Truncated Back-propagation for Bilevel Optimization

Amirreza Shaban* **Ching-An Cheng*** **Nathan Hatch** **Byron Boots**
Georgia Institute of Technology *Equal contribution

Abstract

Bilevel optimization has been recently revisited for designing and analyzing algorithms in hyperparameter tuning and meta learning tasks. However, due to its nested structure, evaluating exact gradients for high-dimensional problems is computationally challenging. One heuristic to circumvent this difficulty is to use the approximate gradient given by performing truncated back-propagation through the iterative optimization procedure that solves the lower-level problem. Although promising empirical performance has been reported, its theoretical properties are still unclear. In this paper, we analyze the properties of this family of approximate gradients and establish sufficient conditions for convergence. We validate this on several hyperparameter tuning and meta learning tasks. We find that optimization with the approximate gradient computed using few-step back-propagation often performs comparably to optimization with the exact gradient, while requiring far less memory and half the computation time.

1 INTRODUCTION

Bilevel optimization has been recently revisited as a theoretical framework for designing and analyzing algorithms for hyperparameter optimization [1] and meta learning [2]. Mathematically, these problems can be formulated as a stochastic optimization problem with an equality constraint (see Section 1.1):

$$\begin{aligned} \min_{\lambda} F(\lambda) &:= \mathbb{E}_S \left[f_S(\hat{w}_S^*(\lambda), \lambda) \right] \\ \text{s.t.} \quad \hat{w}_S^*(\lambda) &\approx_\lambda \arg\min_{w} g_S(w, \lambda) \end{aligned} \tag{1}$$

where w and λ are the *parameter* and the *hyperparameter*, F and f_S are the expected and the sampled *upper-level objective*, g_S is the sampled *lower-level objective*, and S is a random variable called the *context*. The notation $\approx_\lambda$ means that $\hat{w}_S^*(\lambda)$ equals the unique return value of a prespecified iterative algorithm (e.g. gradient descent) that approximately finds a local minimum of g_S. This algorithm is part of the problem definition and can also be parametrized by λ (e.g. step size). The motivation to explicitly consider the approximate solution $\hat{w}_S^*(\lambda)$ rather than an exact minimizer w_S^* of g_S is that w_S^* is usually not available in closed form. This setup enables λ to account for the imperfections of the lower-level optimization algorithm.

Solving the bilevel optimization problem in (1) is challenging due to the complicated dependency of the upper-level problem on λ induced by $\hat{w}_S^*(\lambda)$. This difficulty is further aggravated when λ and w are high-dimensional, precluding the use of black-box optimization techniques such as grid/random search [3] and Bayesian optimization [4, 5].

Recently, first-order bilevel optimization techniques have been revisited to solve these problems. These methods rely on an estimate of the Jacobian $\nabla_\lambda \hat{w}_S^*(\lambda)$ to optimize λ. Pedregosa [6] and Gould et al. [7] assume that $\hat{w}_S^*(\lambda) = w_S^*$ and compute $\nabla_\lambda \hat{w}_S^*(\lambda)$ by implicit differentiation. By contrast, Maclaurin et al. [8] and Franceschi et al. [9] treat the iterative optimization algorithm in the lower-level problem as a dynamical system, and compute $\nabla_\lambda \hat{w}_S^*(\lambda)$ by automatic differentiation through the dynamical system. In comparison, the latter approach is less sensitive to the optimality of $\hat{w}_S^*(\lambda)$ and can also learn hyperparameters that control the lower-level optimization process (e.g. step size). However, due to superlinear time or space complexity (see Section 2.2), neither of these methods is applicable when both λ and w are high-dimensional [9].

Few-step reverse-mode automatic differentiation [10, 11] and few-step forward-mode automatic differentiation [9] have recently been proposed as heuristics to address this issue. By ignoring long-term dependencies, the time and space complexities to compute approximate gradients can be greatly reduced. While exciting empirical results have been reported, the theoretical properties of these methods remain unclear.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

In this paper, we study the theoretical properties of these *truncated back-propagation* approaches. We show that, when the lower-level problem is locally strongly convex around $\hat{w}_S^*(\lambda)$, on-average convergence to an ϵ-approximate stationary point is guaranteed by $O(\log 1/\epsilon)$-step truncated back-propagation. We also identify additional problem structures for which asymptotic convergence to an exact stationary point is guaranteed. Empirically, we verify the utility of this strategy for hyperparameter optimization and meta learning tasks. We find that, compared to optimization with full back-propagation, optimization with truncated back-propagation usually shows competitive performance while requiring half as much computation time and significantly less memory.

1.1 Applications

Hyperparameter Optimization The goal of hyperparameter optimization [12, 13] is to find hyperparameters λ for an optimization problem P such that the approximate solution $\hat{w}^*(\lambda)$ of P has low cost $c(\hat{w}^*(\lambda))$ for some cost function c. In general, λ can parametrize both the objective of P and the algorithm used to solve P. This setup is a special case of the bilevel optimization problem (1) where the upper-level objective c does not depend directly on λ. In contrast to meta learning (discussed below), c can be deterministic [9]. See Section 4.2 for examples.

Many low-dimensional problems, such as choosing the learning rate and regularization constant for training neural networks, can be effectively solved with grid search. However, problems with thousands of hyperparameters are increasingly common, for which gradient-based methods are more appropriate [8, 14].

Meta Learning Another important application of bilevel optimization, meta learning (or learning-to-learn) uses statistical learning to optimize an algorithm $\mathcal{A}_\lambda$ over a distribution of tasks $\mathcal{T}$ and contexts S:

$$\min_\lambda \mathbb{E}_\mathcal{T} \mathbb{E}_{S|\mathcal{T}} \left[c_\mathcal{T}(\mathcal{A}_\lambda(S)) \right]. \tag{2}$$

It treats $\mathcal{A}_\lambda$ as a parametric function, with hyperparameter λ, that takes task-specific context information S as input and outputs a decision $\mathcal{A}_\lambda(S)$. The goal of meta learning is to optimize the algorithm's performance $c_\mathcal{T}$ (e.g. the generalization error) across tasks $\mathcal{T}$ through empirical observations. This general setup subsumes multiple problems commonly encountered in the machine learning literature, such as multi-task learning [15, 16] and few-shot learning [17, 18, 19].

Bilevel optimization emerges from meta learning when the algorithm computes $\mathcal{A}_\lambda(S)$ by internally solving a *lower-level* minimization problem with variable w. The motivation to use this class of algorithms is that the lower-level problem can be designed so that, even for tasks $\mathcal{T}$ distant from the training set, $\mathcal{A}_\lambda$ falls back upon a sensible optimization-based approach [20, 11]. By contrast, treating $\mathcal{A}_\lambda$ as a general function approximator relies on the availability of a large amount of meta training data [21, 22].

In other words, the decision is $\mathcal{A}_\lambda(S) = (\hat{w}_S^*(\lambda), \lambda)$ where $\hat{w}_S^*(\lambda)$ is an approximate minimizer of some function $g_S(w, \lambda)$. Therefore, we can identify

$$\mathbb{E}_{\mathcal{T}|S} \left[c_\mathcal{T}(\hat{w}_S^*(\lambda), \lambda) \right] =: f_S(\hat{w}_S^*(\lambda), \lambda) \tag{3}$$

and write (2) as (1).[1] Compared with λ, the lower-level variable w is usually task-specific and fine-tuned based on the given context S. For example, in few-shot learning, a warm start initialization or regularization function (λ) can be learned through meta learning, so that a task-specific network (w) can be quickly trained using regularized empirical risk minimization with few examples S. See Section 4.3 for an example.

2 BILEVEL OPTIMIZATION

2.1 Setup

Let $\lambda \in \mathbb{R}^N$ and $w \in \mathbb{R}^M$. We consider solving (1) with first-order methods that sample S (like stochastic gradient descent) and focus on the problem of computing the gradients for a given S. Therefore, we will simplify the notation below by omitting the dependency of variables and functions on S and λ (e.g. we write $\hat{w}_S^*(\lambda)$ as $\hat{w}^*$ and g_S as g). We use d_x to denote the total derivative with respect to a variable x, and ∇_x to denote the partial derivative, with the convention that $\nabla_\lambda f \in \mathbb{R}^N$ and $\nabla_\lambda \hat{w}^* \in \mathbb{R}^{N \times M}$.

To optimize λ, stochastic first-order methods use estimates of the gradient $\mathrm{d}_\lambda f = \nabla_\lambda f + \nabla_\lambda \hat{w}^* \nabla_{\hat{w}^*} f$. Here we assume that both $\nabla_\lambda f \in \mathbb{R}^N$ and $\nabla_{\hat{w}^*} f \in \mathbb{R}^M$ are available through a stochastic first-order oracle, and focus on the problem of computing the matrix-vector product $\nabla_\lambda \hat{w}^* \nabla_{\hat{w}^*} f$ when both λ and w are high-dimensional.

2.2 Computing the hypergradient

Like [8, 9], we treat the iterative optimization algorithm that solves the lower-level problem as a dynamical system. Given an initial condition $w_0 = \Xi_0(\lambda)$ at $t = 0$, the update rule can be written as[2]

$$w_{t+1} = \Xi_{t+1}(w_t, \lambda), \qquad \hat{w}^* = w_T \tag{4}$$

[1] We have replaced $\mathbb{E}_\mathcal{T} \mathbb{E}_{S|\mathcal{T}}$ with $\mathbb{E}_S \mathbb{E}_{\mathcal{T}|S}$, which is valid since both describe the expectation over the joint distribution. The algorithm $\mathcal{A}_\lambda$ only perceives S, not $\mathcal{T}$.

[2] For notational simplicity, we consider the case where w_t is the state of (4); our derivation can be easily generalized to include other internal states, e.g. momentum.

Table 1: Comparison of the additional time and space to compute $\mathrm{d}_\lambda f = \nabla_\lambda f + \nabla_\lambda \hat{w}^* \nabla_{\hat{w}^*} f$, where $\lambda \in \mathbb{R}^N$, $w \in \mathbb{R}^M$, and $c = c(M, N)$ is the time complexity to compute the transition function Ξ. †Checkpointing doubles the constant in time complexity, compared with other approaches.

Method	Time	Space	Exact
FMD	$O(cNT)$	$O(MN)$	✓
RMD	$O(cT)$	$O(MT)$	✓
Checkpointing every $\sqrt{T}$ steps†	$O(cT^\dagger)$	$O(M\sqrt{T})$	✓
K-RMD	$O(cK)$	$O(MK)$	

in which Ξ_t defines the transition and and T is the number iterations performed. For example, in gradient descent, $\Xi_{t+1}(w_t, \lambda) = w_t - \gamma_t(\lambda)\nabla_w g(w_t, \lambda)$, where $\gamma_t(\lambda)$ is the step size.

By unrolling the iterative update scheme (4) as a computational graph, we can view $\hat{w}^*$ as a function of λ and compute the required derivative $\mathrm{d}_\lambda f$ [23]. Specifically, it can be shown by the chain rule[3]

$$\mathrm{d}_\lambda f = \nabla_\lambda f + \textstyle\sum_{t=0}^{T} B_t A_{t+1} \cdots A_T \nabla_{\hat{w}^*} f \qquad (5)$$

where $A_{t+1} = \nabla_{w_t} \Xi_{t+1}(w_t, \lambda)$, $B_{t+1} = \nabla_\lambda \Xi_{t+1}(w_t, \lambda)$ for $t \geq 0$, and $B_0 = \mathrm{d}_\lambda \Xi_0(\lambda)$.

The computation of (5) can be implemented either in reverse mode or forward mode [9]. Reverse-mode differentiation (RMD) computes (5) by back-propagation:

$$\begin{aligned} \alpha_T &= \nabla_{\hat{w}^*} f, \quad h_T = \nabla_\lambda f, \\ h_{t-1} &= h_t + B_t \alpha_t, \quad \alpha_{t-1} = A_t \alpha_t \end{aligned} \qquad (6)$$

and finally $\mathrm{d}_\lambda f = h_{-1}$. Forward-mode differentiation (FMD) computes (5) by forward propagation:

$$\begin{aligned} Z_0 &= B_0, \quad Z_{t+1} = Z_t A_{t+1} + B_{t+1}, \\ \mathrm{d}_\lambda f &= Z_T \nabla_{\hat{w}^*} f + \nabla_\lambda f \end{aligned} \qquad (7)$$

The choice between RMD and FMD is a trade-off based on the size of $w \in \mathbb{R}^M$ and $\lambda \in \mathbb{R}^N$ (see Table 1 for a comparison). For example, one drawback of RMD is that all the intermediate variables $\{w_t \in \mathbb{R}^M\}_{t=1}^T$ need to be stored in memory in order to compute A_t and B_t in the backward pass. Therefore, RMD is only applicable when MT is small, as in [20]. Checkpointing [24] can reduce this to $M\sqrt{T}$, but it *doubles* the computation time. Complementary to RMD, FMD propagates the matrix $Z_t \in \mathbb{R}^{M \times N}$ in line with the forward evaluation of the dynamical system (4), and does not require any additional memory to save the intermediate variables. However, propagating the matrix Z_t instead of vectors requires memory of size MN and is N-times slower compared with RMD.

[3]Note that this assumes g is twice differentiable.

3 TRUNCATED BACK-PROPAGATION

In this paper, we investigate approximating (5) with partial sums, which was previously proposed as a heuristic for bilevel optimization ([10] Eq. 3, [11] Eq. 2). Formally, we perform K-step truncated back-propagation (K-RMD) and use the intermediate variable h_{T-K} to construct an approximate gradient:

$$h_{T-K} = \nabla_\lambda f + \textstyle\sum_{t=T-K+1}^{T} B_t A_{t+1} \cdots A_T \nabla_{\hat{w}^*} f \quad (8)$$

This approach requires storing only the last K iterates w_t, and it also saves computation time. Note that K-RMD can be combined with checkpointing for further savings, although we do not investigate this.

3.1 General properties

We first establish some intuitions about why using K-RMD to optimize λ is reasonable. While building up an approximate gradient by truncating back-propagation in general optimization problems can lead to large bias, the bilevel optimization problem in (1) has some nice structure. Here we show that if the lower-level objective g is locally strongly convex around $\hat{w}^*$, then the bias of h_{T-K} can be exponentially small in K. That is, choosing a small K would suffice to give a good gradient approximation in finite precision. The proof is given in Appendix A.

Proposition 3.1. *Assume g is β-smooth, twice differentiable, and locally α-strongly convex in w around $\{w_{T-K-1}, \ldots, w_T\}$. Let $\Xi_{t+1}(w_t, \lambda) = w_t - \gamma \nabla_w g(w_t, \lambda)$. For $\gamma \leq \frac{1}{\beta}$, it holds*

$$\|h_{T-K} - \mathrm{d}_\lambda f\| \leq 2^{T-K+1}(1 - \gamma\alpha)^K \|\nabla_{\hat{w}^*} f\| M_B \quad (9)$$

where $M_B = \max_{t \in \{0,\ldots,T-K\}} \|B_t\|$. In particular, if g is globally α-strongly convex, then

$$\|h_{T-K} - \mathrm{d}_\lambda f\| \leq \tfrac{(1-\gamma\alpha)^K}{\gamma\alpha} \|\nabla_{\hat{w}^*} f\| M_B. \qquad (10)$$

Note $0 \leq (1 - \gamma\alpha) < 1$ since $\gamma \leq \frac{1}{\beta} \leq \frac{1}{\alpha}$. Therefore, Proposition 3.1 says that if $\hat{w}^*$ converges to the *neighborhood* of a strict local minimum of the lower-level optimization, then the bias of using the approximate gradient of K-RMD decays exponentially in K. This exponentially decaying property is the main reason why using h_{T-K} to update the hyperparameter λ works.

Next we show that, when the lower-level problem g is second-order continuously differentiable, $-h_{T-K}$ actually is a sufficient descent direction. This is a much stronger property than the small bias shown in Proposition 3.1, and it is critical in order to prove convergence to exact stationary points (cf. Theorem 3.4). To build intuition, here we consider a simpler problem where g is globally strongly convex and $\nabla_\lambda f = 0$. These assumptions will be relaxed in the next subsection.

Lemma 3.2. *Let g be globally strongly convex and $\nabla_\lambda f = 0$. Assume g is second-order continuously differentiable and B_t has full column rank for all t. Let $\Xi_{t+1}(w_t, \lambda) = w_t - \gamma \nabla_w g(w_t, \lambda)$. For all $K \geq 1$, with T large enough and γ small enough, there exists $c > 0$, s.t. $h_{T-K}^\top \mathrm{d}_\lambda f \geq c \|\nabla_{\hat{w}^*} f\|^2$. This implies h_{T-K} is a sufficient descent direction, i.e. $h_{T-K}^\top \mathrm{d}_\lambda f \geq \Omega(\|\mathrm{d}_\lambda f\|^2)$.*

The full proof of this non-trivial result is given in Appendix B. Here we provide some ideas about why it is true. First, by Proposition 3.1, we know the bias decays exponentially. However, this alone is not sufficient to show that $-h_{T-K}$ is a sufficient descent direction. To show the desired result, Lemma 3.2 relies on the assumption that g is second-order continuously differentiable and the fact that using gradient descent to optimize a well-conditioned function has linear convergence [25]. These two new structural properties further reduce the bias in Proposition 3.1 and lead to Lemma 3.2. Here the full rank assumption for B_t is made to simplify the proof. We conjecture that this condition can be relaxed when $K > 1$. We leave this to future work.

3.2 Convergence

With these insights, we analyze the convergence of bilevel optimization with truncated back-propagation. Using Proposition 3.1, we can immediately deduce that optimizing λ with h_{T-K} converges on-average to an ϵ-approximate stationary point. Let $\nabla F(\lambda_\tau)$ denote the hypergradient in the τth iteration.

Theorem 3.3. *Suppose F is smooth and bounded below, and suppose there is $\epsilon < \infty$ such that $\|h_{T-K} - \mathrm{d}_\lambda f\| \leq \epsilon$. Using h_{T-K} as a stochastic first-order oracle with a decaying step size $\eta_\tau = O(1/\sqrt{\tau})$ to update λ with gradient descent, it follows after R iterations,*

$$\mathbb{E}\left[\sum_{\tau=1}^{R} \frac{\eta_\tau \|\nabla F(\lambda_\tau)\|^2}{\sum_{\tau=1}^{R} \eta_\tau}\right] \leq \widetilde{O}\left(\epsilon + \frac{\epsilon^2 + 1}{\sqrt{R}}\right).$$

That is, under the assumptions in Proposition 3.1, learning with h_{T-K} converges to an ϵ-approximate stationary point, where $\epsilon = O((1-\gamma\alpha)^{-K})$.

We see that the bias becomes small as K increases. As a result, it is sufficient to perform K-step truncated back-propagation with $K = O(\log 1/\epsilon)$ to update λ.

Next, using Lemma 3.2, we show that the bias term in Theorem 3.3 can be removed if the problem is more structured. As promised, we relax the simplifications made in Lemma 3.2 into assumptions 2 and 3 below and only assume g is locally strongly convex.

Theorem 3.4. *Under the assumptions in Proposition 3.1 and Theorem 3.3, if in addition*

1. *g is second-order continuously differentiable*
2. *B_t has full column rank around w_T*
3. *$\nabla_\lambda f^\top (\mathrm{d}_\lambda f + h_{T-K} - \nabla_\lambda f) \geq \Omega(\|\nabla_\lambda f\|^2)$*
4. *the problem is deterministic (i.e. $F = f$)*

then for all $K \geq 1$, with T large enough and γ small enough, the limit point is an exact stationary point, i.e. $\lim_{\tau\to\infty} \|\nabla F(\lambda_\tau)\| = 0$.

Theorem 3.4 shows that if the partial derivative $\nabla_\lambda f$ does not interfere strongly with the partial derivative computed through back-propagating the lower-level optimization procedure (assumption 3), then optimizing λ with h_{T-K} converges to an *exact* stationary point. This is a very strong result for an interesting special case. It shows that even with one-step back-propagation h_{T-1}, updating λ can converge to a stationary point.

This non-interference assumption unfortunately is necessary; otherwise, truncating the full RMD leads to constant bias, as we show below (proved in Appendix E).

Theorem 3.5. *There is a problem, satisfying all but assumption 3 in Theorem 3.4, such that optimizing λ with h_{T-K} does not converge to a stationary point.*

Note however that the non-interference assumption is satisfied when $\nabla_\lambda f = 0$, i.e. when the upper-level problem does not directly depend on the hyperparameter. This is the case for many practical applications: e.g. hyperparameter optimization, meta-learning regularization models, image desnosing [26, 14], data hypercleaning [9], and task interaction [27].

3.3 Relationship with implicit differentiation

The gradient estimate h_{T-K} is related to implicit differentiation, which is a classical first-order approach to solving bilevel optimization problems [12, 13]. Assume g is second-order continuously differentiable and that its optimal solution uniquely exists such that $w^* = w^*(\lambda)$. By the implicit function theorem [28], the total derivative of f with respect to λ can be written as

$$\mathrm{d}_\lambda f = \nabla_\lambda f - \nabla_{\lambda,w} g \nabla_{w,w}^{-1} g \nabla_{\hat{w}^*} f \tag{11}$$

where all derivatives are evaluated at $(w^*(\lambda), \lambda)$ and $\nabla_{\lambda,w} g = \nabla_\lambda(\nabla_w g) \in \mathbb{R}^{N\times M}$.

Here we show that, in the limit where $\hat{w}^*$ converges to w^*, h_{T-K} can be viewed as approximating the matrix inverse in (11) with an order-K Taylor series. This can be seen from the next proposition.

Proposition 3.6. *Under the assumptions in Proposition 3.1, suppose w_t converges to a stationary point w^*. Let $A_\infty = \lim_{t\to\infty} A_t$ and $B_\infty = \lim_{t\to\infty} B_t$. For $\gamma < \frac{1}{\beta}$, it satisfies that*

$$-\nabla_{\lambda,w} g \nabla_{w,w}^{-1} g = B_\infty \textstyle\sum_{k=0}^{\infty} A_\infty^k \tag{12}$$

By Proposition 3.6, we can write $\mathrm{d}_\lambda f$ in (11) as

$$\begin{aligned}\mathrm{d}_\lambda f &= \nabla_\lambda f - \nabla_{\lambda,w} g \nabla_{w,w}^{-1} g \nabla_{\hat{w}^*} f \\ &= h_{T-K} + B_\infty \sum_{k=K}^{\infty} A_\infty^k \nabla_{\hat{w}^*} f\end{aligned}$$

That is, h_{T-K} captures the first K terms in the Taylor series, and the residue term has an upper bound as in Proposition 3.1.

Given this connection, we can compare the use of h_{T-K} and approximating (11) using K steps of conjugate gradient descent for high-dimensional problems [6]. First, both approaches require local strong-convexity to ensure a good approximation. Specifically, let $\kappa = \frac{\beta}{\alpha} > 0$ locally around the limit. Using h_{T-K} has a bias in $O((1-\frac{1}{\kappa})^K)$, whereas using (11) and inverting the matrix with K iterations of conjugate gradient has a bias in $O((1-\frac{1}{\sqrt{\kappa}})^K)$ [29]. Therefore, when w^* is available, solving (11) with conjugate gradient descent is preferable. However, in practice, this is hardly true. When an approximate solution $\hat{w}^*$ to the lower-level problem is used, adopting (11) has no control on the approximate error, nor does it necessarily yield a descent direction. On the contrary, h_{T-K} is based on Proposition 3.1, which uses a weaker assumption and does not require the convergence of w_t to a stationary point. Truncated back-propagation can also optimize the hyperparameters that control the lower-level optimization process, which the implicit differentiation approach cannot do.

4 EXPERIMENTS

4.1 Toy problem

Consider the following simple problem for $\lambda, w \in \mathbb{R}^2$:

$$\begin{aligned}&\min_\lambda \|\hat{w}^*\|^2 + 10\|\sin(\hat{w}^*)\|^2 =: f(\hat{w}^*, \lambda) \\ &\text{s.t. } \hat{w}^* \approx \arg\min_w \tfrac{1}{2}(w-\lambda)^\top G (w-\lambda) =: g(w, \lambda)\end{aligned}$$

where $\|\cdot\|$ is the ℓ_2 norm, sine is applied elementwise, $G = \mathrm{diag}(1, \frac{1}{2})$, and we define $\hat{w}^*$ as the result of $T = 100$ steps of gradient descent on g with learning rate $\gamma = 0.1$, initialized at $w_0 = (2, 2)$. A plot of $f(\cdot, \lambda)$ is shown in Figure. 1. We will use this problem to visualize the theorems and explore the empirical properties of truncated back-propagation.

This deterministic problem satisfies all of the assumptions in the previous section, particularly those of Theorem 3.4: g is 1-smooth and $\frac{1}{2}$-strongly convex, with

$$B_{t+1} = \nabla_\lambda [w_t - \gamma \nabla_w g(w_t, \lambda)] = \gamma G$$

and $B_0 = 0$. Although f is somewhat complicated, with many saddle points, it satisfies the non-interference assumption because $\nabla_\lambda f = 0$.

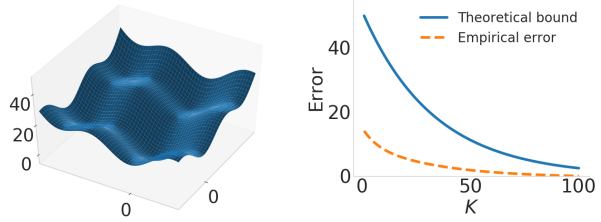

Figure 1: Graph of f and visualization of Prop. 3.1.

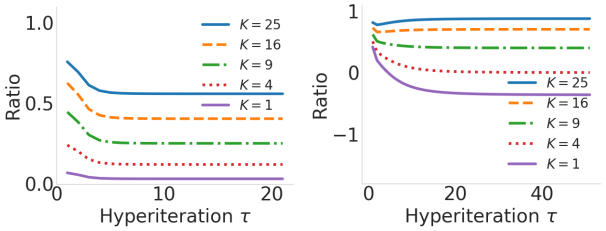

Figure 2: The ratio $h_{T-K}^\top \mathrm{d}_\lambda f / \|\mathrm{d}_\lambda f\|^2$ at various λ_τ, for f and $\widetilde{f}$ respectively.

Figure 1 visualizes Proposition 3.1 by plotting the approximation error $\|h_{T-K} - \mathrm{d}_\lambda f\|$ and the theoretical bound $\frac{(1-\gamma\alpha)^K}{\gamma\alpha}\|\nabla_{\hat{w}^*} f\| M_B$ at $\lambda = (1, 1)$. For this problem, $\alpha = \frac{1}{2}$, $M_B = \|\gamma G\| = \gamma$, and $\nabla_{\hat{w}^*} f$ can be found analytically from $\hat{w}^* = Cw_0 + (I - C)\lambda$, where $C = (I - \gamma G)^T$. Figure 4 (left) plots the iterates λ_τ when optimizing f using 1-RMD and a decaying meta-learning rate $\eta_\tau = \frac{\eta_0}{\sqrt{\tau}}$.[4] In comparison with the true gradient $\mathrm{d}_\lambda f$ at these points, we see that h_{T-1} is indeed a descent direction. Figure 2 (left) visualizes this in a different way, by plotting $h_{T-K}^\top \mathrm{d}_\lambda f / \|\mathrm{d}_\lambda f\|^2$ for various K at each point λ_τ along the $K = 1$ trajectory. By Lemma 3.2, this ratio stays well away from zero.

To demonstrate the biased convergence of Theorem 3.3, we break assumption 3 of Theorem 3.4 by changing the upper objective to $\widetilde{f}(\hat{w}^*, \lambda) := f(\hat{w}^*, \lambda) + 5\|\lambda - (1, 0)\|^2$ so that $\nabla_\lambda \widetilde{f} \neq 0$. The guarantee of Lemma 3.2 no longer applies, and we see in Figure 2 (right) that $h_{T-K}^\top \mathrm{d}_\lambda f / \|\mathrm{d}_\lambda f\|^2$ can become negative. Indeed, Figure 3 shows that optimizing $\widetilde{f}$ with h_{T-1} converges to a suboptimal point. However, it also shows that using larger K rapidly decreases the bias.

For the original objective f, Theorem 3.4 guarantees exact convergence. Figure 4 shows optimization trajectories for various K, and a log-scale plot of their convergence rates. Note that, because the lower-level problem cannot be perfectly solved within T steps, the optimal λ is offset from the origin. Truncated back-propagation can handle this, but it breaks the assumptions required by the implicit differentiation approach to bilevel optimization.

[4] Because $\|h_{T-K}\|$ varies widely with K, we tune η_0 to ensure that the first update $\eta_1 h_{T-K}(\lambda_1)$ has norm 0.6.

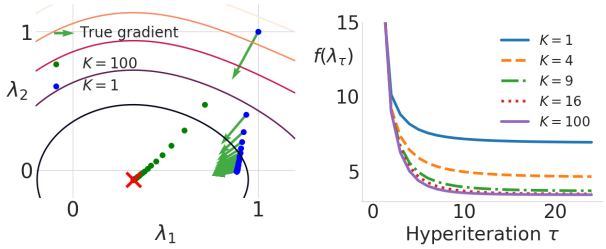

Figure 3: Biased convergence for $\widetilde{f}$. The red X marks the optimal λ.

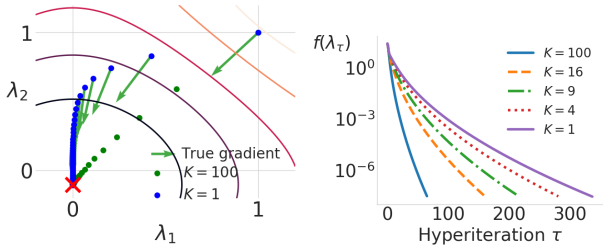

Figure 4: Convergence for f.

4.2 Hyperparameter optimization problems

4.2.1 Data hypercleaning

In this section, we evaluate K-RMD on a hyperparameter optimization problem. The goal of data hypercleaning [9] is to train a linear classifier for MNIST [30], with the complication that half of our training labels have been corrupted. To do this with hyperparameter optimization, let $W \in \mathbb{R}^{10\times 785}$ be the weights of the classifier, with the outer objective f measuring the cross-entropy loss on a cleanly labeled validation set. The inner objective is defined as *weighted* cross-entropy training loss plus regularization:

$$g(W, \lambda) = \sum_{i=1}^{5000} -\sigma(\lambda_i) \log(e_{y_i}^\top W x_i) + 0.001\|W\|_F^2$$

where (x_i, y_i) are the training examples, σ denotes the sigmoid function, $\lambda_i \in \mathbb{R}$, and $\|\cdot\|_F$ is the Frobenius norm. We optimize λ to minimize validation loss, presumably by decreasing the weight of the corrupted examples. The optimization dimensions are $|\lambda| = 5000$, $|W| = 7850$. Franceschi et al. [9] previously solved this problem with full RMD, and it happens to satisfy many of our theoretical assumptions, making it an interesting case for empirical study.[5]

We optimize the lower-level problem g through $T = 100$ steps of gradient descent with $\gamma = 1$ and consider how

[5]We have reformulated the constrained problem from [9] as an unconstrained one that more closely matches our theoretical assumptions. For the same reason, we regularized g to make it strongly convex. Finally, we do not retrain on the hypercleaned training + validation data. This is because, for our purposes, comparing the performance of $\hat{w}^*$ across K is sufficient.

Table 2: Hypercleaning metrics after 1000 hyperiters.

K	Test Acc.	Val. Acc.	Val. Loss	F1
1	87.50	89.32	0.413	0.85
5	88.05	89.90	0.383	0.89
25	88.12	89.94	0.382	0.89
50	88.17	90.18	0.381	0.89
100	88.33	90.24	0.380	0.88

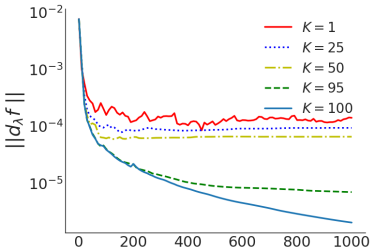

Figure 5: $\|\mathrm{d}_\lambda f\|$ vs. hyperiteration for hypercleaning.

adjusting K changes the performance of K-RMD.[6] Our hypothesis is that K-RMD for small K works almost as well as full RMD in terms of validation and test accuracy, while requiring less time and far less memory. We also hypothesize that K-RMD does almost as well as full RMD in identifying which samples were corrupted [9]. Because our formulation of the problem is unconstrained, the weights $\sigma(\lambda_i)$ are never exactly zero. However, we can calculate an F1 score by setting a threshold on λ: if $\sigma(\lambda_i) < \sigma(-3) \approx 0.047$, then the hyper-cleaner has marked example i as corrupted.[7]

Table 2 reports these metrics for various K. We see that 1-RMD is somewhat worse than the others, and that validation loss (the outer objective f) decreases with K more quickly than generalization error. The F1 score is already maximized at $K = 5$. These preliminary results indicate that in situations with limited memory, K-RMD for small K (e.g. $K = 5$) may be a reasonable fallback: it achieves results close to full backprop, and it runs about twice as fast.

From a theoretical optimization perspective, we wonder whether K-RMD converges to a stationary point of f. Data hypercleaning satisfies all of the assumptions of Theorem 3.4 except that B_t is not full column rank (since $M < N$). In particular, the validation loss f is deterministic and satisfies $\nabla_\lambda f = 0$. Figure 5 plots the norm of the true gradient $\mathrm{d}_\lambda f$ on a log scale at the K-RMD iterates for various K. We see that, despite satisfying almost all assumptions, this problem exhibits biased convergence. The limit of $\|\mathrm{d}_\lambda f\|$ decreases slowly with K, but recall from Table 2 that practical metrics improve more quickly.

[6]See Appendix G.1 for more experimental setup.

[7]F1 scores for other choices of the threshold were very similar. See Appendix G.1 for details.

4.2.2 Task interaction

We next consider the problem of multitask learning [27]. Similar to [9], we formulate this as a hyperparameter optimization problem as follows. The lower-level objective $g(w, \{C, \rho\})$ learns V different linear models with parameter set $w = \{w_v\}_{v=1}^V$:

$$l(w) + \sum_{1 \le i,j \le K} C_{ij} \|w_i - w_j\|^2 + \rho \sum_{v=1}^{V} \|w_v\|^2$$

where $l(w)$ is the training loss of the multi-class linear logistic regression model, ρ is a regularization constant, and C is a nonnegative, symmetric hyperparameter matrix that encodes the similarity between each pair of tasks. After 100 iterations of gradient descent with learning rate 0.1, this yields $\hat{w}^*$. The upper-level objective $c(\hat{w}^*)$ estimates the linear regression loss of the learned model $\hat{w}^*$ on a validation set. Presumably, this will be improved by tuning C to reflect the true similarities between the tasks. The tasks that we consider are image recognition trained on very small subsets of the datasets CIFAR-10 and CIFAR-100.[8]

From an optimization standpoint, we are most interested in the upper-level loss on the validation set, since that is what is directly optimized, and its value is a good indication of the performance of the inexact gradient. Figure 6 plots this learning curve along with two other metrics of theoretical interest: norm of the true gradient, and cosine similarity between the true and approximate gradients. In CIFAR100, the validation error and gradient norm plots show that K-RMD converges to an approximate stationary point with a bias that rapidly decreases as K increases, agreeing with Proposition 3.1. Also, we find that negative values exist in the cosine similarity of 1-RMD, which implies that not all the assumptions in Theorem 3.4 hold for this problem (e.g. B_t might not be full rank, or the the inner problem might not be locally strong convex around $\hat{w}^*$.) In CIFAR10, some unusal behavior happens. For $K > 1$, the truncated gradient and the full gradient directions eventually become almost the same. We believe this is a very interesting observation but beyond the scope of the paper to explain.

In Table 3, we report the testing accuracy over 10 trials. While in general increasing the number of back-propagation steps improves accuracy, the gaps are small. A thorough investigation of the relationship between convergence and generalization is an interesting open question of both theoretical and practical importance.

4.3 Meta-learning: One-shot classification

The aim of this experiment is to evaluate the performance of truncated back-propagation in multi-task, stochastic optimization problems. We consider in particular the one-shot classification problem [20], where each task $\mathcal{T}$ is a k-way classification problem and the goal is learn a hyperparameter λ such that each task can be solved with few training samples.

[8] See Appendix G.2 for more details.

Table 3: Test accuracy for task interaction. Few-step K-RMD achieves similar performance as full RMD.

	Method	Avg. Acc.	Avg. Iter.	Sec/iter.
CIFAR-10	1-RMD	61.11 ± 1.23	3300	0.8
	5-RMD	61.33 ± 1.08	4950	1.3
	25-RMD	61.31 ± 1.24	4825	1.4
	Full RMD	61.28 ± 1.21	4500	2.2
CIFAR-100	1-RMD	34.37 ± 0.63	7440	1.0
	5-RMD	34.34 ± 0.68	8805	1.4
	25-RMD	34.51 ± 0.69	8660	1.6
	Full RMD	34.70 ± 0.64	5670	2.8

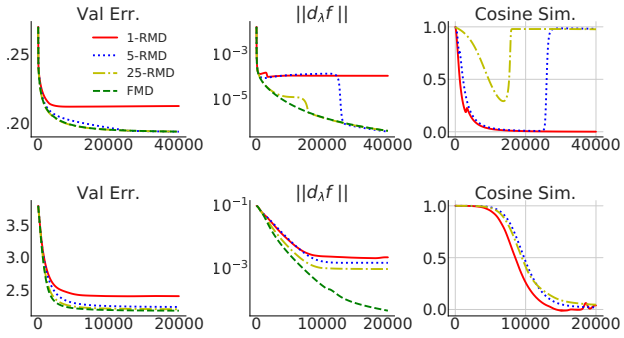

Figure 6: Upper-level objective loss (first column), norm of the exact gradient (second column), and cosine similarity (last column) vs. hyper-iteration on CIFAR10 (first row) and CIFAR100 (second row) datasets.

In each hyper-iteration, we sample a task, a training set, and a validation set as follows: First, k classes are randomly chosen from a pool of classes to define the sampled task $\mathcal{T}$. Then the training set $S = \{(x_i, y_i)\}_{i=1}^k$ is created by randomly drawing one training example (x_i, y_i) from each of the k classes. The validation set Q is constructed similarly, but with more examples from each class. The lower-level objective $g_S(w, \lambda)$ is

$$\textstyle\sum_{(x_i,y_i)\in S} l(nn(x_i; w, \lambda), y_i) + \sum_{j=1}^{V} \rho_j ||w_j - c_j||^2$$

where $l(\cdot, \cdot)$ is the k-way cross-entropy loss, and $nn(\cdot; w, \lambda)$ is a deep neural network parametrized by $w = \{w_1, \ldots, w_V\}$ and optionally hyperparameter λ. To prevent overfitting in the lower-level optimization, we regularize each parameter w_j to be close to center c_j with weight $\rho_j > 0$. Both c_j and ρ_j are hyperparameters, as well as the inner learning rate γ. The upper-level objective is the loss of the trained network on the sampled validation set Q. In contrast to other experiments, this is a stochastic optimization problem. Also, $\mathcal{A}_\lambda(S)(x_i) = nn(x_i; \hat{w}^*, \lambda)$ depends directly on the hyperparameter λ, in addition to the indirect dependence through $\hat{w}^*$ (i.e. $\nabla_\lambda f \neq 0$).

Table 4: Results for one-shot learning on Omniglot dataset. K-RMD reaches similar performance as full RMD, is considerably faster, and requires less memory.

Method	Accuracy	iter.	Sec/iter.
1-RMD	95.6	5K	0.4
10-RMD	96.3	5K	0.7
25-RMD	96.1	5K	1.3
Full RMD	95.8	5K	2.2
1-RMD	97.7	15K	0.4
10-RMD	97.8	15K	0.7
Short horizon	96.6	15K	0.1

We use the Omniglot dataset [31] and a similar neural network as used in [20] with small modifications. Please refer to Appendix G.3 for more details about the model and the data splits. We set $T = 50$ and optimize over the hyperparameter $\lambda = \{\lambda_{l_1}, \lambda_{l_2}, c, \rho, \gamma\}$. The average accuracy of each model is evaluated over 120 randomly sampled training and validation sets from the meta-testing dataset. For comparison, we also try using full RMD with a very short horizon $T = 1$, which is common in recent work on few-shot learning [20].

The statistics are shown in Table 4 and the learning curves in Figure 7. In addition to saving memory, all truncated methods are faster than full RMD, sometimes even five times faster. These results suggest that running few-step back-propagation with more hyper-iterations can be more efficient than the full RMD. To support this hypothesis, we also ran 1-RMD and 10-RMD for an especially large number of hyper-iterations (15k). Even with this many hyper-iterations, the total runtime is less than full RMD with 5000 iterations, and the results are significantly improved. We also find that while using a short horizon ($T = 1$) is faster, it achieves a lower accuracy at the same number of iterations.

Finally, we verify some of our theorems in practice. Figure 7 (fourth plot) shows that when the lower-level problem is regularized, the relative ℓ_2 error between the K-RMD approximate gradient and the exact gradient decays exponentially as K increases. This was guaranteed by Proposition 3.1. However, this exponential decay is not seen for the non-regularized model ($\rho = 0$). This suggests that the local strong convexity assumption is essential in order to have exponential decay in practice. Figure 7 (third plot) shows the cosine similarity between the inexact gradient and full gradient over the course of meta-training. Note that the cosine similarity measures are always positive, indicating that the inexact gradients are indeed descent directions. It also seems that the cosine similarities show a slight decay over time.

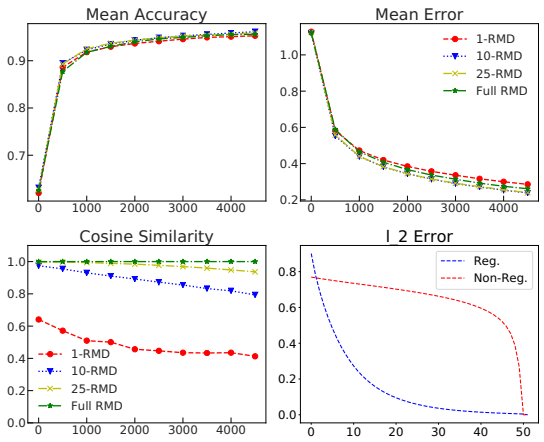

Figure 7: Omniglot results. **Plots 1 and 2:** Test accuracy and val. error vs. number of hyper-iterations for different RMD depths. K-RMD methods show similar performance as the full RMD. **Plot 3:** Cosine similarity between inexact gradient and full RMD over hyper-iterations. **Plot 4:** Relative ℓ_2 error of inexact gradient and full RMD vs. reverse depth. Regularized version shows exponential decay.

5 CONCLUSION

We analyze K-RMD, a first-order heuristic for solving bilevel optimization problems when the lower-level optimization is itself approximated in an iterative way. We show that K-RMD is a valid alternative to full RMD from both theoretical and empirical standpoints. Theoretically, we identify sufficient conditions for which the hyperparameters converge to an approximate or exact stationary point of the upper-level objective. The key observation is that when $\hat{w}^*$ is near a strict local minimum of the lower-level objective, gradient approximation error decays exponentially with reverse depth. Empirically, we explore the properties of this optimization method with four proof-of-concept experiments. We find that although exact convergence appears to be uncommon in practice, the performance of K-RMD is close to full RMD in terms of application-specific metrics (such as generalization error). It is also roughly twice as fast. These results suggest that in hyperparameter optimization or meta learning applications with memory constraints, truncated back-propagation is a reasonable choice.

Our experiments use a modest number of parameters M, hyperparameters N, and horizon length T. This is because we need to be able to calculate both K-RMD and full RMD in order to compare their performance. One promising direction for future research is to use K-RMD for bilevel optimization problems that require powerful function approximators at both levels of optimization. Truncated RMD makes this approach feasible and enables comparing bilevel optimization to other meta-learning methods on difficult benchmarks.

References

[1] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326, 2012.

[2] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. A bridge between hyperparameter optimization and larning-to-learn. *NIPS 2017 Workshop on Meta-learning*, 2017.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[4] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010.

[5] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[6] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning*, pages 737–746, 2016.

[7] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

[8] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

[9] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on International Conference on Machine Learning*, 2017.

[10] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International Conference on Machine Learning*, pages 2952–2960, 2016.

[11] Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations*, 2018.

[12] Jan Larsen, Lars Kai Hansen, Claus Svarer, and M Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing [1996] VI. IEEE Signal Processing Society Workshop*, pages 62–71. IEEE, 1996.

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

[14] Yunjin Chen, Rene Ranftl, and Thomas Pock. Insights into analysis operator learning: From patch-based sparse models to higher order mrfs. *IEEE Transactions on Image Processing*, 23(3): 1060–1072, 2014.

[15] Rich Caruana. Multitask learning. In *Learning to learn*, pages 95–133. Springer, 1998.

[16] Rajeev Ranjan, Vishal M Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[17] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[18] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[19] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

[21] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[22] Ke Li and Jitendra Malik. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

[23] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine

learning: A survey. *Journal of Machine Learning Research*, 18:153:1–153:43, 2017.

[24] Laurent Hascoet and Mauricio Araya-Polo. Enabling user-driven checkpointing strategies in reverse-mode automatic differentiation. *arXiv preprint cs/0606042*, 2006.

[25] Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[26] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867. IEEE, 2005.

[27] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(Apr):615–637, 2005.

[28] Walter Rudin. *Principles of Mathematical Analysis*, volume 3. New York: McGraw-Hill, 1964.

[29] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

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

[31] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[32] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge University Press, 1990.

[33] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.