LOOKAHEAD CONVERGES TO STATIONARY POINTS OF SMOOTH NON-CONVEX FUNCTIONS

Jianyu Wang[⋆†] *Vinayak Tantia*[†] *Nicolas Ballas*[†] *Michael Rabbat*[†]

[⋆]Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, USA
[†]Facebook AI Research, Montreal, Canada
Email: jianyuw1@andrew.cmu.edu, {tantia, ballasn, mikerabbat}@fb.com

ABSTRACT

The Lookahead optimizer [Zhang et al., 2019] was recently proposed and demonstrated to improve performance of stochastic first-order methods for training deep neural networks. Lookahead can be viewed as a two time-scale algorithm, where the fast dynamics (inner optimizer) determine a search direction and the slow dynamics (outer optimizer) perform updates by moving along this direction. We prove that, with appropriate choice of step-sizes, Lookahead converges to a stationary point of smooth non-convex functions. Although Lookahead is described and implemented as a serial algorithm, our analysis is based on viewing Lookahead as a multi-agent optimization method with two agents communicating periodically.

Index Terms— Lookahead optimizer, deep learning, stochastic non-convex optimization.

1. INTRODUCTION

Deep neural networks (DNNs) are essential contemporary tools for machine learning tasks across a range of domains, including speech recognition, computer vision, and natural language understanding [1, 2, 3]. Training DNNs for supervised learning tasks essentially involves solving a finite-sum optimization problem of the form,

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}), \qquad \text{with } f(\boldsymbol{x}) := \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{x}), \tag{1}$$

where f_i quantifies the loss of the model being trained with respect to the ith training sample when the model has parameters $\boldsymbol{x}$, for a model with d parameters and training dataset with m training samples. The objective of training is to find model parameters $\boldsymbol{x}$ which make the loss as small as possible on average over the entire training set. In the training of DNNs, one challenge is that the functions f_i are not convex. Moreover, the problem dimensions d and m may both be very large.

The most widely-used training methods make use of stochastic gradients. For example, *stochastic gradient descent* (SGD) performs updates

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k g(\boldsymbol{x}_k;\xi_k),$$

where γ_k is the learning rate and $g(\boldsymbol{x}_k;\xi_k)$ is the stochastic gradient, typically assumed to satisfy properties such as being unbiased and having bounded variance; *i.e.*,

$$\mathbb{E}_\xi\big[g(\boldsymbol{x};\xi) \;\big|\; \boldsymbol{x}\big] = \nabla f(\boldsymbol{x}), \text{ and} \tag{2}$$

$$\mathbb{E}_\xi\big[\|g(\boldsymbol{x};\xi) - \nabla f(\boldsymbol{x})\|^2 \;\big|\; \boldsymbol{x}\big] \le \sigma^2 \tag{3}$$

where σ^2 is a finite, non-negative scalar. In practice, stochastic gradients arise when one estimates the gradient of f with a mini-batch gradient: the average of the gradients of a small, randomly sampled subset of terms f_i.

Training large DNNs on large datasets is time-consuming, and the search for more efficient optimization methods for this task remains very active [4]. The recently-introduced Lookahead optimizer [5] takes a novel approach to improving stability and reducing variance for DNN training. Zhang et al. [5] propose the Lookahead optimizer, which we describe in detail below, and empirically demonstrate that it improves the performance of other methods like SGD and ADAM [6] on image classification, language modeling, and neural machine translation tasks. Zhang et al. [5] also provide some theoretical guarantees in the case where f is quadratic. Specifically, they derive an optimal step size and show that the variance of Lookahead is lower than that of SGD when using stochastic gradients, and they also provide some convergence guarantees in the setting where f is quadratic and gradients are deterministic.

Our contribution in this paper is to provide a convergence analysis of Lookahead in a more general setting. Specifically, we assume that f is smooth (*i.e.*, differentiable, with Lipschitz gradients) and that the stochastic gradients satisfy the properties (2) and (3). In this setting, our main results show that Lookahead converges to a first-order stationary point; specifically, if one uses a sequence of step sizes that decrease at an appropriate rate, then $\mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|^2] \to 0$, and if one uses an appropriately-chosen constant step size, then

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(\boldsymbol{x}_k)\|^2] = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

Perhaps surprisingly, although it is a serial optimization method, our proof approach involves viewing Lookahead as a multi-agent optimization method [7] with two agents. This allows us to leverage existing analysis techniques for multi-agent stochastic optimization methods, and it also opens the door to natural parallel, multi-agent extensions of Lookahead.

2. THE LOOKAHEAD OPTIMIZER

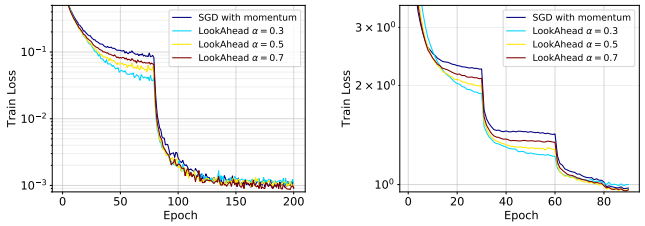
The pseudo-code of Lookahead is listed in Algorithm 1. Lookahead maintains two copies of the model parameters, the *fast model* and a *slow model*. The fast model is updated at each iteration using the selected base optimizer (e.g., SGD, ADAM). The slow model is only updated every τ steps and considers the change of the fast model, i.e. $\boldsymbol{x}_{t,\tau} - \boldsymbol{x}_{t,0}$, as the descent direction and uses $\alpha < 1$ as the step size as shown in line 6 of Algorithm 1. In the next round, the initial fast model will be reset to the current value of slow model (see line 2).

We compare LookAhead with SGD Momentum on the CIFAR-10 and ImageNet datasets. We train a Resnet-34 following the experimental protocol of [2] on CIFAR10 and an Resnet-50 on ImageNet

Algorithm 1: Lookahead Optimizer of Zhang et al. [5]

Input: Initial model parameter $\boldsymbol{z}_0$; Objective function $f(\boldsymbol{x})$; Base optimizer A; Slow learning rate α; Inner loop length τ.

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  Synchronize parameters $\boldsymbol{x}_{t,0} \leftarrow \boldsymbol{z}_{t-1}$
  for $l \in \{0, 1, \ldots, \tau-1\}$ **do**
    Base optimizer step: $\boldsymbol{x}_{t,l+1} = \boldsymbol{x}_{t,l} - A(f, \boldsymbol{x}_{t,l})$
  end
  Update slow model: $\boldsymbol{z}_t = \boldsymbol{z}_{t-1} + \alpha(\boldsymbol{x}_{t,\tau} - \boldsymbol{x}_{t,0})$
end
Return slow model $\boldsymbol{z}_{T-1}$



(a) Resnet34 on CIFAR-10 (b) Resnet50 on ImageNet

Fig. 1: Comparison between SGD with Momentum and LookAhead optimizers on CIFAR10 and ImageNet. We observe that LookAhead provides optimization benefit over SGD with momentum on both datasets. Using low values for the slow learning rate α can speed-up the optimization initially, but do not achieve as good final training loss as higher α values.

using a similar set-up than [8]. We use a batch-size of 256 and step-wise learning rate decay. In both case, we observe that LookAhead provides optimization benefit over SGD with momentum. Using low value for the slow learning rate α can further speed-up the optimization initially, but do not achieve as good final training loss as higher α values. LookAhead did not lead to significantly better generalization performances than SGD, in both settings.

3. CONVERGENCE ANALYSIS

In this section we provide a new perspective to analyze the Lookahead optimizer by viewing the optimizer from a multi-agent perspective [7]. Specifically, one agent updates the fast model $\boldsymbol{x}$ and the other agent updates the slow model $\boldsymbol{z}$. While the fast model is updated after every iteration using the base optimizer (*e.g.*, SGD), the slow model is only changed through the periodic synchronization between agents. In a typical two-agent problem, the agents would cooperate to minimize $f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2)$ subject to a consensus constraint $\boldsymbol{x}_1 = \boldsymbol{x}_2$. Here we have $f_1 = f$, $f_2 = 0$, $\boldsymbol{x}_1 = \boldsymbol{x}$, and $\boldsymbol{x}_2 = \boldsymbol{z}$.

The description of Lookahead originally presented in [5], and as shown in Algorithm 1, involves nested loops, with the fast variables updated in the inner loop, and the slow variables updated once in each outer loop. Of course, the algorithm can be written equivalently with a single loop (let us refer to an iteration counter k) where the fast variables are updated at every iteration and the slow variables are only updated at those iterations where, e.g., $(k+1) \bmod \tau = 0$. We will analyze this single-loop setting.

Formally, let $\boldsymbol{x}_k$ denote the fast model after k base optimizer steps, and $\boldsymbol{z}_k$ denote the corresponding slow model. Define the parameter matrix $\boldsymbol{X}_k = [\boldsymbol{x}_k, \boldsymbol{z}_k] \in \mathbb{R}^{d\times 2}$ and stochastic gradients matrix $\boldsymbol{G}_k = [g(\boldsymbol{x}_k; \xi_k), \boldsymbol{0}] \in \mathbb{R}^{d\times 2}$. Then, the update rule of the Lookahead optimizer can be written as

$$\boldsymbol{X}_{k+1} = [\boldsymbol{X}_k - \gamma_k \boldsymbol{G}_k]\, \boldsymbol{P}_k \tag{4}$$

where $\boldsymbol{P}_k \in \mathbb{R}^{2\times 2}$ represents the model synchronization matrix,

$$\boldsymbol{P}_k = \begin{cases} \boldsymbol{a}\boldsymbol{1}^\top, & (k+1) \bmod \tau = 0 \\ \boldsymbol{I}, & \text{otherwise} \end{cases} \tag{5}$$

with $\boldsymbol{a} = [\alpha, 1-\alpha]^\top$. It is worth noting that $(\boldsymbol{P}_k)_{k\geq 0}$ is a sequence of time-varying column stochastic matrices, unlike many previous works on decentralized optimization [7, 9, 10] which require $\boldsymbol{P}_k$ to be row-stochastic or doubly-stochastic. Moreover, not every $\boldsymbol{P}_k$ has its second largest eigenvalue strictly less than one in magnitude. Previous studies using column-stochastic matrices include [11, 12, 13] and those incorporating push-sum updates [14, 15].

Note that $\boldsymbol{1}^\top \boldsymbol{a} = \boldsymbol{a}^\top \boldsymbol{1} = 1$ and $\boldsymbol{I}\boldsymbol{a} = \boldsymbol{a}$. Therefore, $\boldsymbol{P}_k \boldsymbol{a} = \boldsymbol{a}$. Multiplying the vector $\boldsymbol{a}$ on both sides of (4), we have

$$\boldsymbol{X}_{k+1}\boldsymbol{a} = [\boldsymbol{X}_k - \gamma_k \boldsymbol{G}_k]\, \boldsymbol{P}_k \boldsymbol{a} \tag{6}$$

$$= \boldsymbol{X}_k \boldsymbol{a} - \alpha\gamma_k g(\boldsymbol{x}_k; \xi_k) \tag{7}$$

For the purpose of analysis, let us define the sequence $\boldsymbol{y}_k = \boldsymbol{X}_k \boldsymbol{a} = \alpha\boldsymbol{x}_k + (1-\alpha)\boldsymbol{z}_k, \forall k \geq 0$. Substituting the definition of $\boldsymbol{y}_k$ into (7), we obtain the simple vector-form update rule:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \alpha\gamma_k g(\boldsymbol{x}_k; \xi_k). \tag{8}$$

Comparing (8) to the normal SGD updates ($\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \gamma_k g(\boldsymbol{y}_k; \xi_k)$), we can observe that $\boldsymbol{y}_k$ performs perturbed gradient updates using an effective learning rate $\alpha\gamma_k$, and the stochastic gradients are evaluated at $\boldsymbol{x}_k$ instead of $\boldsymbol{y}_k$. Besides, note that, $\forall t \geq 0$,

$$\boldsymbol{y}_{t\tau} - \boldsymbol{x}_{t\tau} = (1-\alpha)(\boldsymbol{z}_{t\tau} - \boldsymbol{x}_{t\tau}) = 0, \tag{9}$$

$$\boldsymbol{y}_{t\tau} - \boldsymbol{z}_{t\tau} = \alpha(\boldsymbol{x}_{t\tau} - \boldsymbol{z}_{t\tau}) = 0. \tag{10}$$

The fast model, slow model and $\boldsymbol{y}_k$ are equal to each other after every τ steps. Thus, if $\boldsymbol{y}_k$ converges to a stationary point then the fast model $\boldsymbol{x}_k$ and slow model $\boldsymbol{z}_k$ do too. In the sequel, we will focus on analyzing the convergence of $\boldsymbol{y}_k$.

3.1. Main Results

The convergence analysis is based on the assumption that stochastic gradients are unbiased and have bounded variance, as presented in (2) and (3). Moreover, assume the objective function $f(\boldsymbol{x})$ is L-smooth, *i.e.*,

$$\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{11}$$

We first show that Lookahead with diminishing learning rate can converge to a stationary point in the considered setting. The proofs of all theorems are deferred to Section 3.2.

Theorem 1 (Convergence of Lookahead, diminishing learning rate)**.** *Suppose the Lookahead optimizer is initialized at $\boldsymbol{y}_0 = \boldsymbol{x}_0 = \boldsymbol{z}_0$. If the fast learning rate is kept as a constant within each inner loop, i.e., $\gamma_{t\tau+l} = \gamma_{t\tau}, \forall t \geq 0, l \in \{0, 1, \ldots, \tau-1\}$, and satisfies*

$$\sum_{t=0}^{\infty} \gamma_{t\tau} = \infty, \sum_{t=0}^{\infty} \gamma_{t\tau}^2 < \infty, \sum_{t=0}^{\infty} \gamma_{t\tau}^3 < \infty, \tag{12}$$

$$\alpha\gamma_{t\tau}L + (1-\alpha)^2\gamma_{t\tau}^2 L^2 \tau(\tau-1) \leq 1, \forall t \geq 0 \tag{13}$$

then under Assumptions (2)*,* (3) *and* (11)*, we have that*

$$\liminf_{k\to\infty} \mathbb{E}[\|\nabla f(\boldsymbol{y}_k)\|^2] = 0.$$

In order to get a sense of the convergence rate and how the optimization error bound is influenced by different hyper-parameters, we have the following theorem.

Theorem 2 (Convergence of Lookahead, fixed learning rate)**.** *Suppose the Lookahead optimizer is initialized at $\boldsymbol{y}_0 = \boldsymbol{x}_0 = \boldsymbol{z}_0$. If the fast learning rate is kept as a constant within each inner loop, i.e., $\gamma_{t\tau+l} = \gamma, \forall t \geq 0, l \in \{0, 1, \ldots, \tau-1\}$, and satisfies* (13)*, then under Assumptions* (2)*,* (3) *and* (11)*, we have that:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(\boldsymbol{y}_k)\|^2] \leq \frac{2[f(\boldsymbol{y}_0) - f_{inf}]}{\alpha\gamma K} + \alpha\gamma L\sigma^2 + (1-\alpha)^2\gamma^2 L^2\sigma^2(\tau-1) \tag{14}$$

where f_{inf} denotes the lower bound of the objective function. Furthermore, if we set $\gamma = 1/\sqrt{K}$, then the above bound becomes:

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(\boldsymbol{y}_k)\|^2] \leq \frac{2[f(\boldsymbol{y}_0) - f_{inf}] + \alpha^2 L\sigma^2}{\alpha\sqrt{K}} + \frac{(1-\alpha)^2 L^2\sigma^2(\tau-1)}{K} \tag{15}$$

$$= \mathcal{O}\left(\frac{1}{\sqrt{K}}\right). \tag{16}$$

Theorem 2 shows that when the learning rate is configured properly, Lookahead can achieve the same asymptotic convergence rate $1/\sqrt{K}$ as mini-batch SGD [4]. Furthermore, note that when γ is fixed and K approaches to the infinity, the optimization error bound (14) becomes $\alpha\gamma L\sigma^2 + (1-\alpha)^2\gamma^2 L^2\sigma^2(\tau-1)$. There should be a best value of α that minimizes the error bound. Besides, comparing to the asymptotic error floor of mini-batch SGD ($\gamma L\sigma^2$), one can observe that only when $(1-\alpha)\gamma L(\tau-1) \leq 1$, Lookahead can achieve a better bound. This puts a constraint on the minimal value of α and is corroborated well by experiments in Figure 1: a small α cannot achieve as good final training loss as higher α values.

3.2. Proofs of Main Theorems

It follows from (11) and the update rule (8) that

$$f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{y}_k) \leq -\alpha\gamma_k \langle \nabla f(\boldsymbol{y}_k),\, g(\boldsymbol{x}_k;\xi_k)\rangle + \frac{\alpha^2\gamma_k^2 L}{2}\|g(\boldsymbol{x}_k;\xi_k)\|^2. \tag{17}$$

For the ease of writing, we denote by $\mathbb{E}_k[\cdot]$ the conditional expectation $\mathbb{E}_{\xi_k\sim\mathcal{D}}[\cdot|\mathcal{F}_k]$, where $\mathcal{F}_k$ is the sigma algebra generated by the noise in stochastic gradients until iteration k. Taking the conditional expectation on both sides of (17), for the first term on RHS, we have

$$\mathbb{E}_k[\langle \nabla f(\boldsymbol{y}_k),\, g(\boldsymbol{x}_k;\xi_k)\rangle] = \langle \nabla f(\boldsymbol{y}_k),\, \nabla f(\boldsymbol{x}_k)\rangle \tag{18}$$

$$= \frac{1}{2}\left[\|\nabla f(\boldsymbol{y}_k)\|^2 + \|\nabla f(\boldsymbol{x}_k)\|^2 - \|\nabla f(\boldsymbol{y}_k) - \nabla f(\boldsymbol{x}_k)\|^2\right] \tag{19}$$

$$\geq \frac{1}{2}\left[\|\nabla f(\boldsymbol{y}_k)\|^2 + \|\nabla f(\boldsymbol{x}_k)\|^2 - L^2\|\boldsymbol{y}_k - \boldsymbol{x}_k\|^2\right] \tag{20}$$

$$= \frac{1}{2}\left[\|\nabla f(\boldsymbol{y}_k)\|^2 + \|\nabla f(\boldsymbol{x}_k)\|^2 - (1-\alpha)^2 L^2\|\boldsymbol{x}_k - \boldsymbol{z}_k\|^2\right] \tag{21}$$

where (20) follows (11) and (21) comes from the definition of $\boldsymbol{y}_k$. For the second term on RHS of (17), applying Assumptions (2) and (3),

$$\mathbb{E}_k\left[\|g(\boldsymbol{x}_k;\xi_k)\|^2\right] \leq \|\nabla f(\boldsymbol{x}_k)\|^2 + \sigma^2. \tag{22}$$

Plugging (21) and (22) back into (17) and taking the total expectation, we get

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{y}_{k+1})] \leq& \mathbb{E}[f(\boldsymbol{y}_k)] - \frac{\alpha\gamma_k}{2}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_k)\|^2\right] + \frac{\alpha^2\gamma_k^2 L\sigma^2}{2} \\ &- \frac{\alpha\gamma_k}{2}(1-\alpha\gamma_k L)\,\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \\ &+ \frac{\alpha\gamma_k(1-\alpha)^2 L^2}{2}\mathbb{E}\left[\|\boldsymbol{x}_k - \boldsymbol{z}_k\|^2\right].\end{aligned} \tag{23}$$

Without loss of generality, assume that $k = t\tau + l$, where $t \geq 0$ denotes the index of outer iteration and $l \in \{0, 1, \ldots, \tau-1\}$ denotes the index of inner loop steps. Note that the fast learning rate is kept the same within each inner loop, that is, $\gamma_{t\tau+l} = \gamma_{t\tau}, \forall t, l$. Then, summing over the t-th outer iteration,

$$\begin{aligned}&\mathbb{E}[f(\boldsymbol{y}_{(t+1)\tau})] - \mathbb{E}[f(\boldsymbol{y}_{t\tau})] \\ \leq& -\frac{\alpha\gamma_{t\tau}}{2}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right] + \frac{\alpha^2\gamma_{t\tau}^2 L\sigma^2\tau}{2} \\ &- \frac{\alpha\gamma_{t\tau}}{2}(1-\alpha\gamma_{t\tau}L)\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2\right] \\ &+ \frac{\alpha\gamma_{t\tau}(1-\alpha)^2 L^2}{2}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\boldsymbol{x}_{t\tau+l} - \boldsymbol{z}_{t\tau+l}\|^2\right].\end{aligned} \tag{24}$$

Now, we are going to bound the last term in (24). Recall that $\boldsymbol{x}_{t\tau} = \boldsymbol{z}_{t\tau} = \boldsymbol{z}_{t\tau+l}, \forall t \geq 0, l < \tau$. It follows that

$$\mathbb{E}\left[\|\boldsymbol{x}_{t\tau+l} - \boldsymbol{z}_{t\tau+l}\|^2\right] = \mathbb{E}\left[\|\boldsymbol{x}_{t\tau+l} - \boldsymbol{x}_{t\tau}\|^2\right] \tag{25}$$

$$= \mathbb{E}\left[\left\|\sum_{j=t\tau}^{t\tau+l-1}\gamma_{t\tau} g(\boldsymbol{x}_j;\xi_j)\right\|^2\right] \tag{26}$$

$$= \gamma_{t\tau}^2\mathbb{E}\left[\left\|\sum_{j=t\tau}^{t\tau+l-1} g(\boldsymbol{x}_j;\xi_j)\right\|^2\right]. \tag{27}$$

Repeatedly using the fact $\left\|\sum_{i=1}^n a_i\right\|^2 \leq n\sum_{i=1}^n \|a_i\|^2$, we get

$$\begin{aligned}&\mathbb{E}\left[\left\|\sum_{j=t\tau}^{t\tau+l-1} g(\boldsymbol{x}_j;\xi_j)\right\|^2\right] \\ \leq& 2\mathbb{E}\left[\left\|\sum_{j=t\tau}^{t\tau+l-1}(g(\boldsymbol{x}_j;\xi_j) - \nabla f(\boldsymbol{x}_j))\right\|^2 + \left\|\sum_{j=t\tau}^{t\tau+l-1}\nabla f(\boldsymbol{x}_j)\right\|^2\right]\end{aligned} \tag{28}$$

$$\leq 2\sigma^2 l + 2\mathbb{E}\left[\left\|\sum_{j=t\tau}^{t\tau+l-1}\nabla f(\boldsymbol{x}_j)\right\|^2\right] \tag{29}$$

$$\leq 2\sigma^2 l + 2l\sum_{j=t\tau}^{t\tau+l-1}\mathbb{E}[\|\nabla f(\boldsymbol{x}_j)\|^2] \tag{30}$$

where (29) uses the fact that $\{\xi_j\}$ is an i.i.d. sequence of random variables, and hence, for all j, s,

$$\mathbb{E}\left[\langle g(\boldsymbol{x}_j;\xi_j) - \nabla f(\boldsymbol{x}_j),\, g(\boldsymbol{x}_s;\xi_s) - \nabla f(\boldsymbol{x}_s)\rangle\right] = 0. \tag{31}$$

Combining (27) and (30) and summing from $l=0$ to $l=\tau-1$,

$$\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\boldsymbol{x}_{t\tau+l}-\boldsymbol{z}_{t\tau+l}\|^2\right] \tag{32}$$

$$\leq\sigma^2\gamma_{t\tau}^2(\tau-1)\tau+2\gamma_{t\tau}^2\sum_{l=0}^{\tau-1}l\sum_{j=t\tau}^{t\tau+l-1}\mathbb{E}[\|\nabla f(\boldsymbol{x}_j)\|^2] \tag{33}$$

$$=\sigma^2\gamma_{t\tau}^2(\tau-1)\tau+2\gamma_{t\tau}^2\sum_{l=0}^{\tau-2}\left[\mathbb{E}[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2]\sum_{s=l+1}^{\tau-1}s\right] \tag{34}$$

$$=\sigma^2\gamma_{t\tau}^2(\tau-1)\tau+\gamma_{t\tau}^2\sum_{l=0}^{\tau-2}\left[\mathbb{E}[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2](\tau+l)(\tau-1-l)\right]. \tag{35}$$

Note that $(\tau+l)(\tau-1-l)$ achieves its maximal value when $l=0$. Therefore, we have

$$\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\boldsymbol{x}_{t\tau+l}-\boldsymbol{z}_{t\tau+l}\|^2\right] \tag{36}$$

$$\leq\sigma^2\gamma_{t\tau}^2(\tau-1)\tau+\gamma_{t\tau}^2(\tau-1)\tau\sum_{l=0}^{\tau-2}\mathbb{E}[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2] \tag{37}$$

$$\leq\gamma_{t\tau}^2(\tau-1)\tau\left[\sigma^2+\sum_{l=0}^{\tau-1}\mathbb{E}[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2]\right]. \tag{38}$$

Here, we finish bounding the last term in (24). Substituting (38) into (24), one can obtain

$$\begin{aligned}&\mathbb{E}[f(\boldsymbol{y}_{(t+1)\tau})]-\mathbb{E}[f(\boldsymbol{y}_{t\tau})]\\ \leq&-\frac{\alpha\gamma_{t\tau}}{2}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right]+\frac{\alpha^2\gamma_{t\tau}^2L\sigma^2\tau}{2}\\ &+\frac{(1-\alpha)^2\alpha\gamma_{t\tau}^3L^2\sigma^2\tau(\tau-1)}{2}-\frac{\alpha\gamma_{t\tau}C}{2}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_{t\tau+l})\|^2\right]\end{aligned} \tag{39}$$

where $C=1-\alpha\gamma_{t\tau}L-(1-\alpha)^2\gamma_{t\tau}^2L^2\tau(\tau-1)$. According to the constraint given in (13), $C\geq 0$ and thus,

$$\begin{aligned}&\mathbb{E}[f(\boldsymbol{y}_{(t+1)\tau})]-\mathbb{E}[f(\boldsymbol{y}_{t\tau})]\\ \leq&-\frac{\alpha\gamma_{t\tau}}{2}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right]+\frac{\alpha^2\gamma_{t\tau}^2L\sigma^2\tau}{2}\\ &+\frac{(1-\alpha)^2\alpha\gamma_{t\tau}^3L^2\sigma^2\tau(\tau-1)}{2}\end{aligned} \tag{40}$$

Summing over both sides of (40) from $t=0$ to $t=T-1$,

$$\begin{aligned}&\mathbb{E}[f(\boldsymbol{y}_{T\tau})-f(\boldsymbol{y}_0)]\\ \leq&-\frac{1}{2}\sum_{t=0}^{T-1}\alpha\gamma_{t\tau}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right]+\frac{\alpha^2L\sigma^2\tau}{2}\sum_{t=0}^{T-1}\gamma_{t\tau}^2\\ &+\frac{(1-\alpha)^2\alpha L^2\sigma^2(\tau-1)\tau}{2}\sum_{t=0}^{T-1}\gamma_{t\tau}^3.\end{aligned} \tag{41}$$

After minor rearranging, we have

$$\begin{aligned}&\frac{1}{\tau S_T}\sum_{t=0}^{T-1}\gamma_{t\tau}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right]\\ \leq&\frac{2\mathbb{E}[f(\boldsymbol{y}_0)-f(\boldsymbol{y}_{T\tau})]}{\alpha\tau S_T}+\alpha L\sigma^2\frac{\sum_{t=0}^{T-1}\gamma_{t\tau}^2}{S_T}\\ &+(1-\alpha)^2L^2\sigma^2(\tau-1)\frac{\sum_{t=0}^{T-1}\gamma_{t\tau}^3}{S_T}\end{aligned} \tag{42}$$

$$\begin{aligned}\leq&\frac{2[f(\boldsymbol{y}_0)-f_{\text{inf}}]}{\alpha\tau S_T}+\alpha L\sigma^2\frac{\sum_{t=0}^{T-1}\gamma_{t\tau}^2}{S_T}\\ &+(1-\alpha)^2L^2\sigma^2(\tau-1)\frac{\sum_{t=0}^{T-1}\gamma_{t\tau}^3}{S_T}\end{aligned} \tag{43}$$

where $S_T=\sum_{t=0}^{T-1}\gamma_{t\tau}$. Recall the learning rate constraint (12), as the increase of T, the RHS of (43) approaches to 0. Here, we complete the proof of Theorem 1.

When the (fast) learning rate is a fixed constant, *i.e.*, $\gamma_{t\tau}=\gamma,\forall t\geq 0$, the upper bound (43) changes to

$$\begin{aligned}&\frac{1}{\tau T}\sum_{t=0}^{T-1}\sum_{l=0}^{\tau-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_{t\tau+l})\|^2\right]=\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_k)\|^2\right]\\ \leq&\frac{2[f(\boldsymbol{y}_0)-f_{\text{inf}}]}{\alpha\gamma K}+\alpha\gamma L\sigma^2+(1-\alpha)^2\gamma^2L^2\sigma^2(\tau-1)\end{aligned} \tag{44}$$

where $K=T\tau$ is the total number of iterations. When we set $\gamma=1/\sqrt{K}$, it follows that

$$\begin{aligned}&\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla f(\boldsymbol{y}_k)\|^2\right]\\ \leq&\frac{2[f(\boldsymbol{y}_0)-f_{\text{inf}}]+\alpha^2L\sigma^2}{\alpha\sqrt{K}}+\frac{(1-\alpha)^2L^2\sigma^2(\tau-1)}{K},\end{aligned} \tag{45}$$

which completes the proof of Theorem 2.

4. CONCLUSIONS

We have shown that Lookahead converges to a first-order stationary point of smooth non-convex settings when the stochastic gradients are unbiased and have finite variance and the base optimizer is SGD. Our proof approach involves viewing Lookahead, which is readily described as a two-timescale algorithm, as a multi-agent optimization method with two agents (one fast, one slow) which periodically synchronize.

While a direct approach to scaling up Lookahead would be to use a method like large mini-batch SGD as the fast optimizer, the multi-agent perspective also naturally leads to a decentralized implementation where nodes use approximate distributed averaging. Another natural extension of Lookahead is to incorporate some form of momentum into the slow update step. We found that using momentum in the slow update did not provide any advantage for serial Lookahead. However, using momentum in the slow updates provides a substantial improvement in the multi-agent setting. We investigate this further in our recent submission [16].

5. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech

recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 11, pp. 82–97, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, Jun. 2016, pp. 770–778.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Long Beach, USA, Dec. 2017, pp. 5998–6008.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[5] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead optimizer: k steps forward, 1 step back," in *Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, San Diego, USA, May 2015.

[7] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[8] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: Training ImageNet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[9] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *International Conference on Machine Learning*, Long Beach, USA, Jun. 2019.

[10] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[11] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks – Part I: Transient analysis," *IEEE Trans. Information Theory*, vol. 61, no. 6, pp. 3487–3517, Jun. 2015.

[12] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4–5, 2014.

[13] S. Vlaski and A. H. Sayed, "Distributed learning in non-convex environments – Part I: Agreement at a linear rate," *arXiv preprint 1907.01848*, Jul. 2019.

[14] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *IEEE Conf. Decision and Control (CDC)*, 2012, pp. 5453–5458.

[15] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[16] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, "SlowMo: Improving communication-efficient distributed SGD with slow momentum," in *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, Apr. 2020.