# A Fully Single Loop Algorithm for Bilevel Optimization without Hessian Inverse

**Junyi Li,[1] Bin Gu,[2] Heng Huang [1]***

[1] Electrical and Computer Engineering, University of Pittsburgh, PA, USA [2] MBZUAI, United Arab Emirates
junyili.ai@gmail.com, jsgubin@gmail.com, heng.huang@pitt.edu

## Abstract

In this paper, we propose a new Hessian inverse free Fully Single Loop Algorithm (FSLA) for bilevel optimization problems. Classic algorithms for bilevel optimization admit a double loop structure which is computationally expensive. Recently, several single loop algorithms have been proposed with optimizing the inner and outer variable alternatively. However, these algorithms not yet achieve fully single loop. As they overlook the loop needed to evaluate the hyper-gradient for a given inner and outer state. In order to develop a fully single loop algorithm, we first study the structure of the hyper-gradient and identify a general approximation formulation of hyper-gradient computation that encompasses several previous common approaches, *e.g.* back-propagation through time, conjugate gradient, *etc.* Based on this formulation, we introduce a new state variable to maintain the historical hyper-gradient information. Combining our new formulation with the alternative update of the inner and outer variables, we propose an efficient fully single loop algorithm. We theoretically show that the error generated by the new state can be bounded and our algorithm converges with the rate of $O(\epsilon^{-2})$. Finally, we verify the efficacy our algorithm empirically through multiple bilevel optimization based machine learning tasks.

## 1 Introduction

In this paper, we study the bilevel optimization problem, which includes two levels of optimization: an outer problem and an inner problem. The outer problem depends on the solution of the inner problem. Many machine learning tasks can be formulated as a bilevel optimization problem, such as hyper-parameter optimization (Lorraine and Duvenaud 2018), meta learning (Franceschi et al. 2018), Stackelberg game model (Ghadimi and Wang 2018), equilibrium model (Grazzi et al. 2020), *etc.* However, Bilevel optimization is challenging to solve. The gradient-based algorithm (Ghadimi and Wang 2018) requires a double loop structure. For each inner loop, the inner problem is solved with the given outer state. In the outer loop, the hyper-gradient (the gradient *w.r.t* the outer variable) is evaluated based on the solution of the inner loop and the outer state is updated with a gradient-based optimizer (such as SGD). This simple double loop algorithm is guaranteed to converge under mild assumptions and works well for small scale problems. But when the inner problem is in large scale, the double loop algorithm is very slow and becomes impractical.

In fact, hyper-gradient evaluation is the major bottleneck of bilevel optimization. The solution of the inner problem is usually implicitly defined over the outer state, and naturally its gradient *w.r.t* the outer variable is also implicitly defined. To evaluate hyper-gradient, we need to approximate this implicit gradient via an iterative algorithm (the inner loop). Various algorithms have been proposed for hyper-gradient evaluation (Ferris and Mangasarian 1991; Ghadimi and Wang 2018; Grazzi et al. 2020; Lorraine and Duvenaud 2018; Liao et al. 2018). These methods were designed from various perspectives and based on different techniques, thus they look quite different on the first sight. However, we can use a general formulation to incorporate all these methods under one framework. Roughly, the hyper-gradient evaluation can be expressed as a finite sum of terms defined over a sequence of inner states and momentum coefficients, and these states and coefficients are chosen differently in distinct algorithms. Our general formulation provides a new perspective to help understand the properties of these classic methods. In particular, we derive a sufficient condition such that the general formulation converges to the exact hyper-gradient.

One benefit of our general formulation is to inspire new algorithms for bilevel optimization. Based on our general formulation, we propose a new fully single loop algorithm named as 'FSLA'. Compared to previous single loop algorithms (Guo et al. 2021; Chen, Sun, and Yin 2021; Ji, Yang, and Liang 2020; Khanduri et al. 2021; Huang and Huang 2021a), our FSLA does not require any inner loop. In literature (Ji, Yang, and Liang 2020; Khanduri et al. 2021), the existing methods either reuse the last hyper-iteration's inner solution as a warm start of the current iteration, or just alternatively update the inner and outer variables with carefully designed learning rate schedule (Hong et al. 2020). They also utilize the variance-reduction techniques to control the variance (Khanduri et al. 2021). However, these methods focus on solving the inner problem and pay little attention to the hyper-gradient evaluation process which also requires a loop. In our new algorithm, we introduce a new state $v_k$ to keep track of the historical hyper-gradient information.

*This work was supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, IIA 2040588.

During each iteration, we perform one step update. We study the bias caused by $v_k$ and theoretically show that the convergence of our new algorithm is with rate $O(\epsilon^{-2})$. The main contributions of this paper can be summarized as follows:

1. We propose a general formulation to unify different existing hyper-gradient approximation methods under the same framework, and identify the sufficient condition on which it converges to the exact hyper-gradient;
2. We propose a new fully single loop algorithm for bilevel optimization. The new algorithm avoids the need of time-consuming Hessian inverse by introducing a new state to track the historical hyper-gradient information;
3. We prove that our new algorithm has fast $\mathrm{O}(\epsilon^{-2})$ convergence rate for the nonconvex-strongly-convex case, and we validate effectiveness of our algorithm over different bilevel optimization based machine learning tasks.

**Organization:** The rest of this paper is organized as follows: in Section 2, we briefly review the recent development of Bilevel optimization; in Section 3, we introduce the general formulation of hyper-gradient approximation and the sufficient condition of convergence; in Section 4, we formally propose our new fully single loop algorithm, FSLA; in Section 5, we present the convergence result of our algorithm; in Section 6, we perform experiments to validate our proposed methods; in Section 7, we conclude and summarize the paper.
**Notations:** We use $\nabla_x$ to denote the full gradient *w.r.t.* the variable $x$, where the subscript is omitted if clear from the context, and $\partial_x$ denotes the partial derivative. Higher order derivatives follow similar rules. $||\cdot||$ represents $\ell_2$-norm for vectors and spectral norm for matrices. $[K]$ represents the sequence from 0 to $K$. $\Pi_{i=m}^{n} A_i = A_m \times \dots A_n$ if $m \leq n$, and $\Pi_{i=m}^{n} A_i = I$ if $m > n$.

## 2 Related Works

Bilevel optimization dates back to the 1960s when Willoughby (1979) proposed a regularization method, and then followed by many research works (Ferris and Mangasarian 1991; Solodov 2007; Yamada, Yukawa, and Yamagishi 2011; Sabach and Shtern 2017). In machine learning community, similar ideas in the name of implicit differentiation were also used in Hyper-parameter Optimization for a long time (Larsen et al. 1996; Chen and Hagan 1999; Bengio 2000; Do, Foo, and Ng 2007). However, implicit differentiation needs to compute an accurate inner problem solution in each update of the outer variable, which leads to high computational cost for large-scale problems. Thus, researchers turned to solve the inner problem with a fix number of steps, and computed the gradient *w.r.t* the outer variables with the 'back-propagation through time' technique (Domke 2012; Maclaurin, Duvenaud, and Adams 2015; Franceschi et al. 2017; Pedregosa 2016; Shaban et al. 2018). Domke (2012) considered the case when the inner optimizer is gradient-descent, heavy ball, and LBFGS method, and derived their reversing dynamics with the energy models as example; Maclaurin, Duvenaud, and Adams (2015) considered momentum-based SGD method, furthermore, Franceschi et al. (2017) discussed two modes, a forward mode and a backward mode, and compared the trade-off of these two modes in terms of memory-usage and computation efficiency; Pedregosa (2016) studied the influence of inner solution errors to the convergence of bilevel optimization; Shaban et al. (2018) proposed to truncate the back propagation path to save computation.

The back-propagation based methods work well in practice, but the number of inner steps usually relies on trial and error, furthermore, it is still expensive to perform multiple inner steps for modern machine learning models with hundreds of millions of parameters. Recently, it witnessed a surge of interest in using implicit differentiation to derive single loop algorithms. Ghadimi and Wang (2018) introduced BSA, an accelerated approximate implicit differentiation method with Neumann Series. Hong et al. (2020) proposed TTSA, a single loop algorithm with a two-timescale learning rate schedule. Ji, Yang, and Liang (2020) and Ji and Liang (2021) presented warm start strategy to reduce the number of inner steps needed at each iteration. Khanduri et al. (2021) designed SUSTAIN which applied variance reduction technique (Cutkosky and Orabona 2019; Huang, Li, and Huang 2021) over both the inner and the outer variable. Chen, Sun, and Yin (2021) proposed STABLE, a single loop algorithm accumulating the Hessian matrix, and achieved the same order of sample complexity as the single-level optimization problems. Yang, Ji, and Liang (2021) proposed two algorithms: MRBO and VRBO. The MRBO method uses double variance reduction trick and resembles SUSTAIN, while VRBO is based on SARAH/SPIDER. Huang and Huang (2021b) proposed BiO-BreD which is also based on the variance reduction technique with a better dependence over the condition number of inner problem. Meanwhile, there are also works utilizing other strategies like penalty methods (Mehra and Hamm 2019), and also other formulations like the case where the inner problem has non-unique minimizers (Li, Gu, and Huang 2020).

The bilevel optimization has been widely applied to various machine learning applications. Hyper-parameter optimization (Lorraine and Duvenaud 2018; Okuno, Takeda, and Kawana 2018; Franceschi et al. 2018) uses bilevel optimization extensively. Besides, the idea of bilevel optimization has also been applied to meta learning (Zintgraf et al. 2019; Song et al. 2019; Soh, Cho, and Cho 2020), neural architecture search (Liu, Simonyan, and Yang 2018; Wong et al. 2018; Xu et al. 2019), adversarial learning (Tian et al. 2020; Yin et al. 2020; Gao et al. 2020), deep reinforcement learning (Yang et al. 2018; Tschiatschek et al. 2019), *etc.* For a more thorough review of these applications, please refer to the Table 2 of the survey paper by Liu et al. (2021).

## 3 A General Formulation of Hyper-Gradient Approximation

In general, bilevel optimization has the following form:

$$\min_{\lambda \in \Lambda} \; f(\lambda) \coloneqq F(\lambda, \omega_\lambda) \quad \textit{s.t. } \omega_\lambda = \arg\min_{\omega} G(\lambda, \omega) \tag{1}$$

where $F$, $G$ denote the outer and inner problems, $\lambda$, $\omega$ denote the outer and inner variables. Under mild assumptions, the hyper-gradient $\nabla_\lambda f$ can be expressed in Proposition 1:

**Proposition 1.** *If for any $\lambda \in \Lambda$, $\omega_\lambda$ is unique, and $\partial_{\omega^2} G(\lambda, \omega_\lambda)$ is invertible, we have:*

$$\nabla_\lambda f = \partial_\lambda F(\lambda, \omega_\lambda) + \nabla_\lambda \omega_\lambda \times \partial_\omega F(\lambda, \omega_\lambda) \tag{2}$$

*and* $\nabla_\lambda \omega_\lambda = -\partial_{\omega\lambda} G(\lambda, \omega_\lambda) \partial_{\omega^2} G(\lambda, \omega_\lambda)^{-1}$.

The proof of this proposition is a direct application of the implicit function theorem (we include it in Appendix B.2 for completion). Eq. (2) is hard to evaluate due to the involved matrix inversion, and we instead evaluate it approximately. Various approximate methods are proposed in the literature. The most well-known ones are: Back Propagation through Time (BP), Neumann series (NS) and conjugate gradient descent (CG). They look quite different on the first sight: BP is derived based on the chain rule, NS and CG are based on different ways of approximating $\partial_{\omega^2} G(\lambda, \omega_\lambda)^{-1}$. However, they share a common structure, as stated in the following lemma. We use $\nabla_\lambda f$ to represent $\nabla_\lambda f(\lambda)$ when the outer state $\lambda$ is clear from the context. In the remainder of this section: we use $[K]$ to denote the sequence, $k$ refers to $k_{th}$ element of $[K]$, while $s$ refers to $s_{th}$ element of sub-sequence $[k]$.

**Lemma 2.** *Given a positive integer $K$, a sequence of inner variable states $\{\omega_k\}$, vectors $\{p_k\}$ and a sequence of coefficients $\{\beta_k\}$ for $k \in [K]$. A general form of approximate hyper-gradient evaluated at state $\lambda$ is:*

$$\nabla_\lambda f_K = \partial_\lambda F(\lambda, \omega_K) - \sum_{k=0}^{K-1} \beta_k s_k$$

*where $s_k$ has two modes:*

$$s_k = \begin{cases} \partial_{\omega\lambda} G(\lambda, \omega_k) \prod_{s=k+1}^{K-1} (I - \beta_s \partial_{\omega^2} G(\lambda, \omega_s)) p_K \\ \partial_{\omega\lambda} G(\lambda, \omega_K) \prod_{s=k+1}^{K-1} (I - \beta_s \partial_{\omega^2} G(\lambda, \omega_s)) p_k \end{cases}$$

*We call these two modes as backward and forward respectively (in terms of $p_k$). More specifically, we have:*

1. *BP is in backward mode with $\omega_k = \hat{\omega}_k$, $p_k = \partial_\omega F(\lambda, \hat{\omega}_k)$ and $\beta_k = \eta_k$ for $k \in [K]$. $\{\hat{\omega}_k\}$ are an inner variable sequence generated by the gradient descent algorithm and $\{\eta_k\}$ is the corresponding learning rate sequence;*
2. *NS is in backward mode with $\omega_k = \hat{\omega}$, $p_k = \partial_\omega F(\lambda, \hat{\omega})$ and $\beta_k = \beta$ for $k \in [K]$. $\hat{\omega}$ is an inner variable state and $\beta$ is some constant;*
3. *CG is in the forward mode with $\omega_k = \hat{\omega}$ for $k \in [K]$. $\hat{\omega}$ is an inner variable state, $\{p_k\}$ and $\{\beta_k\}$ is chosen adaptively by the conjugate steps.*

*Proof.* In the proof, we justify that the three cases mentioned above indeed satisfy the proposed general hyper-gradient computation formulation.

*Case (1)*: Without loss of generality, assume we run a gradient descent optimizer $K$ steps over the inner problem in the BP method. In other words, it solves:

$$\min_{\lambda \in \Lambda} \; f_K(\lambda) \coloneqq F(\lambda, \hat{\omega}_K)$$
$$s.t.\ \hat{\omega}_k = \hat{\omega}_{k-1} - \eta_k \nabla G(\lambda, \hat{\omega}_{k-1}), k \in [K]$$

where $\eta_k$ is the learning rate. By the chain rule, we get the hyper-gradient $\nabla_\lambda f_K$ of this problem:

$$\nabla_\lambda f_K = \partial_\lambda F(\lambda, \hat{\omega}_K) - \sum_{k=0}^{K-1} \Bigg( \eta_k \partial_{\omega\lambda} G(\lambda, \hat{\omega}_k) \times \prod_{s=k+1}^{K-1} (I - \eta_k \partial_{\omega^2} G(\lambda, \hat{\omega}_s)) \Bigg) \partial_\omega F(\lambda, \hat{\omega}_K)$$

It is easy to verify the claim in the lemma;

*Case (2)*: The NS method is based on the hyper-gradient expression in Proposition 1, but it assumes access of an approximate solution $\hat{\omega}$ instead of the optimum $\omega_\lambda$ and then approximates $\partial_{\omega^2} G(\lambda, \hat{\omega})^{-1}$ with the first $K$ terms of the Neumann series. More precisely:

$$\partial_{\omega^2} G(\lambda, \hat{\omega})^{-1} \approx \beta \sum_{k=0}^{K-1} \Big( I - \beta \partial_{\omega^2} G(\lambda, \hat{\omega}) \Big)^k \qquad (3)$$

where $\beta$ is a small constant. Then we replace $\omega_\lambda$ with $\hat{\omega}$ in Eq. (2) and combine with Eq. (3). We have:

$$\nabla_\lambda f_K = \partial_\lambda F(\lambda, \hat{\omega}) - \sum_{k=0}^{K-1} \Bigg( \beta \partial_{\omega\lambda} G(\lambda, \hat{\omega}) \times \Big( I - \beta \partial_{\omega^2} G(\lambda, \hat{\omega}) \Big)^k \Bigg) \partial_\omega F(\lambda, \hat{\omega})$$

Substitute $k$ with $K - k - 1$, it is straightforward to verify the claim in the lemma;

*Case (3)*: The CG method uses the fact that $x = \partial_{\omega^2} G(\lambda, \hat{\omega})^{-1} \partial_\omega F(\lambda, \hat{\omega})$ is the minimizer of the quadratic optimization problem: $\arg\min_v \frac{1}{2} x^T A x - x^T b$, where $A = \partial_{\omega^2} G(\lambda, \hat{\omega})$ and $b = \partial_\omega F(\lambda, \hat{\omega})$. Solving this quadratic problem with the conjugate gradient descent, we have the following update rule:

$$x_{k+1} = x_k - \alpha_k(p_k) = (I - \alpha_k A) x_k + \alpha_k (b + \gamma_k p_{k-1})$$

Then second equality follows the update rule of linear CG algorithm. $\alpha_k, \gamma_k$ are the learning rate and $p_k$ is the conjugate directions. Please refer to section 5 by Nocedal and Wright for more details. Then $x_K$ takes the following form:

$$x_K = \sum_{k=0}^{K-1} \alpha_k \prod_{s=k+1}^{K-1} (I - \alpha_s A)(b + \gamma_k p_{k-1}) \qquad (4)$$

Substitute the values of $A$ and $b$ into Eq. (4) and then combine it with Eq. (2), where we approximate $\partial_{\omega\lambda} G(\lambda, \omega_\lambda)$ with $\partial_{\omega\lambda} G(\lambda, \hat{\omega})$. It is straightforward to verify the claim in the lemma. This completes the proof. □

The general formulation in the lemma provides a unified view of the BP, NS and CG methods. Firstly, we can verify that using the NS method is equivalent to solving the quadratic problem (defined in case (3)) with (constant learning rate) the gradient descent. Since the CG method usually performs better than gradient descent in solving linear systems, we expect the CG method requires smaller $K$ than the NS method to reach a given estimation error. Then we compare NS with BP, their difference lies in the sequence $\{\omega_k\}$ and $\{\beta_k\}$: NS uses a single state $\hat{\omega}(\beta)$ , while BP uses a sequence of states $\{\omega_k\}(\{\beta_k\})$.

It would be interesting to identify sufficient conditions for $\{\beta_k\}$, $\{\omega_k\}$ and $\{p_k\}$ such that the general formulation converges to the exact hyper-gradient $\nabla_\lambda f$. In fact, we have the following lemma:

**Lemma 3.** *Suppose we denote* $m_k = \prod_{s=0}^{k}(1-\mu_G\beta_s)$, $e_{\omega,k} = ||\omega_k - \omega_\lambda||$, $e_{p,k} = ||p_k - \partial_\omega F(\lambda,\omega_\lambda)||$ *for* $k \in [K]$. *Then if* $\lim_{K\to\infty} m_K = 0$, $\lim_{k\to\infty} e_{\omega,K} = 0$, $\lim_{K\to\infty} m_K \sum_{k=0}^{K-1} \beta_k e_{\omega,k}/m_k$ *is finite, in addition, for the backward mode:* $\lim_{K\to\infty} e_{p,K} = 0$*; for the forward mode:* $\lim_{K\to\infty} m_K \sum_{k=0}^{K-1} \beta_k e_{p,k}/m_k$ *is finite. Then we have* $\nabla_\lambda f_K \to \nabla_\lambda f$*, when* $K \to \infty$*, where* $\nabla_\lambda f_K$ *is defined in Lemma 2.*

We defer the proof of Lemma 3 in in Appendix C, we show some basic ideas here. Firstly, it is straightforward to show $\partial_\lambda F(\lambda,\omega_K) \to \partial_\lambda F(\lambda,\omega_\lambda)$ by using the smoothness assumption and the condition $\omega_K \to \omega_\lambda$. For the term $\sum_{k=0}^{K-1}\beta_k s_k$, we take the backward mode as an example (the forward mode follows similar idea). We denote $A_K$ and $A^*$ as:

$$A_K = \sum_{k=0}^{K-1} \beta_k \partial_{\omega\lambda} G(\lambda,\omega_k) \prod_{s=k+1}^{K-1} (I - \beta_s \partial_{\omega^2} G(\lambda,\omega_s)) \tag{5}$$

and $A^* = \partial_{\omega\lambda}G(\lambda,\omega_\lambda)\partial_{\omega^2}G(\lambda,\omega_\lambda)^{-1}$. To show $A_K \to A^*$, we use the recursive relation of $A_k$ and $A^*$:

$$\begin{aligned} A_{k+1} =& A_k\left(I - \beta_k\partial_{\omega^2}G(\lambda,\omega_k)\right) + \beta_k\partial_{\omega\lambda}(\lambda,\omega_k) \\ =& (1-\beta_k)A_k + \beta_k(A_k(I - \partial_{\omega^2}G(\lambda,\omega_k)) \\ &+ \partial_{\omega\lambda}G(\lambda,\omega_k)) \end{aligned} \tag{6}$$

and $A^* = A^*(I - \partial_{\omega^2}G(\lambda,\omega_\lambda)) + \partial_{\omega\lambda}G(\lambda,\omega_\lambda)$. Based on the recursive relation above and some linear algebra derivation, we get:

$$||A_{k+1} - A^*|| \leq (1-\mu_G\beta_k)||A_k - A^*|| + C_1\beta_k e_{\omega,k} \tag{7}$$

It is straightforward to verify that the conditions in the lemma guarantee the convergence of Eq. (7). As shown in Eq. (7), the momentum coefficient $\beta_k$ represents the trade-off between the progress and the induced error in one iteration: Larger $\beta_k$ leads to better contraction factor $\epsilon$, but also bigger bias term $e_{\omega,k}$. In fact, we can get meaningful convergence rate of $\nabla_\lambda f_K$ for several special choices of sequences. As shown in Corollary 3.1 and Corollary 3.2:

**Corollary 3.1.** *Given a positive integer* $K$*, inner state* $\hat{\omega}_K$ *and constant* $\beta$*. Then we set* $\omega_k = \hat{\omega}_K$, $p_k = \partial_\omega F(\lambda,\hat{\omega}_K)$ *and* $\beta_k = \beta$ *for* $k \in [K]$*. If* $\exists\ \epsilon \in (0,1)$*, such that* $\beta_k \in (\epsilon/\mu_G, 1/\mu_G)$*, then we have:*

$$||\nabla_\lambda f_K - \nabla_\lambda f|| = O(e_{\omega,K})$$

**Corollary 3.2.** *Given a positive integer* $K$*, we have sequence* $\{\hat{\omega}_k\}$ *and* $\{\hat{\beta}_k\}$ *for* $k \in [K]$*. We pick* $\omega_k = \hat{\omega}_k$, $p_k = \partial_\omega F(\lambda,\hat{\omega}_K)$ *and* $\beta_k = \hat{\beta}_k$ *for* $k \in [K]$*. Suppose we have* $\beta_k = O(k^{-1})$ *and* $e_{\omega,k} = O(k^{-0.5})$*, then:*

$$||\nabla_\lambda f_K - \nabla_\lambda f|| = O(K^{-0.5})$$

In fact, the sequences chosen in Corollary 3.1 correspond to that in the NS method, and we show that if we choose $\beta$ properly, the hyper-gradient estimation converges in the rate of $e_{\omega,K}$. The BP method corresponds to Corollary 3.2. Suppose we optimize the inner problem with stochastic gradient descent and learning rate $\beta_k = O(1/k)$, we get $e_{\omega,k} = O(1/\sqrt{k})$. This satisfies the condition in the corollary. To the best of our knowledge, this is first complexity result of the stochastic case. Grazzi et al. considered the linear convergence case for BP method. We introduce some more examples of the application of Lemma 3 in the Appendix C.

## 4 New Fully Single Loop Algorithm (FSLA)

In this section, we introduce a new single loop algorithm for bilevel optimization. In section 3, we identify a general formulation of hyper-gradient approximation in Lemma 2: $\nabla_\lambda f_K = \partial_\lambda F(\lambda,\omega_K) - \sum_{k=0}^{K-1}\beta_k s_k$, where $s_k$ has two modes: forward mode and backward mode. For both modes, they can be expressed by a recursive equation. In the backward mode, we write a recursive relation in terms of $A_k$ as defined in Eq. (5) and Eq. (6). Similarly, we define $v_K = \sum_{k=0}^{K-1}\beta_k\prod_{s=k+1}^{K-1}\left(I - \beta_s\partial_{\omega^2}G(\lambda,\omega_s)\right)p_k$ in the forward mode, and derive a recursive relation as:

$$v_{k+1} = \left(I - \beta_k\partial_{\omega^2}G(\lambda,\omega_k)\right)v_k + \beta_k p_k \tag{8}$$

Based on this observation, we can propose a new single loop bilevel optimization algorithm without Hessian Inverse. In previous literature, researchers maintain a inner state $\omega_k$ to avoid the inner loop solving $\omega_\lambda$ for each new outer state $\lambda$. On top of this, we maintain a new state $v_k$ and evaluate the hyper-gradient as follows:

$$\begin{aligned} v_k &= \beta_k\partial_\omega F(\lambda,\omega_k) + (I - \beta_k\partial_{\omega^2}G(\lambda,\omega_k))v_{k-1} \\ \nabla_\lambda f_k &= \partial_\lambda F(\lambda,\omega_k) - \partial_{\omega\lambda}G(\lambda,\omega_k)v_k \end{aligned} \tag{9}$$

Note we set $p_k = \partial_\omega F(\lambda,\omega_k)$. Then we alternatively update $\omega_k$, $v_k$ and $\lambda_k$, and achieve a fully single loop algorithm without Hessian-Inverse. Note that it is also possible to keep track of $A_k$, but then we need to store a matrix, which cost more storage. More formally, we get the following new alternative update rule:

$$\begin{aligned} \lambda_k &= \lambda_{k-1} - \alpha_{k-1}\nabla_\lambda f_{k-1} \\ \omega_k &= \omega_{k-1} - \tau_k\partial_\omega G(\lambda_k,\omega_{k-1}) \\ v_k &= \beta_k\partial_\omega F(\lambda_k,\omega_k) + (I - \beta_k\partial_{\omega^2}G(\lambda_k,\omega_k))v_{k-1} \\ \nabla_\lambda f_k &= \partial_\lambda F(\lambda_k,\omega_k) - \partial_{\omega\lambda}G(\lambda_k,\omega_k)v_k \end{aligned} \tag{10}$$

where $\tau_k$ and $\alpha_k$ are learning rates for inner and outer updates. As a comparison, existing single loop algorithms recompute $\nabla_\lambda f$ from scratch for each new $\lambda$, while our algorithm performs one step update over $v_k$. What's more, we can express $\nabla_\lambda f_k$ in Eq. (10) as follows:

$$\begin{aligned} \nabla_\lambda f_K(\lambda_K) =& \partial_\lambda F(\lambda_K,\omega_K) - \sum_{k=0}^{K-1}\beta_k\partial_{\omega\lambda}G(\lambda_K,\omega_K)\times \\ & \prod_{s=k+1}^{K-1}(I - \beta_s\partial_{\omega^2}G(\lambda_s,\omega_s))\partial_\omega F(\lambda_k,\omega_k) \end{aligned}$$

This almost fits the forward mode of the general formulation in Lemma 2 except that the outer state is a sequence $\{\lambda_k\}$. In

Algorithm 1: Fully Single Loop Bilevel Optimization Algorithm (FSLA)

1: **Input:** Initial state $\lambda_0 \in \Lambda$, $\omega_0 \in R^n$; the number of hyper-iterations $K$; constants $c_\tau$, $c_\beta$, $c_\eta$, $\delta$
2: **for** $k \leftarrow 0$ **to** $K-1$ **do**
3: $\alpha_k \leftarrow \delta/\sqrt{k}$, $\lambda_{k+1} \leftarrow \lambda_k - \alpha_k d_k$
4: $\tau_{k+1} \leftarrow c_\tau \alpha_k$, $\beta_{k+1} \leftarrow c_\beta \alpha_k$ and $\eta_{k+1} \leftarrow c_\eta \alpha_k$
5: Sample $\xi_{k+1}(\xi_{k+1,1} - \xi_{k+1,5})$
6: $\omega_{k+1} \leftarrow \omega_k - \tau_{k+1}\partial_\omega G(\lambda_{k+1}, \omega_k; \xi_{k+1,1})$
7: $v_{k+1} \leftarrow \beta_{k+1}\partial_\omega F(\lambda_{k+1}, \omega_k; \xi_{k+1,2}) + (I - \beta_{k+1}\partial_{\omega^2} G(\lambda_{k+1}, \omega_k; \xi_{k+1,3}))v_k$
8: $\nabla f_{k+1}(\xi_{k+1}) \leftarrow \partial_\lambda F(\lambda_{k+1}, \omega_{k+1}; \xi_{k+1,4}) - \partial_{\omega\lambda} G(\lambda_{k+1}, \omega_{k+1}; \xi_{k+1,5})v_{k+1}$
9: $d_{k+1} \leftarrow \nabla f_{k+1}(\xi_{k+1}) + (1 - \eta_{k+1})(d_k - \nabla f_k(\xi_{k+1}))$;
10: **end for**

Lemma 3, we show that $\omega_K \to \omega_\lambda$ is a sufficient condition of $\nabla_\lambda f_K(\lambda) \to \nabla_\lambda f(\lambda)$. It is reasonable to guess that if $\lambda_K \to \lambda$ and $\omega_K \to \omega_\lambda$, the convergence is also guaranteed. But the analysis is more challenging than Lemma 3 as the three terms $\lambda_k$, $\omega_k$ and $\nabla_\lambda f_k$ entangle with each other. However, we will show in the next section that $\lambda_K \to \lambda^*$ when $K \to \infty$, where $\lambda^*$ is the optimal point of the outer function $f(\lambda)$.

Finally in Algorithm 1, we provide the pseudo code of our fully single loop algorithm. Compared to Eq. (10), we assume access of the stochastic estimate of related values. What's more, we also maintain a momentum of the hyper-gradient $d_k$ with variance reduction correction in Line 10. This term is used to control the stochastic noise. The momentum-based variance reduction technique is recently widely used in the single level stochastic optimization, such as STORM (Cutkosky and Orabona 2019).

## 5 Theoretical Analysis

In this section, we prove the convergence of our proposed single loop bilevel optimization algorithm. We consider the non-convex-strongly-convex case. We first briefly state some assumptions needed in our theoretical analysis for the convenience of discussion. A formal description of the assumptions is in the Appendix A.

**Assumption A.** (Outer Function) Function $F$ is possibly non-convex, Lipschitz continuous with constant $L_{F,\lambda}$ (*w.r.t* $\lambda$) and $L_{F,\omega}$ (*w.r.t* $\omega$), and has bounded gradient with constant $C_F$;

**Assumption B.** (Inner Function) Function G is continuously twice differentiable, $\mu_G$-strongly convex w.r.t $\omega$ for any given $\lambda$, Lipschitz continuous with constant $L_{G,\omega}$ (*w.r.t* $\omega$). For higher-order derivatives, we have:

a) $\|\nabla^2_{\omega\lambda} G(\lambda, \omega)\| \le C_{G,\omega\lambda}$ for some constant $C_{G,\omega\lambda}$
b) $\nabla^2_{\omega\lambda} G(\lambda, \omega)$ and $\nabla^2_{\omega} G(\lambda, \omega)$ are Lipschitz continuous with constant $L_{G,\omega\lambda}$ and $L_{G,\omega\omega}$ respectively

**Assumption C.** (Bounded Variance) We have an unbiased stochastic oracle with bounded variance for estimating the related properties (gradient and Hessian), *e.g.* $E[\partial_\lambda F(\lambda, \omega; \xi)] = \partial_\lambda F(\lambda, \omega)$ and $var(\partial_\lambda F(\lambda, \omega; \xi)) \le \sigma^2$

We first bound the one iteration progress in the following lemma, which follows the usage of smoothness assumption.

**Lemma 4.** *Under Assumptions A, B, C, we have:*

$$\begin{aligned}
&E[f(\lambda_{k+1})] \\
\le& f(\lambda_k) - \frac{\alpha_k}{2}||\nabla f(\lambda_k)||^2 + \alpha_k \Gamma_2^2 E[||\omega_k - \omega_{\lambda_k}||^2] \\
&+ 4\alpha_k \Gamma_1^2 E[||v_k - v_{\lambda_k}||^2] + \alpha_k E[||\nabla f_k - d_k||^2] \\
&- \frac{\alpha_k}{2}(1 - \alpha_k L_f) E[||d_k||^2]
\end{aligned}$$

*where* $v_\lambda = (I - \partial_\omega \Phi(\lambda, \omega_\lambda))^{-1}\partial_\omega F(\lambda, \omega_\lambda)$ *and* $\Gamma_1^2 = C^2_{G,\omega\lambda}$, $\Gamma_2^2 = 2L^2_{F,\lambda} + 4C^2_{F,\omega}L^2_{G,\omega\lambda}/\mu_G^2$ *are constants.*

The proof is in Appendix D.2. Lemma 4 shows that there are three kinds of errors at each iteration: estimation error of $\omega_{\lambda_k}$ ($||\omega_{k+1} - \omega_{\lambda_k}||^2$), error of $v_{\lambda_k}$ ($||v_{k+1} - v_{\lambda_k}||^2$) and error of momentum $d_k$ ($||\nabla f_k - d_k||^2$). We denote them as $A_k$, $B_k$ and $C_k$ in the remainder of this section. The three errors entangle with each other as shown by Line 7-10 of Algorithm 1, *e.g.* the estimation error to $\omega_{\lambda_{k-1}}$ will contribute to the error of momentum $d_k$ as shown in Line 8. In fact, we can bound them with the following inequality (use $A_k$ as an example):

$$A_k \le \beta A_{k-1} + C_1 B_{k-1} + C_2 C_{k-1} + C_3$$

with $\beta < 1$ and $C_1$, $C_2$, $C_3$ are terms not relevant to $A_k$, $B_k$ and $C_k$. Please check the Appendix D.3 for more details. Specially, the momentum term $C_k$ reduces the variance similarly to that in the single level variance-reduction optimizers, by which we mean:

$$\begin{aligned}
&E[||d_k - \nabla f_k||^2] \\
\le&(1 - \eta_k)^2 E[||d_{k-1} - \nabla f_{k-1}||^2] + 2\eta_k^2 \sigma^2 \\
&+ 2(1 - \eta_k)^2 \underbrace{E[||\nabla f_k(\xi_k) - \nabla f_{k-1}(\xi_k)||^2]}_{\Pi}
\end{aligned}$$

The part of noise proportional to $(1 - \eta_k^2)\sigma^2$ is absorbed in the term $\Pi$ due to the correction made by $\nabla f_{k-1}(\xi_k)$ in the $d_k$ update rule. However, the key difference is that $\Pi$ not only relies on $E[||d_{k-1}||^2]$ but also $A_{k-1}$ and $B_{k-1}$.

Finally, to show the the convergence of Algorithm 1, we denote the following potential function:

$$\Phi_k = f(\lambda_k) + D_1 A_k + D_2 B_k + D_3 C_k$$

where $D_1$, $D_2$ and $D_3$ are some constants. Combine Lemma 4 with the inequalities for $A_k$, $B_k$ and $C_k$, and we have: $\Phi_{k+1} - \Phi_k \le -\alpha_k/2||\nabla f(\lambda_k)||^2 + C\alpha_k^2\sigma^2$ where $C$ is some constant. With the above inequality, we can get a convergence rate of $O(1/\sqrt{K})$ ($O(1/\epsilon^2)$) by choosing the learning rate with $O(1/\sqrt{k})$. More formally, we have:

**Theorem 5.** *With Assumption A, B, C hold, and take* $\beta_k = c_\beta \alpha_k$, $\tau_k = c_\tau \alpha_k$, $\eta_k = c_\eta \alpha_k$, *and* $\alpha_k = \frac{\delta}{\sqrt{k}}$. *We have:*

$$\frac{1}{K}\sum_{k=0}^{K-1} ||\nabla f(\lambda_k)||^2 \le \frac{2\Phi_0}{\delta\sqrt{K}} + \frac{2\delta\bar{C}\sigma^2}{\sqrt{K}}$$

*where* $\bar{C}$, $\delta$, $c_\beta$, $c_\tau$ *and* $c_\eta$ *are some constants.*

The proof of the theorem is included in Appendix D.4.

## 6 Experiments

In this section, we perform experiments to empirically verify the effectiveness of our algorithm. We first perform experiments over a quadratic objective with synthetic dataset to validate Lemma 3. Then we perform a common benchmark task in bilevel optimization: data hyper-cleaning. The experiments are run over a machine with Intel Xeon E5-2683 CPU and 4 Nvidia Tesla P40 GPUs.

### 6.1 Synthetic Dataset: Quadratic Objective

In this experiment, we verify Lemma 3 over some synthetic data. We consider the bilevel optimization problem where both outer and inner problems are quadratic. To make it simpler, the outer problem does not depend on the outer variable directly. More precisely, we study:

$$\min_{\lambda\in\Lambda} f(\lambda) := ||A_o\omega_\lambda - b_o||^2$$
$$s.t.\ \omega_\lambda = \arg\min_{\omega} ||A_{i,\lambda}\lambda + A_{i,\omega}\omega - b_i||^2$$

For this bilevel problem, we can solve the exact minimizer of the inner problem and then evaluate the exact hyper-gradient based on the Proposition 1. This makes it easier to compute and compare the approximation error of different methods. In experiments, we pick problem dimension 5 and randomly sample 10000 data points. We construct the dataset as follows: first randomly sample $A_o, A_{i,\lambda}, A_{i,\omega} \in \mathbb{R}^{10^4\times 5}$ and $\lambda, \omega, \omega_\lambda \in \mathbb{R}^5$ from the Uniform distribution, then we construct $b_o$ and $b_i$ by $A_o\omega_\lambda + \sigma_o$ and $A_{i,\lambda}\lambda + A_{i,\omega}\omega + \sigma_i$, where $\sigma_i$ and $\sigma_o$ are Gaussian noise with mean zero and variance 0.1. We use this simple task to validate our claim in Lemma 3. More precisely, we fix the outer state $\lambda$ and estimate the hyper-gradient with different sequence $\{\omega_k\}, \{\beta_k\}, \{p_k\}$ and then compare their estimation errors. The results are shown in Figure 1.

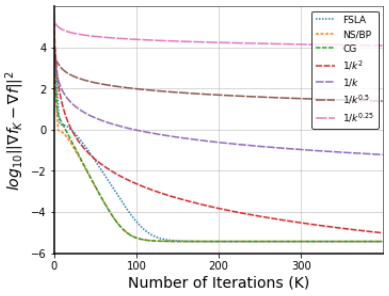


Figure 1: The estimation error of hyper-gradient $||\nabla f_K - \nabla f||^2$ for different sequences $\{\omega_k\}$.

We perform two sets of experiments. The first set includes our single loop method FSLA, and NS, BP and CG, which are three cases discussed in Lemma 2. For these methods, we generate the sequence $\{\omega_k\}$ through solving the inner problem with $K$ steps of gradient descent and we use learning rate $\beta_k$ we pick $2\times 10^{-5}$. Note since the second order derivatives of the quadratic objective are constant, BP and NS are the same. As shown by the figure, the hyper-gradient estimation errors of all the four methods converge. Our FSLA takes a bit more number of iterations to converge, however, our algorithm takes less running time. Our method requires $O(1)$ matrix-vector query for every given $K$, while the other methods requires $O(K)$ queries. In the next set of experiments, we compare with some synthetic $\{\omega_k\}$ sequences which have different convergence rate. More precisely, we use sequence $\{\omega + \tilde{\omega}/(k^\alpha)\}$, where $\tilde{\omega}$ is some random start point, we pick different alpha values (2, 1, 0.5, 0.25). We estimate $\nabla_\lambda f$ according to Corollary 3.1 (same as the NS), the learning rate is chosen as $2\times 10^{-5}$. Corollary 3.1 bounds the convergence rate of $\nabla_\lambda f_K$ with the rate of $\omega_k$, which is well verified through the results shown in Figure 1.

### 6.2 Hyper Data-cleaning

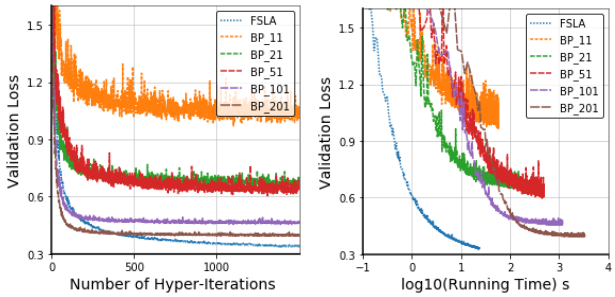


Figure 2: FSLA *vs* BP plot of Validation Loss w.r.t Number of hyper-iterations (Left) and $log10$(Running Time) (Right). The perturbation rate $\gamma$ is 0.8. The post-fix of legend represents the number of inner iterations $T$.

In this experiment, we demonstrate the efficiency of our FSLA, especially the effect of tracking hyper-gradient history with $v_k$. More precisely, we compare with three hyper-gradient evaluation methods: BP, NS and CG. For NS and CG methods, we consider both the double loop version and the single loop version. In the single loop version, we update the inner variable with the warm start strategy.

Data cleaning denotes the task of cleaning a noisy dataset. Suppose there is a noisy dataset $D_i$ with $N_i$ samples (the label of some samples are corrupted), the aim of the task is to identify those corrupted data samples. Hyper Data-cleaning approaches this problem by learning a weight per sample. More precisely, we solve the following bilevel optimization problem:

$$\min_{\lambda\in\Lambda} l(\omega_\lambda; D_o)\ s.t.\ \omega_\lambda = \arg\min_{\omega\in\mathbb{R}^d} \frac{1}{N_i}\sum_{j=1}^{N_i} \sigma(\lambda_j) l(\omega, D_{i,j})$$

In the inner problem, we minimize a weighted average loss $l$ over the training dataset $D_i$, with $\sigma(\lambda)$ the sample-wise weight ($\sigma(\cdot)$ is a normalization function and we use $Sigmoid$ in experiments), suppose the minimizer of the inner problem is $\omega_\lambda$. In in the outer problem, we evaluate $\omega_\lambda$ (a function of $\lambda$) over a validation dataset $D_o$. Then the bilevel problem will find $\lambda$ such that $\omega_\lambda$ is optimal as evaluated by the validation set.

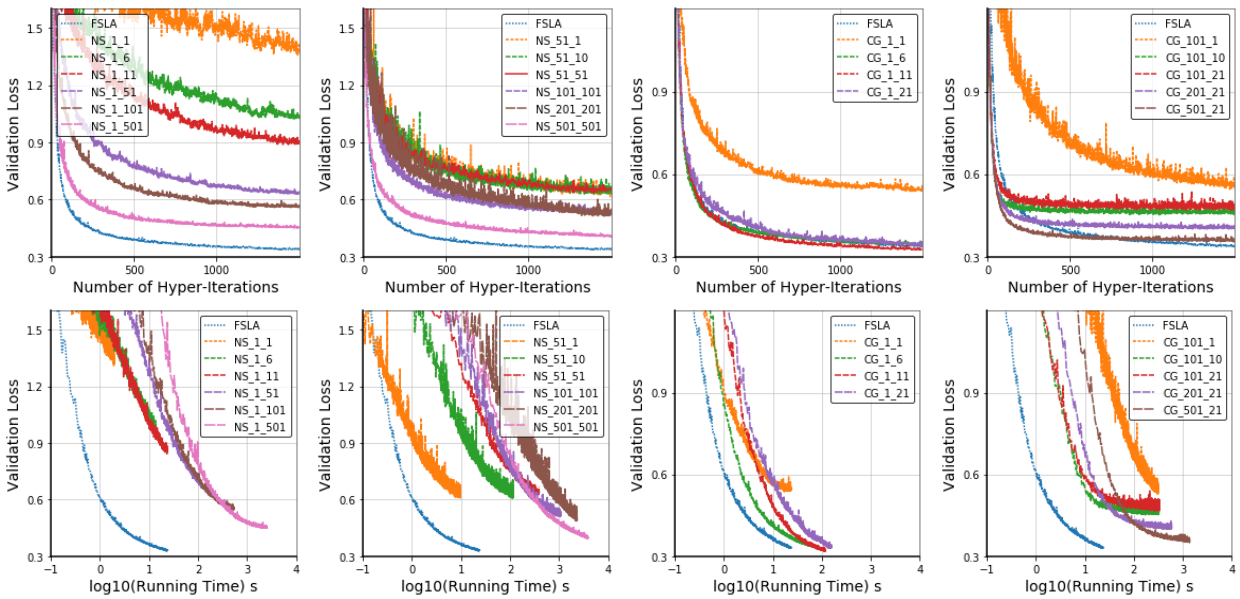


Figure 3: FSLA *vs* NS and CG plot of Validation Loss w.r.t Number of hyper-iterations (Top) and $log10$(Running Time) (Bottom). The perturbation rate $\gamma$ is 0.8. The post-fix of legend represents the number of inner iterations $T$ and approximate steps $K$.If inner gradient steps equal to 1, we use the warm start trick, otherwise not.

More specifically, we perform this task over several datasets: MNIST (LeCun, Cortes, and Burges 2010), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) and QMNIST (Yadav and Bottou 2019). We construct the datasets as follows: for the training set $D_i$, we choose 5000 images from the training set, and randomly perturb the label of $\gamma$ percentage of images. While for the validation set $D_o$, we randomly select another 5000 images but without any perturbation (all labels are correct). We adopt a 4-layer convolutional neural network in the training. The experimental results are shown in Figure 2 and Figure 3. For all the methods, we solve the inner problem with stochastic gradient descent, while for the outer optimizer, all the methods use the variance reduction for fair comparison. In experiments, we vary the number of inner gradient descent steps $T$ and the number of hyper-gradient approximation steps $K$. The legend in the figures has the form of *method*-$T$-$K$. For other hyper-parameters, we perform grid search for each method and choose the best one (hyper-parameters selection is in Appendix E).

As shown by the figures, our method converges much faster than the baseline methods. Compared with BP, FSLA surpasses the best BP variant $BP_{201}$ at the 500 iteration, as for the running time, FSLA runs much faster. For NS and CG, $T = 1$ in figures represents using the warm start, where the inner variable is updated from the state of last hyper-iteration. The warm start trick has some kind of acceleration effects. However, for NS, the running time is dominated by the evaluation of Neumann Series. For CG, it converges with around 10 steps, but the extra time is still considerable compared to FSLA. Even with similar computation cost, FSLA still outperforms CG. *CG_1_1* and FSLA both perform one step update per hyper-iteration, but CG converges much slower.

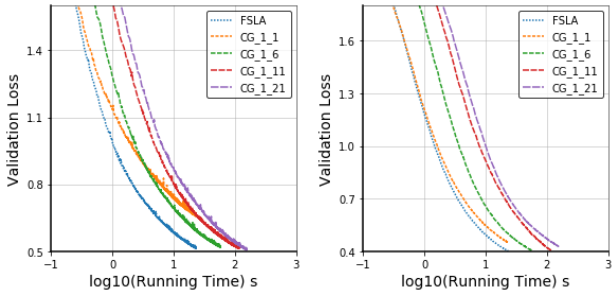


Figure 4: FSLA *vs* CG plot of Validation Loss w.r.t $log10$(Running Time). The Left plot shows Fashion-MNIST and the right plot shows the QMNIST. $\gamma$ is set 0.8.

This is due to failure of reusing the historical information. Finally, Figure 4 includes results for Fashion-MNIST and QMNIST, where we compare with the best baseline method CG. Our FSLA still outperforms it.

## 7 Conclusion

In this paper, we studied the bilevel optimization problem. More specifically, we first proposed a general formulation of hyper-gradient approximation. This formulation encompasses several important methods in the bilevel optimization. Then inspired by this, we introduced a new fully single loop algorithm, which performs alternative optimization of inner and outer variables. Our algorithm attains convergence rate $O(\epsilon^{-2})$. Moreover, the empirical results also verify the superior performance of our new algorithm.

## References


Bengio, Y. 2000. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8): 1889–1900.

Chen, D.; and Hagan, M. T. 1999. Optimal use of regularization and cross-validation in neural network modeling. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 2, 1275–1280. IEEE.

Chen, T.; Sun, Y.; and Yin, W. 2021. A single-timescale stochastic bilevel optimization method. *arXiv preprint arXiv:2102.04671*.

Cutkosky, A.; and Orabona, F. 2019. Momentum-based variance reduction in non-convex sgd. *arXiv preprint arXiv:1905.10018*.

Do, C. B.; Foo, C.-S.; and Ng, A. Y. 2007. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, volume 2007, 377–384. Citeseer.

Domke, J. 2012. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, 318–326. PMLR.

Ferris, M. C.; and Mangasarian, O. L. 1991. Finite perturbation of convex programs. *Applied Mathematics and Optimization*, 23(1): 263–273.

Franceschi, L.; Donini, M.; Frasconi, P.; and Pontil, M. 2017. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1165–1173. JMLR. org.

Franceschi, L.; Frasconi, P.; Salzo, S.; Grazzi, R.; and Pontil, M. 2018. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*.

Gao, C.; Chen, Y.; Liu, S.; Tan, Z.; and Yan, S. 2020. Adversarialnas: Adversarial neural architecture search for gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5680–5689.

Ghadimi, S.; and Wang, M. 2018. Approximation Methods for Bilevel Programming. *arXiv preprint arXiv:1802.02246*.

Grazzi, R.; Franceschi, L.; Pontil, M.; and Salzo, S. 2020. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, 3748–3758. PMLR.

Guo, Z.; Xu, Y.; Yin, W.; Jin, R.; and Yang, T. 2021. On Stochastic Moving-Average Estimators for Non-Convex Optimization. *arXiv preprint arXiv:2104.14840*.

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

Huang, F.; and Huang, H. 2021a. BiAdam: Fast Adaptive Bilevel Optimization Methods. *arXiv preprint arXiv:2106.11396*.

Huang, F.; and Huang, H. 2021b. Enhanced Bilevel Optimization via Bregman Distance. *arXiv preprint arXiv:2107.12301*.

Huang, F.; Li, J.; and Huang, H. 2021. SUPER-ADAM: Faster and Universal Framework of Adaptive Gradients. *arXiv preprint arXiv:2106.08208*.

Ji, K.; and Liang, Y. 2021. Lower Bounds and Accelerated Algorithms for Bilevel Optimization. *arXiv preprint arXiv:2102.03926*.

Ji, K.; Yang, J.; and Liang, Y. 2020. Provably Faster Algorithms for Bilevel Optimization and Applications to Meta-Learning. *arXiv preprint arXiv:2010.07962*.

Khanduri, P.; Zeng, S.; Hong, M.; Wai, H.-T.; Wang, Z.; and Yang, Z. 2021. A Near-Optimal Algorithm for Stochastic Bilevel Optimization via Double-Momentum. *arXiv preprint arXiv:2102.07367*.

Larsen, J.; Hansen, L. K.; Svarer, C.; and Ohlsson, M. 1996. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, 62–71. IEEE.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2.

Li, J.; Gu, B.; and Huang, H. 2020. Improved bilevel model: Fast and optimal algorithm with theoretical guarantee. *arXiv preprint arXiv:2009.00690*.

Liao, R.; Xiong, Y.; Fetaya, E.; Zhang, L.; Yoon, K.; Pitkow, X.; Urtasun, R.; and Zemel, R. 2018. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, 3082–3091. PMLR.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Liu, R.; Gao, J.; Zhang, J.; Meng, D.; and Lin, Z. 2021. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *arXiv preprint arXiv:2101.11517*.

Lorraine, J.; and Duvenaud, D. 2018. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*.

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

Mehra, A.; and Hamm, J. 2019. Penalty method for inversion-free deep bilevel optimization. *arXiv preprint arXiv:1911.03432*.

Nocedal, J.; and Wright, S. 2006. *Numerical optimization*. Springer Science & Business Media.

Okuno, T.; Takeda, A.; and Kawana, A. 2018. Hyperparameter learning via bilevel nonsmooth optimization. *arXiv preprint arXiv:1806.01520*.

Pedregosa, F. 2016. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*.

Sabach, S.; and Shtern, S. 2017. A first order method for solving convex bilevel optimization problems. *SIAM Journal on Optimization*, 27(2): 640–660.

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

Soh, J. W.; Cho, S.; and Cho, N. I. 2020. Meta-transfer learning for zero-shot super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3516–3525.

Solodov, M. 2007. An explicit descent method for bilevel convex optimization. *Journal of Convex Analysis*, 14(2): 227.

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

Tian, Y.; Shen, L.; Su, G.; Li, Z.; and Liu, W. 2020. Alphagan: Fully differentiable architecture search for generative adversarial networks. *arXiv preprint arXiv:2006.09134*.

Tschiatschek, S.; Ghosh, A.; Haug, L.; Devidze, R.; and Singla, A. 2019. Learner-aware teaching: Inverse reinforcement learning with preferences and constraints. *arXiv preprint arXiv:1906.00429*.

Willoughby, R. A. 1979. Solutions of ill-posed problems (an tikhonov and vy arsenin). *SIAM Review*, 21(2): 266.

Wong, C.; Houlsby, N.; Lu, Y.; and Gesmundo, A. 2018. Transfer learning with neural automl. *arXiv preprint arXiv:1803.02780*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. PC-DARTS: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Yadav, C.; and Bottou, L. 2019. Cold case: The lost mnist digits. In *Advances in Neural Information Processing Systems*, 13443–13452.

Yamada, I.; Yukawa, M.; and Yamagishi, M. 2011. Minimizing the Moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, 345–390. Springer.

Yang, J.; Ji, K.; and Liang, Y. 2021. Provably Faster Algorithms for Bilevel Optimization. *arXiv preprint arXiv:2106.04692*.

Yang, Z.; Fu, Z.; Zhang, K.; and Wang, Z. 2018. Convergent reinforcement learning with function approximation: A bilevel optimization perspective.

Yin, H.; Li, D.; Li, X.; and Li, P. 2020. Meta-cotgan: A meta cooperative training paradigm for improving adversarial text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9466–9473.

Zintgraf, L.; Shiarli, K.; Kurin, V.; Hofmann, K.; and Whiteson, S. 2019. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 7693–7702. PMLR.