{\mathcal{T}} = \{\mathcal{T}^{(1)}, \mathcal{T}^{(2)}\}$ where both tasks $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}$ are equally likely under $p(\mathcal{T})$, $V(\theta, \mathcal{T}) = \theta$ and functions $\mathcal{L}^{in}(\theta, \phi, \mathcal{T}^{(i)}) = \mathcal{L}^{out}(\theta, \phi, \mathcal{T}^{(i)})$ are independent of θ and are convex piecewise-polynomials of ϕ for (see Appendix E for details). Inner-GD step sizes are the same ($\forall j : \alpha_j = \alpha$). As a demonstration of our theoretical findings in a simple experimental setup, we simulate via exact gradients, both FOM and (non-adaptive) UFOM for this synthetic ABLO formulation. We opt for a single parameter ($s = p = 1$) and inner-GD length of $r = 10$.

Figures 1a, 1b, 1c demonstrate the result of Theorems 1 and 2, illustrating an example problem where UFOM ($q = 0.1$) converges to a stationary point, while FOM does not. In Figure 1d, we vary the α parameter and illustrate how approximate values of $\mathbb{V}^2, \mathbb{D}^2$ change. On Figure 1e, we compare a theoretical estimate of q^* obtained by solving (21), and an empirical estimate of q^* obtained by grid search. We observe that our theoretical findings roughly agree with the experiment. Finally, on Figure 1f, we illustrate the case where the theoretically optimal q^* is less than 1. We show that, indeed, optimization is faster when $q = q^* \approx 0.08$ than when $q = 1$ (exact memory-efficient gradient computation). More details can be found in Appendix B.

5.2 Data Hypercleaning

We evaluate Adaptive UFOM in a hyperparameter optimization problem on MNIST, based on the setup from (Shaban et al., 2018; Franceschi et al., 2017). The task is to train a classifier $g(\phi, X_i) \in \mathbb{R}^{10}, X_i \in \mathbb{R}^{784}$, parametrized by ϕ, on a subset of 5000 labeled images, where 2500 labels Y_i have been corrupted. For that, we define $\theta \in \mathbb{R}^{5000}$, $|\Omega_{\mathcal{T}}| = 1$ and the inner loss $\mathcal{L}^{in}$ has the form $\mathcal{L}^{in}(\theta, \phi, \mathcal{T}) = \sum_{i=1}^{5000} \sigma(\theta^{(i)}) l_{CCE}(g(\phi, X_i), Y_i)$, where $\sigma(\cdot)$ is a sigmoid function, $\theta^{(i)}$ is the ith element of θ and $l_{CCE}(\cdot, Y)$ is a categorical cross entropy (CCE) with respect to a label $Y_i \in \{0, \dots, 9\}$. $\mathcal{L}^{out}$ is defined as a cross entropy on the validation set. As in (Shaban et al., 2018), we set $r = 100$ and $\alpha = 1$. This way, the inner loop trains classifier $g(\phi, X)$, while the outer loop is optimizing weights of each training object. Presumably, $\sigma(\theta^{(i)})$ should assign the bigger weight to uncorrupted examples. We modify a setup of (Shaban et al., 2018) by using a two- instead of one-layer feedforward network, with ReLU nonlinearity, to obtain a nonconvex optimization problem. For Adaptive UFOM, on a validation score comparison we find that

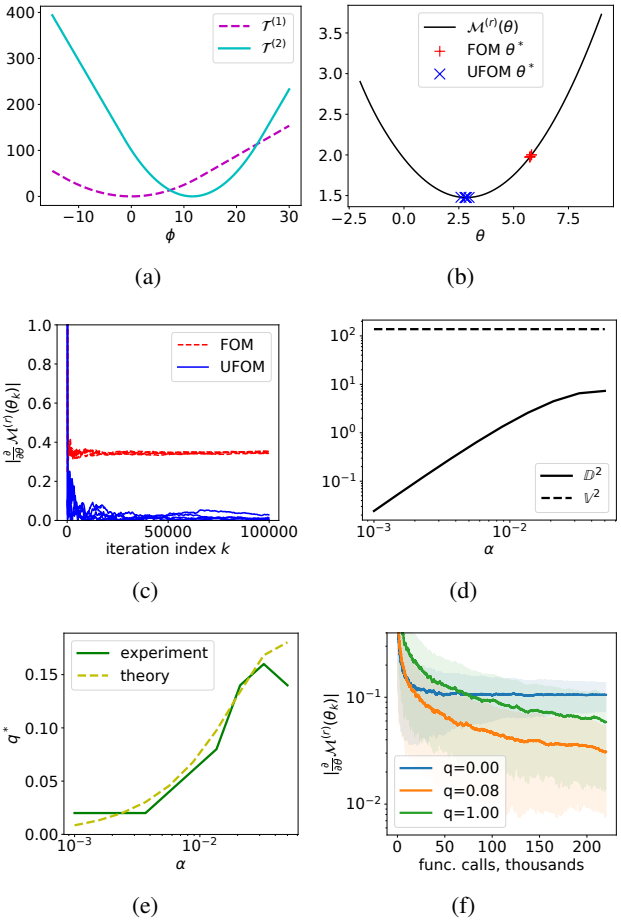

Figure 1: **(a)** Two tasks are sampled with equal probability from $p(\mathcal{T})$. The goal of task i's inner GD is to optimize convex piecewise-polynomial $f_i(\phi)$, $\phi \in \mathbb{R}$. **(b)** Resulting ABLO objective $\mathcal{M}^{(r)}(\theta)$ where θ is a starting ϕ value for $(r = 10)$-step inner GD. Markers indicate results of outer SGD from a random starting parameter θ_0 using FOM and UFOM ($q = 0.1$). **(c)** Convergence of $|\frac{\partial}{\partial\theta}\mathcal{M}^{(r)}(\theta_k)|$, k is an outer-SGD iteration. UFOM is approaching zero gradient norm, which is not true for FOM. **(d)** We vary inner-GD learning rate α in a range of $[10^{-3}, 5 \cdot 10^{-2}]$ and output numerically approximated $\mathbb{V}^2$ and $\mathbb{D}^2$. **(e)** Using obtained $\mathbb{V}^2$ and $\mathbb{D}^2$ estimates, we compute theoretically optimal q^* as proposed in Section 4.4 ("theory" curve). Also, we find optimal q empirically by grid search ("experiment" curve). **(f)** Using the setup from Figures (d), (e), we fix $\alpha = 10^{-2}$ and plot optimization curves for $q = 0$ (FOM), $q = 1$ (exact gradient) and $q = q^* \approx 0.08$, where q^* is taken from the "theory" plot on Figure (e). We observe the fastest convergence when $q = q^*$.

$q_{min} = 0.05, \beta = 0.99$ performs reasonably well. Further, we empirically find that Adaptive UFOM works best when (22) is modified so that $\overline{\mathbb{D}_k^2} = 0.1 \cdot \mathbb{D}_{sm,k}^2 / (1 - \beta^{k_{upd}})$ (dividing initial $\overline{\mathbb{D}_k^2}$ by 10). We optimize all compared methods

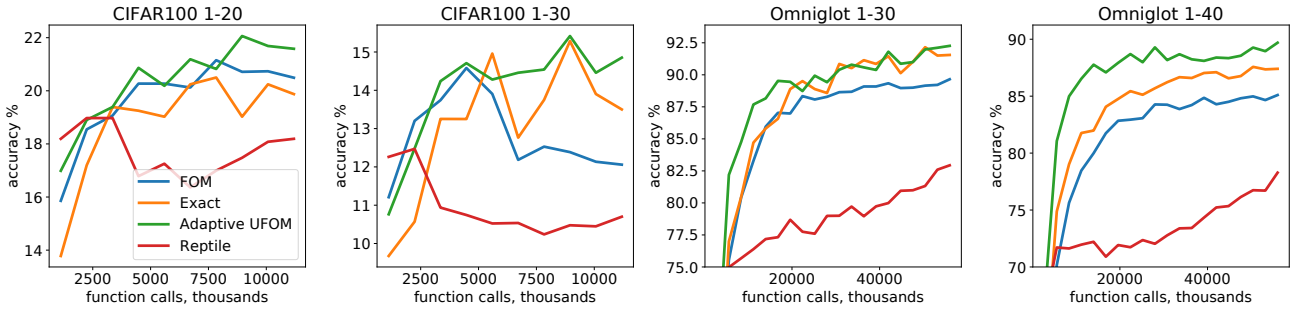

Figure 2: Test-set accuracy curves. K-m in the title corresponds to K-shot m-way setup.

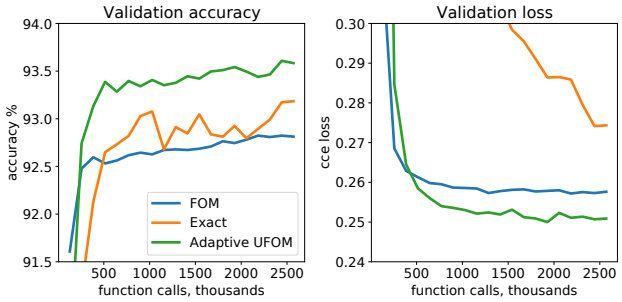

Figure 3: Data hypercleaning, training curves.

for the same amount of function calls – see more details, including the evolution of adaptive probabilities, in Appendix C. Table 1 and Figure 3 demonstate optimization results: as compared to exact gradient and FOM, Adaptive UFOM results in a faster optimization and better generalization. We keep the same value of q_{min}, β and modification of (22) in our subsequent few-shot learning experiments.

5.3 Few-shot Image Classification

Few-shot image classification (Finn et al., 2017) addresses adaptation to a new task when supplied with a small amount of training data. Each task $\mathcal{T}$ is then a pair $\mathcal{T} = (\mathcal{D}_{\mathcal{T}}^{tr}, \mathcal{D}_{\mathcal{T}}^{test})$, where

$$\mathcal{D}_{\mathcal{T}}^{tr} = ((X_i^{tr}, Y_i^{tr}))_{i=1}^{t}, \quad \mathcal{D}_{\mathcal{T}}^{test} = ((X_i^{test}, Y_i^{test}))_{i=1}^{t'},$$

where X are classified images, $Y \in \{1, \dots, m\}$ are labels, $\mathcal{D}_{\mathcal{T}}^{tr}$ is a training set of a small size t, and $\mathcal{D}_{\mathcal{T}}^{test}$ is a test set of size t'. Let $g(\phi, X) \in \mathbb{R}^m$, be a classifier with parameters ϕ and input X. Define $\mathcal{L}_{\text{CCE}}(\phi, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(X,Y)\in\mathcal{D}} l_{\text{CCE}}(g(\phi, X), Y)$. *Model-Agnostic Meta-Learning* (MAML) (Finn et al., 2017) states the problem of few-shot classification as ABLO (2-4) where $V(\theta, \mathcal{T}) \equiv \theta \in \mathbb{R}^p$, $\mathcal{L}^{in}(\theta, \phi, \mathcal{T}) \equiv \mathcal{L}_{\text{CCE}}(\phi, \mathcal{D}_{\mathcal{T}}^{tr})$ and $\mathcal{L}^{out}(\theta, \phi, \mathcal{T}) \equiv \mathcal{L}_{\text{CCE}}(\phi, \mathcal{D}_{\mathcal{T}}^{test})$. This way, inner GD corresponds to fitting $g(\phi, \cdot)$ to a training set $\mathcal{D}_{\mathcal{T}}^{tr}$ of a small size, while the outer SGD is searching for an initialization $\theta = \phi_0$ maximizing generalization on the unseen data $\mathcal{D}_{\mathcal{T}}^{test}$.

We evaluate Adaptive UFOM on CIFAR100 and Omniglot. Both datasets consist of many classes with a few images for each class. To sample from $p(\mathcal{T})$ in the K-shot m-way setting, m classes are chosen randomly and $K+1$ examples are drawn from each class: K examples for training and 1 for testing, i.e. $s = mK, t = m$. We reuse convolutional architectures for $g(\phi, X)$ from (Finn et al., 2017) and set inner-loop length to $r = 10$, as in (Nichol et al., 2018). We also compare with Reptile (Nichol et al., 2018) – another modification of MAML which does not store intermediate states in memory. We run all methods for the same amount of function calls (Figure 2 and Table 2). Adaptive UFOM shows the best performance in all setups. More details, including adaptive q plots, can be found in Appendix D.

Table 1: Data hypercleaning, accuracy (%) and CCE. First row shows accuracy on the uncorrupted part of the train set.

	TRAIN	TEST	TEST CCE
EXACT	94.07	90.82	0.4795
FOM	92.04	90.83	0.3187
ADAPTIVE UFOM	**95.27**	**92.24**	**0.2811**

Table 2: Test accuracy (%) in a range of $(K = 1)$-shot setups, with varying m-ways. Abbreviations: "C-100" = CIFAR100 and "OMNI." = Omniglot.

	m	EXACT	REPTILE	FOM	AD.UFOM
C-100	20	19.9	18.5	21.3	**22.1**
C-100	30	13.8	11.0	12.6	**14.7**
OMNI.	30	91.6	83.1	89.3	**92.0**
OMNI.	40	87.5	78.2	84.88	**89.1**

6 Conclusion

We proposed an unbiased first-order method (UFOM), providing a memory-efficient unbiased gradient estimator for approximate bi-level optimization (ABLO). We show that UFOM-based SGD converges to a stationary point of the ABLO problem, and derive an expression for the optimal form of UFOM, which results in faster convergence than the exact gradient and first-order methods. Finally, we propose a method for choosing the parameter of UFOM adaptively.

7 Acknowledgements

We thank John Bronskill for helpful feedback on an early version of the manuscript. We further thank anonymous reviewers for their valuable feedback.

Valerii Likhosherstov acknowledges support from the Cambridge Trust and DeepMind. Adrian Weller acknowledges support from a Turing AI Fellowship under grant EP/V025379/1, The Alan Turing Institute under EPSRC grant EP/N510129/1 and TU/B/000074, and the Leverhulme Trust via CFI.

References

Ablin, P., Peyré, G., and Moreau, T. Super-efficiency of automatic differentiation for functions defined as a minimum, 2020.

Balcan, M., Khodak, M., and Talwalkar, A. Provable guarantees for gradient-based meta-learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 424–433, 2019.

Beatson, A. and Adams, R. P. Efficient optimization of loops and limits with randomized telescoping sums. volume 97 of *Proceedings of Machine Learning Research*, pp. 534–543, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/beatson19a.html.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Chen, Z., Jiang, H., Dai, B., and Zhao, T. Learning to defense by learning to attack. *CoRR*, abs/1811.01213, 2018. URL http://arxiv.org/abs/1811.01213.

Christianson, B. Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, 12 (2):135–150, 04 1992. ISSN 0272-4979. doi: 10.1093/imanum/12.2.135. URL https://doi.org/10.1093/imanum/12.2.135.

Fallah, A., Mokhtari, A., and Ozdaglar, A. E. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. *CoRR*, abs/1908.10400, 2019. URL http://arxiv.org/abs/1908.10400.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Finn, C., Rajeswaran, A., Kakade, S. M., and Levine, S. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 1920–1930, 2019.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1165–1173, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/franceschi17a.html.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/franceschi18a.html.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in neural information processing systems*, pp. 3581–3590, 2017.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *CoRR*, abs/1607.05447, 2016. URL http://arxiv.org/abs/1607.05447.

Griewank, A. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1(1):35–54, 1992. doi: 10.1080/10556789208805505. URL https://doi.org/10.1080/10556789208805505.

Griewank, A. and Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2008. ISBN 9780898717761. URL https://books.google.co.uk/books?id=xoiiLaRxcbEC.

Gu, S., Levine, S., Sutskever, I., and Mnih, A. Muprop: Unbiased backpropagation for stochastic neural networks. In *ICLR (Poster)*, 2016. URL http://arxiv.org/abs/1511.05176.

Hascoet, L. and Araya-Polo, M. Enabling user-driven checkpointing strategies in reverse-mode automatic differentiation. *arXiv preprint cs/0606042*, 2006.

Hazan, E. Introduction to online convex optimization. *CoRR*, abs/1909.05207, 2019. URL http://arxiv.org/abs/1909.05207.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In Bengio, Y. and LeCun, Y. (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pp. 2575–2583, 2015.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-100 (canadian institute for advanced research). 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Lake, B., Salakhutdinov, R., Gross, J., and Tenenbaum, J. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. 2010. *URL http://yann. lecun. com/exdb/mnist*, 7:23, 2010.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. *arXiv preprint arXiv:1911.02590*, 2019.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pp. 2113–2122, 2015.

Maheswaranathan, N., Metz, L., Tucker, G., Choi, D., and Sohl-Dickstein, J. Guided evolutionary strategies: augmenting random search with surrogate gradients. volume 97 of *Proceedings of Machine Learning Research*, pp. 4264–4273, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/maheswaranathan19a.html.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL http://arxiv.org/abs/1803.02999.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 737–746. JMLR.org, 2016.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems 32*, pp. 113–124. 2019.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. *arXiv preprint arXiv:1810.10667*, 2018.

Song, X., Gao, W., Yang, Y., Choromanski, K., Pacchiano, A., and Tang, Y. ES-MAML: Simple hessian-free meta learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1exA2NtDB.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pp. 2627–2636, 2017.

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bylnx209YX.