# DIVERSELY STALE PARAMETERS FOR EFFICIENT TRAINING OF DEEP CONVOLUTIONAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The backpropagation algorithm is the most popular algorithm training neural networks nowadays. However, it suffers from the forward locking, backward locking and update locking problems, especially when a neural network is so large that its layers are distributed across multiple devices. Existing solutions either can only handle one locking problem or lead to severe accuracy loss or memory inefficiency. Moreover, none of them consider the straggler problem among devices. In this paper, we propose **Layer-wise Staleness** and a novel efficient training algorithm, **Diversely Stale Parameters** (DSP), which can address all these challenges without loss of accuracy nor memory issue. We also analyze the convergence of DSP with two popular gradient-based methods and prove that both of them are guaranteed to converge to critical points for non-convex problems. Finally, extensive experimental results on training deep convolutional neural networks demonstrate that our proposed DSP algorithm can achieve significant training speedup with stronger robustness and better generalization than compared methods.

## 1 INTRODUCTION

The increasing depth and size of deep neural network (DNN) are shown to be one of the most important factors leading to its success (Szegedy et al., 2015; Simonyan & Zisserman, 2014). However, as the neural networks get deeper and larger (He et al., 2015; Ioffe & Szegedy, 2015; Hu et al., 2018; Szegedy et al., 2016; Xie et al., 2017), the required expensive training time and hardware resources have become the bottleneck. Data parallelism (Valiant, 1990; Li et al., 2014; Bottou, 2010) and model parallelism (Lee et al., 2014; Krizhevsky, 2014) are two standard parallelism techniques to utilize multiple devices to address this issue.

The data parallelism has been well studied and implemented in existing libraries (Szegedy et al., 2016; Abadi et al., 2016; Chen et al., 2015), but the model parallelism is still underexplored. In this paper, we focus on the model parallelism, where the DNN benefits from being split onto multiple devices. But the resource utilization of standard model parallelism can be very low. The backpropagation algorithm (Rumelhart et al., 1988; LeCun et al., 1989) typically requires two phases to update the model in each training step: the forward pass and backward pass. But the sequential propagation of activation and error gradient leads to *backward locking* and *forward locking* (Jaderberg et al., 2017) respectively, because a layer's computations have dependencies. The *update locking* (Jaderberg et al., 2017) exists as the backward pass will not start until the forward pass has completed. This sequential execution keeps a device inefficiently waiting for the activation input and error gradient.

Several works have been proposed to address these locking issues (Figure 1). (Jaderberg et al., 2017) uses Decoupled Neural Interfaces (DNI) to predict the error gradient via auxiliary networks, so that a layer uses the synthetic gradient and needs not to wait for the error gradient. (Nøkland, 2016) lets hidden layers receive error information directly from the output layer. However, these methods can not converge when dealing with very deep neural networks. (Belilovsky et al., 2019) proposes layer-wise decoupled greedy learning (DGL), which introduces an auxiliary classifier for each block of layers so that a block updates its parameters according to its own classifier. But the objective function of DGL based on greedy local predictions can be very different from the original model. GPipe (Huang et al., 2018) proposes pipeline parallelism and divides each mini-batch into micro-batches, which can be regarded as a combination of model parallelism and data parallelism. However, the forward and backward lockings of the micro-batch still exist, and the update locking is

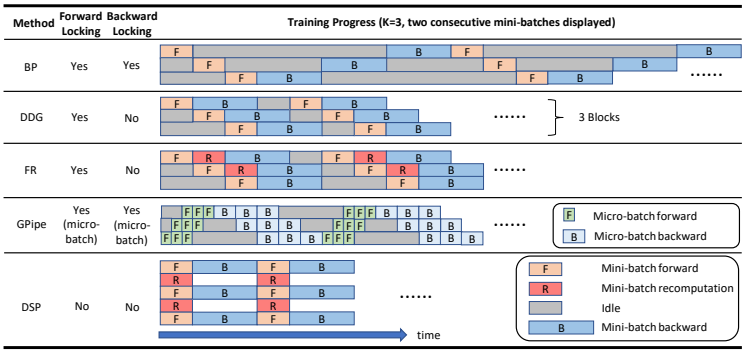


Figure 1: Comparison of different methods with three blocks. The forward and recomputation are overlapped in DSP.

not addressed because GPipe waits for the whole forward and backward pass to finish before updating the parameters. (Huo et al., 2018b) proposes Decoupled Parallel Backpropagation (DDG), which divides the DNN into blocks and removes the backward locking by storing delayed error gradient and intermediate activations at each block. But DDG suffers from large memory consumption due to storing all the intermediate results, and cannot converge when the DNN goes further deeper. Features Replay (FR) (Huo et al., 2018a) improves DDG via storing the history inputs and recomputing the intermediate results. Nevertheless, blocks in DDG and FR still need to wait for the backward error gradient. Besides, neither DDG nor FR addresses the forward locking problem.

To overcome the aforementioned drawbacks, we first propose *Layer-wise Staleness*, a fine-grained staleness within the model to allow different parts to be trained independently. Incorporating staleness is useful for efficient asynchronous execution without synchronization barrier (Ho et al., 2013), which can be interpreted as another form of locking/dependency. The introduction of preset Layer-wise Staleness enables each part of the convolutional neural network (CNN) to run in a very flexible way with certain degree of asynchrony. Based on the concept of Layer-wise Staleness, we propose a novel parallel CNN training algorithm named Diversely Stale Parameters (DSP), where lower layers use more stale information to update parameters. DSP also utilizes recomputation technique (Chen et al., 2016; Griewank, 1999) to reduce memory consumption, which is overlapped with the forward pass. Our contributions are summarized as follows:

- We propose Layer-wise Staleness and Diversely Stale Parameters (Section 3) which breaks the forward, backward and update lockings without memory issues.
- Then, we provide convergence analysis (Section 4) for the proposed method. Even faced with parameters and data of different Layer-wise Staleness, we prove that DSP converges to critical points for non-convex problems with SGD and momentum SGD.
- We evaluate our method via training deep convolutional neural networks (Section 5). Extensive experimental results show that DSP achieves significant training speedup, strong robustness against random stragglers, and generalizes better.

## 2 BACKGROUND

We divide a CNN into $K$ consecutive blocks so that the whole parameters $x = (x_0, x_1, ..., x_{K-1}) \in \mathbb{R}^d$, where $x_k \in \mathbb{R}^{d_k}$ denotes the partial parameters at block $k \in \{0, 1, ..., K-1\}$ and $d = \sum_{k=0}^{K-1} d_k$. Each block $k$ computes activation $h_{k+1} = f_k(h_k; x_k)$, where $h_k$ denotes the input of block $k$. In particular, $h_0$ is the input data. For simplicity, we define $F(h_0; x_0; x_1; ...; x_k) \coloneqq f_k(...f_1(f_0(h_0; x_0); x_1)...; x_k) = h_{k+1}$. The loss is $\mathcal{L}(h_K, l)$, where $l$ is the label. Minimizing the loss of a $K$-block neural network can be represented by the following problem:

$$\min_{x \in \mathbb{R}^d} \quad f(x) \coloneqq \mathcal{L}(F(h_0; x_0; x_1; ...; x_{K-1}), l). \tag{1}$$

Backpropagation algorithm computes the gradient for block $k$ following chain rule via Eq. (2). The forward locking exists because the input of each block is dependent on the output from the lower block. The backward locking exists because each block cannot compute gradients until having received the error gradient $\mathcal{G}_h$ from the upper block. Besides, the backward process can not start until the whole forward process is completed, which is known as the update locking.

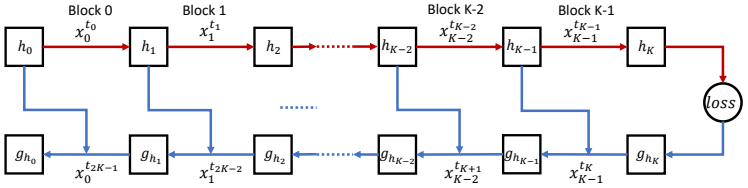


Figure 2: A DSP data traversal in a K-block neural network. Red arrows denote the forward pass; blue arrows denote the backward pass. The arrows use parameters at different timestamps.

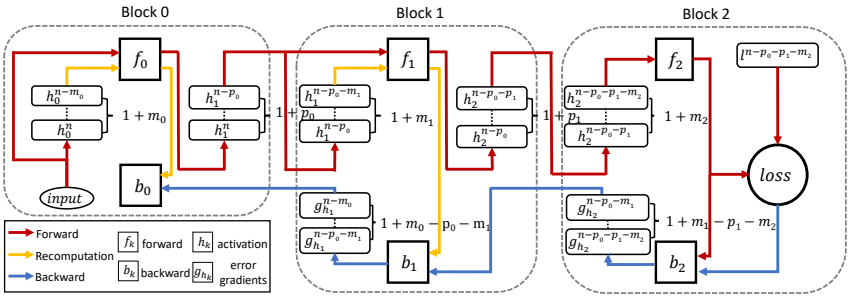


Figure 3: $DSP(p_0, p_1, 0; m_0, m_1, m_2)$ for parallel training (K=3). The input queue is at the L.H.S of a block, while the output queue is at the R.H.S. The gradient queue is below the input queue.

$$\mathcal{G}_{h_k} = \frac{\partial f_k(h_k; x_k)}{\partial h_k}\mathcal{G}_{h_{k+1}}, \quad \mathcal{G}_{h_K} = \frac{\partial \mathcal{L}(h_K, l)}{\partial h_K} \quad \text{and} \quad \mathcal{G}_{x_k} = \frac{\partial f_k(h_k; x_k)}{\partial x_k}\mathcal{G}_{h_{k+1}}. \tag{2}$$

After computing the gradients, stochastic gradient descent (SGD) (Robbins & Monro, 1951) and its variants such as stochastic unified momentum (SUM) (Yang et al., 2016), RMSPROP (Tieleman & Hinton, 2012) and ADAM (Kingma & Ba, 2014) are widely used for updating the model. SGD updates via $x^{n+1} = x^n - \alpha\mathcal{G}(x^n; \xi)$, where $x^n$ is the parameters when training with the $n^{\text{th}}$ feeding data (batch), $\alpha$ is the learning rate, and $\mathcal{G}(x^n; \xi)$ is the stochastic gradient. SUM updates the parameters via Eq. (3), where $\beta$ is the momentum constant and $y$ is the momentum term. When $s = 1$, SUM reduces to stochastic Nesterov's accelerated gradient (SNAG) (Nesterov, 2013).

$$y^{n+1} = x^n - \alpha\mathcal{G}(x^n; \xi),\ y^{s,n+1} = x^n - s\alpha\mathcal{G}(x^n; \xi) \ \text{ and } \ x^{n+1} = y^{n+1} + \beta(y^{s,n+1} - y^{s,n}). \tag{3}$$

## 3 Diversely Stale Parameters

In this section, we propose a novel training method named Diversely Stale Parameters. We will describe how to apply DSP to training neural networks in parallel via layer-wise staleness and how to compute the DSP gradient.

### 3.1 Layer-wise Staleness

We preset each block's Layer-wise Staleness to a different value to break the synchronization barrier of backpropagation. In order to represent the Layer-wise Staleness explicitly, we mark the parameters with a timestamp during the two-phase forward and backward training procedure.

As shown in Figure 2, the data is fowarded with parameters $x_0$ at timestamp $t_0$, $x_1$ at timestamp $t_1$, ..., and $x_{K-1}$ at timestamp $t_{K-1}$. For simplicity we denote the **Forward Parameters** as $\{x_k^{t_k}\}_{k=0,\dots,K-1}$. Similarly we denote the **Backward Parameters** as $\{x_k^{t_{2K-1-k}}\}_{k=0,\dots,K-1}$. Then we define **Layer-wise Staleness** as,

$$\Delta t_k = t_{2K-k-1} - t_k \geq 0\,.$$

We also denote the maximum Layer-wise Staleness as $\Delta t = \max_{k=0,1,\dots,K-1} \Delta t_k$. It is worth noting that,

- Layer-wise Staleness $\Delta t_k$ is a constant and set in advance.
- In standard backpropagation algorithm (Eq. (2)), Layer-wise Staleness $\Delta t_k = 0$.
- Feeding data index is not identical to timestamp / training step.

### 3.2 DSP GRADIENT

Here we introduce the DSP gradient based on the concept of Layer-wise Staleness. We first go back to Figure 2 and set the constraints of DSP as

$$t_0 < t_1 < \ldots < t_{K-1} \leq t_K < t_{K+1} < \ldots < t_{2K-1},$$

such that both the dependencies in the forward and backward pass no longer exist, because we do not need them to finish in the same timestamp anymore. The non-decreasing property corresponds to the fact that the data needs to go through bottom layers before top layers, and the error gradient needs to go through top layers before bottom layers.

Based on backpropagation algorithm and Eq. (2), we should compute the gradients according to the following formulas as we are updating the Backward Parameters $\{x_k^{t_{2K-1-k}}\}_{k=0,\ldots,K-1}$ instead,

$$\begin{aligned}
\mathcal{G}_{x_k} &= \frac{\partial F(h_0; x_0^{t_{2K-1}}; ...; x_k^{t_{2K-1-k}})}{\partial x_k^{t_{2K-1-k}}} \mathcal{G}_{h_{k+1}} \\
\mathcal{G}_{h_k} &= \frac{\partial F(h_0; x_0^{t_{2K-1}}; ...; x_k^{t_{2K-1-k}})}{\partial F(h_0; x_0^{t_{2K-1}}; ...; x_{k-1}^{t_{2K-2-k}})} \mathcal{G}_{h_{k+1}} \quad \text{and} \quad \mathcal{G}_{h_K} = \frac{\partial \mathcal{L}(F(h_0; x_0^{t_{2K-1}}; ...; x_{K-1}^{t_K}), l)}{F(h_0; x_0^{t_{2K-1}}; ...; x_{K-1}^{t_K})}.
\end{aligned} \tag{4}$$

However, during the forward pass it is infeasible to acquire information from future timestamps $t_{2K-1}, t_{2K-2}, \ldots, t_K$, and we can only compute activation as $F(h_0; x_0^{t_0}; ...; x_{k-1}^{t_{k-1}})$. Therefore we incorporate the recomputation technique and utilize both the Forward Parameters and Backward Parameters to compute DSP gradient as follows,

$$\begin{aligned}
\mathcal{G}_{x_k} &= \frac{\partial F(h_0; x_0^{t_0}; ...; x_{k-1}^{t_{k-1}}; \boldsymbol{x_k^{t_{2K-1-k}}})}{\partial \boldsymbol{x_k^{t_{2K-1-k}}}} \mathcal{G}_{h_{k+1}} \\
\mathcal{G}_{h_k} &= \frac{\partial F(h_0; x_0^{t_0}; ...; x_{k-1}^{t_{k-1}}; \boldsymbol{x_k^{t_{2K-1-k}}})}{\partial F(h_0; x_0^{t_0}; ...; x_{k-1}^{t_{k-1}})} \mathcal{G}_{h_{k+1}} \quad \text{and} \quad \mathcal{G}_{h_K} = \frac{\partial \mathcal{L}(F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}}), l)}{F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}})}.
\end{aligned} \tag{5}$$

The intuition behind the DSP gradient of Eq. (5) is that it is equivalent to Eq. (4) with parameters $x^*$ where the gradient is zero ($x_k^{t_k} = x_k^{t_{2K-1-k}}$ afterwards), and as the training proceeds the parameters gradually converge to the optima. It is reasonable considering the results on the optimality of the local optima of DNN (Choromanska et al., 2015; Kawaguchi, 2016).

### 3.3 BATCH PIPELINE FOR PARALLEL TRAINING

The computation of DSP gradient breaks the forward and backward dependencies/lockings of the same data as it will not appear in different blocks at the same timestamp. The update locking is naturally broken. Algorithm 1 in the view of the traversal of a single data is explicitly formed based on Figure 2 and Eq. (5). For parallel implementation of DSP, we incorporate data batch pipeline to keep all the blocks being fed with different data batches and running as shown in Figure 3, which is the same as Figure 2 if considering a single data batch's behavior.

The detail of DSP for parallel training (Figure 3, Algorithm 2) is as follows. We let the data source consecutively feeds and pipelines the data input. Different blocks process different data via FIFO queues, as a result the data travels each block at different timestamps. The block $k$ has an input queue $\mathcal{M}_k$, output queue $\mathcal{P}_k$ and gradient queue $\mathcal{Q}_k$ of length $1+m_k$, $1+p_k$ and $1+q_k$ respectively. It gets data from $\mathcal{P}_{k-1}$, stores it into $\mathcal{M}_k$, computes the forward results and saves it into $\mathcal{P}_k$. Then it gets data from $\mathcal{M}_k$ and error gradient from $\mathcal{Q}_{k+1}$ to do forward (called recomputation) and backward, and saves the backward error gradient into $\mathcal{Q}_k$. Note that $\mathcal{P}_{-1}$ is the input training data source, $\mathcal{Q}_K$ contains error gradient directly from loss function, block $K-1$ does not forward the data from $\mathcal{P}_{K-2}$ and block 0 does not save the error gradient. $p_{K-1}$ is 0 because block $K-1$ has no upper block to send output to; $q_0$ is also 0 because the backward ends at block 0. We denote DSP under this setting as $DSP(p_0, ..., p_{K-1}; m_0, ..., m_{K-1})$. Various settings can be chosen as long as the following constraints are satisfied:

$$\begin{cases}
q_k = m_{k-1} - p_{k-1} - m_k > 0 \quad \forall k \in \{1, \ldots, K-1\}, \quad q_0 = 0, \\
m_k > 0 \quad \forall k \in \{0, \ldots, K-1\}, \\
p_k > 0 \quad \forall k \in \{0, \ldots, K-2\}, \quad p_{K-1} = 0.
\end{cases} \tag{6}$$

**Algorithm 1:** DSP in the view of the data traversal

**Input: Data stream** $\mathcal{P}_{-1} \coloneqq \{h_0^n, l^n\}_{n=0}^{N-1}$;
**Initialize:** parameters $x^0$ and learning rate $\{\alpha_n\}_{n=0}^{N-1}$;
$x_k^t = x_k^0 \, (t \leq t_{0,2K-1-k})$;
**for** *n=0,1,...,N-1* **do**
  /* Forward Pass */
  **for** *k=0,1,...,K-1* **do**
    Compute $h_{k+1}^n = f_k(h_k^n; x_k^{t_{n,K}})$;
  /* Backward Pass */
  Compute error gradient $\mathcal{G}_{h_K} = \frac{\partial \mathcal{L}(h_K^n, l^n)}{\partial h_K^n}$;
  **for** *k=K-1,K-2,...,0* **do**
    Compute $\mathcal{G}_{h_k} = \frac{\partial f_k(h_k^n; x_k^{t_{n,2K-1-k}})}{\partial h_k^n} \mathcal{G}_{h_{k+1}}$;
    Compute $\mathcal{G}_{x_k} = \frac{\partial f_k(h_k^n; x_k^{t_{n,2K-1-k}})}{\partial x_k^{t_{n,2K-1-k}}} \mathcal{G}_{h_{k+1}}$;
    Update $x_k^{t_{n,2K-1-k}}$ with gradient $\mathcal{G}_{x_k}$;

**Algorithm 2:** DSP in the view of the current timestamp

**Input: Data stream** $\mathcal{P}_{-1} \coloneqq \{h_0^n, l^n\}_{n=0}^{N-1}$;
**Initialize:** parameters $x^0$ and learning rate $\{\alpha_n\}_{n=0}^{N-1}$;
FIFO queues $\{\mathcal{P}_k\}_{k=0}^{K-1}$, $\{\mathcal{M}_k\}_{k=0}^{K-1}$, $\{\mathcal{Q}_k\}_{k=0}^{K-1}$ of length $\{1+m_k\}_{k=0}^{K-1}$, $\{1+p_k\}_{k=0}^{K-1}$, $\{1+q_k\}_{k=0}^{K-1}$ satisfying (6) and filled with $\{m_k\}_{k=0}^{K-1}$, $\{p_k\}_{k=0}^{K-1}$, $\{q_k\}_{k=0}^{K-1}$ zeros;
Denote $n_k \coloneqq n - \sum_{i=0}^{k-1} p_i$;
**for** *n=0,1,...,N-1* **do**
  **for** *k=0,1,...,K-1 in parallel* **do**
    /* Forward Pass */
    Pop $h_k^{n_k}$ from $\mathcal{P}_{k-1}$ and push into $\mathcal{M}_k$;
    Pop $h_k^{n_k - m_k}$ from $\mathcal{M}_k$;
    **if** *k≠K-1* **then**
      Compute $H = f_k(h_k^{n_k - m_k}; x_k)$ and $h_{k+1}^{n_k} = f_k(h_k^{n_k}; x_k)$ *in parallel*;
      Push $h_{k+1}^{n_k}$ into $\mathcal{P}_k$;
    **else**
      Compute $H = f_k(h_k^{n_k - m_k}; x_k)$;
      Compute $\mathcal{G}_{h_{k+1}} = \frac{\partial \mathcal{L}(H; l^{n_k - m_k})}{\partial H}$;
    /* Backward Pass */
    Pop $\mathcal{G}_{h_{k+1}}$ from $\mathcal{Q}_{k+1}$ if $k \neq K-1$;
    Compute $\mathcal{G}_{h_k} = \frac{\partial H}{\partial h_k^{n_k - m_k}} \mathcal{G}_{h_{k+1}}$ and $\mathcal{G}_{x_k} = \frac{\partial H}{\partial x_k} \mathcal{G}_{h_{k+1}}$;
    Push $\mathcal{G}_{h_k}$ into $\mathcal{Q}_k$ if $k \neq 0$;
    Update $x_k$ with gradient $\mathcal{G}_{x_k}$;

The first constraint of Eq. (6) is to make the error gradient meet the activation of the same data. Besides adopting recomputation to reduce memory consumption, DSP overlaps recomputation with the forward pass to save time. Using queues also make DSP overlap the communication between blocks with computation. The FIFO queues allow for some asynchrony which is effective for dealing with random stragglers.

**Complexity** The ideal time complexity of DSP is $\mathcal{O}(\frac{T_F + T_B}{K})$ and the space complexity is $\mathcal{O}(L + \sum_{k=0}^{K-1}(m_k + p_k + q_k))$, where $T_F$ and $T_B$ are serial forward and backward time, and $L$ is the number of layers. $m_k$ also represents the Layer-wise Staleness of block $k$. $K$ and the FIFO queues length $m_k + 1, p_k + 1, q_k + 1 \ll L$ for deep models, so the extra space cost is trivial.

## 4 Convergence Analysis

The convergence of DSP with SGD is first analyzed, then DSP with Momentum SGD. For simplicity we denote the Forward and Backward Parameters of data $n$ as $x^{n'}$ and $x^n$ respectively.

**Assumption 1.** ***(Bounded variance)*** *Assume that the DSP stochastic gradient $\mathcal{G}(x; \xi)$ satisfies:*

$$Var\left[\mathcal{G}(x; \xi)\right] \leq \sigma^2.$$

Here $\mathbb{E}\left[\mathcal{G}(x; \xi)\right] = \mathcal{G}(x) \neq \nabla f(x)$.

**Assumption 2.** ***(Lipschitz continuous gradient)*** *Assume that the loss and the output of the blocks have Lipschitz continuous gradient, that is, $\forall k \in \{0, 1, .., K-1\}$, and $\forall (x_{0,1}, ..., x_{k,1}), (x_{0,2}, ..., x_{k,2}) \in \mathbb{R}^{d_0 + d_1 + ... + d_k}$,*

$$\left\|\nabla F\left(h_0; x_{0,1}; ...; x_{k,1}\right) - \nabla F(h_0; x_{0,2}; ...; x_{k,2})\right\| \leq L_k \left\|(x_{0,1}, ..., x_{k,1}) - (x_{0,2}, ..., x_{k,2})\right\|,$$

*and $\forall x_1, x_2 \in \mathbb{R}^d$,*

$$\left\|\nabla f(x_1) - \nabla f(x_2)\right\| \leq L_K \left\|x_1 - x_2\right\|.$$

We define $L \coloneqq \max_{k \in \{0,1,...,K\}} L_k$. Note that $\nabla F\left(h_0; x_{0,1}; ...; x_{k,1}\right)$ and $\nabla F(h_0; x_{0,2}; ...; x_{k,2})$ regarding parameters are Jacobian matrices. In fact, this is assuming that the partial model consisted of the blocks that the data has traveled, has Lipschitz continuous gradient.

**Assumption 3.** ***(Bounded error gradient)*** *Assume that the norm of the error gradient that a block receives is bounded, that is, for any $x \in \mathbb{R}^d$, $\forall k \in \{0, 1, ..., K-2\}$,*

$$\left\| \frac{\partial f_{k+1}(h_{k+1}; x_{k+1})}{\partial h_{k+1}} ... \frac{\partial f_{K-1}(h_{K-1}; x_{K-1})}{\partial h_{K-1}} \frac{\partial \mathcal{L}(h_K, l)}{\partial h_K} \right\| \leq M \quad \textit{and} \quad \left\| \frac{\partial \mathcal{L}(h_K, l)}{\partial h_K} \right\| \leq M.$$

This is assuming that the error gradient at each block does not explode. It is natural to make the above two block-wise assumptions as we are breaking the neural networks into blocks.

**Lemma 1.** *If Assumptions 2 and 3 hold, the difference between DSP gradient and BP gradient regarding the parameters of block $k \in \{0, 1, ..., K-1\}$ satisfies:*

$$\left\| \nabla_{x_k} \mathcal{L}(F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}}), y) - \mathcal{G}_{x_k}(x_0^{t_{2K-1}}; ...; x_{K-1}^{t_K}) \right\| \leq LM \sum_{i=k}^{K-1} \left\| x_i^{t_{2K-1-i}} - x_i^{t_i} \right\|.$$

### 4.1 DSP WITH SGD

**Theorem 1.** *Assume Assumptions 1, 2 and 3 hold. Let $c_0 = M^2K(K+1)^2$, and $c_1 = -(\Delta t^2 + 2) + \sqrt{(\Delta t^2+2)^2 + 2c_0\Delta t^2}$. If the learning rate $\alpha_n \leq \frac{c_1}{Lc_0\Delta t^2}$, then*

$$\frac{\sum_{n=0}^{N-1} \alpha_n \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2}{\sum_{n=0}^{N-1} \alpha_n} \leq \frac{2\left[f(x^0) - f^*\right]}{\sum_{n=0}^{N-1} \alpha_n} + \frac{L\sigma^2(2 + K\Delta t^2 + \frac{1}{4}Kc_1) \sum_{n=0}^{N-1} \alpha_n^2}{\sum_{n=0}^{N-1} \alpha_n}.$$

**Corollary 1.1.** *(Sublinear convergence rate) According to Theorem 1, by setting the learning rate $\alpha_n = \min\left\{\frac{1}{\sqrt{N}}, \frac{c_1}{Lc_0\Delta t^2}\right\}$, when N is large enough we have $\alpha_n = \frac{1}{\sqrt{N}}$ and:*

$$\min_{n=0,...,N-1} \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2 \leq \frac{2(f(x^0) - f^*)}{\sqrt{N}} + \frac{L\sigma^2(2 + K\Delta t^2 + \frac{1}{4}Kc_1)}{\sqrt{N}}.$$

**Corollary 1.2.** *According to Theorem 1, if the learning rate $\alpha_n$ diminishes and satisfies the requirements in (Robbins & Monro, 1951): $\lim_{N\to\infty} \sum_{n=0}^{N-1} \alpha_n = \infty$ and $\lim_{N\to\infty} \sum_{n=0}^{N-1} \alpha_n^2 < \infty$, choose $x^n$ randomly from $\{x^n\}_{n=0}^{N-1}$ with probabilities proportional to $\{\alpha_n\}_{n=0}^{N-1}$. Then we can prove that it converges to critical points for the non-convex problem due to $\lim_{n\to\infty} \mathbb{E} \left\| \nabla f(x^n) \right\|^2 = 0$.*

### 4.2 DSP WITH MOMENTUM SGD

**Theorem 2.** *Assume Assumption 1, 2 and 3 hold. Let $c_2 = \frac{((1-\beta)s-1)^2}{(1-\beta)^2}$, $c_3 = M^2K(K+1)^2\Delta t^2(c_2+s^2)$, $c_4 = 3 + \beta^2 c_2 + 2(1-\beta)^2\Delta t^2(c_2+s^2)$, and $c_5 = \frac{2+\beta^2 c_2}{1-\beta} + 2(1-\beta)\Delta t^2(c_2 + s^2) + \frac{-c_4+\sqrt{c_4^2+4(1-\beta)^2c_3}}{2(1-\beta)}$. If the fixed learning rate $\alpha$ satisfies $\alpha \leq \frac{-c_4+\sqrt{c_4^2+4(1-\beta)^2c_3}}{2(1-\beta)c_3L}$, then*

$$\frac{1}{N} \sum_{n=0}^{N-1} \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2 \leq \frac{2(1-\beta)(f(x^0) - f^*)}{N\alpha} + c_5\sigma^2 L\alpha.$$

**Corollary 2.1.** *(Sublinear convergence rate) According to Theorem 2, by setting the learning rate $\alpha = \min\{\frac{1}{\sqrt{N}}, \frac{-c_4+\sqrt{c_4^2+4(1-\beta)^2c_3}}{2(1-\beta)c_3L}\}$, when N is large enough we have $\alpha = \frac{1}{\sqrt{N}}$ and:*

$$\min_{n=0,...,N-1} \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2 \leq \frac{2(1-\beta)(f(x^0) - f^*)}{\sqrt{N}} + \frac{c_5\sigma^2 L}{\sqrt{N}}.$$

**Remark 2.1.** *The convergence performance of DSP is affected by Layer-wise Staleness rather than the staleness between different blocks.*

## 5 EXPERIMENTS

**Experiment Settings** We implement DSP in TensorFlow (Abadi et al., 2016) and run the experiments on Nvidia Tesla P40 GPUs. The model is divided into $K$ blocks and distributed onto $K$ GPUs. Data augmentation procedures include random cropping, random flipping and standardization. We use SGD with the momentum constant of 0.9. In CIFAR experiments, the batch size is 128. We train ResNet98 and ResNet164 for 300 epochs. The weight decay is $5 \times 10^{-4}$ and the initial learning rate

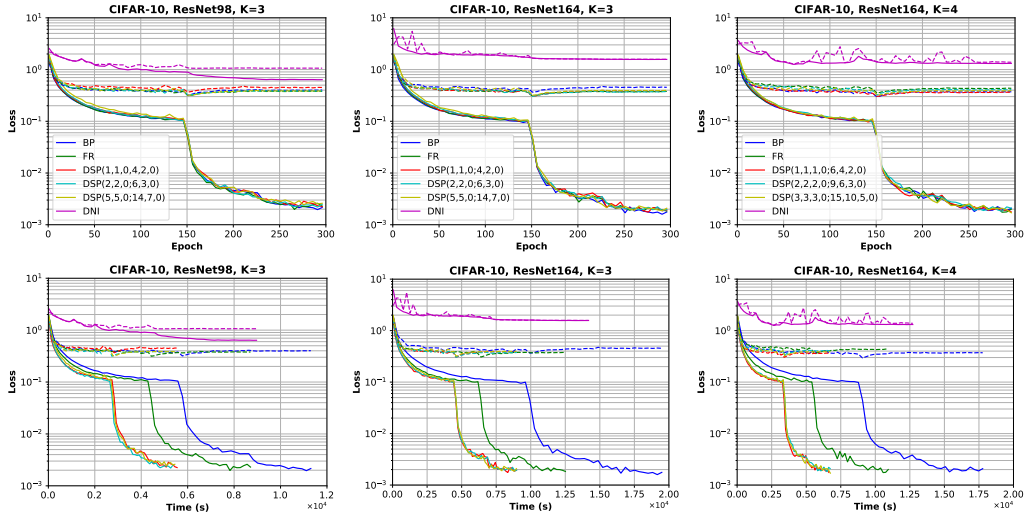


Figure 4: Training loss (solid line) and testing loss (dash line) for ResNet98, ResNet164 on CIFAR-10. The first row and second row plots the loss regarding the training epochs and time respectively.

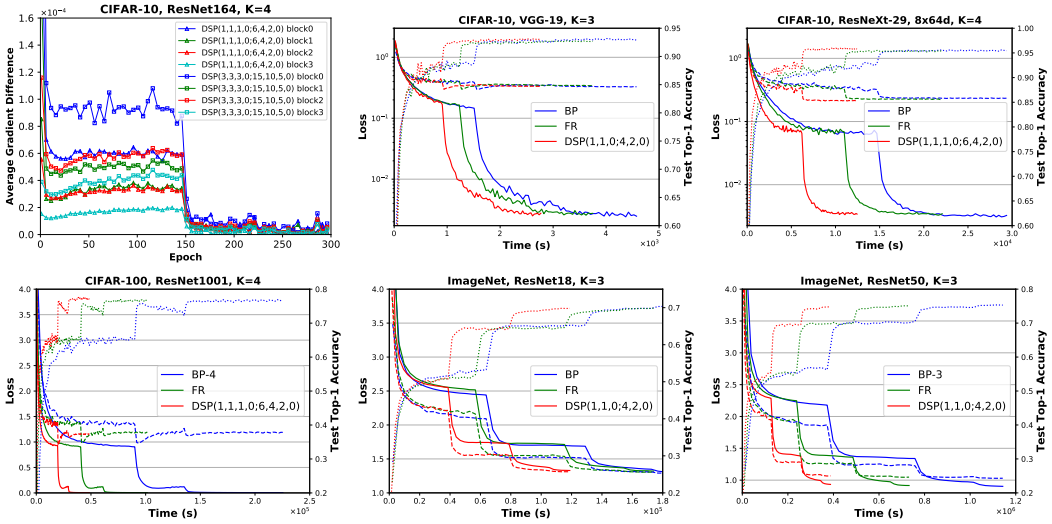


Figure 5: Top left: Average difference of DSP and BP gradient regarding the number of parameters. The rest: Training loss (solid line), testing loss (dash line) and test top-1 accuracy(dot line).

is 0.01 (test performance could be a little lower than 0.1 (Liu et al., 2018)) with a decay of 0.1 at epoch 150, 225; ResNet1001 is trained for 250 epochs. The weight decay is $2 \times 10^{-4}$ and the initial learning rate is 0.1 with a decay of 0.1 at epoch 100, 150, 200; VGG-19 and ResNext-29 are trained for 200 epochs. The weight decay is $5 \times 10^{-4}$ and the initial learning rate is 0.01 with a decay of 0.1 at epoch 100, 150. We also train ResNet on ImageNet for 90 epochs. The batch size is 256, the weight decay is $1 \times 10^{-4}$ and the initial learning rate is 0.1 with a decay of 0.1 at epoch 30, 60, 80. There are four compared methods:

- BP: The standard implementation in TensorFlow. BP (or BP-K) runs on one (or K) GPUs.
- DNI: The Decoupled Neural Interface algorithm in (Jaderberg et al., 2017). The auxiliary network consists of two hidden and one output convolution layers with $5 \times 5$ filters and padding size of 2. The hidden layers also use batch-normalization and ReLU.
- FR: The Features Replay algorithm proposed by (Huo et al., 2018a).
- DSP: Our Diversely Stale Parameters.

Table 1: Speedup Comparison.

| | CIFAR-10 | | | CIFAR-100 | ImageNet | |
|---|---|---|---|---|---|---|
| K, batch size | ResNet164 (4, 128) | ResNext-29 (4, 128) | VGG-19 (3, 128) | ResNet1001 (4, 128) | ResNet50 (3, 256) | ResNet101 (4, 128) |
| BP / BP-K | x1 / - | x1 / - | x1 / - | - / x1 | - / x1 | x1 / - |
| FR | x1.7 | x1.3 | x1.1 | x1.9 | x1.6 | x1.7 |
| GPipe | - | - | - | - | - | x2.2 |
| DSP | **x2.7** | **x2.4** | **x1.5** | **x4.8** | **x3.0** | **x2.7** |

**Faster Convergence** The DSP convergence curves regarding training epochs are nearly the same as FR and BP, while DNI does not converge (Figure 4). But the epoch time of DSP is much less. Due to the overlap of communication and computation, the overheads of DSP are much less than model parallel BP and the speedup can even exceed $K$. To further demonstrate the scalability, we also run experiments on VGG-19 (Simonyan & Zisserman, 2014), ResNeXt-29 (Xie et al., 2017), ResNet1001 and ImageNet (Deng et al., 2009) as shown in Figure 5. The speedup is summarized in Table 1 (GPipe paper only reports speedup of ResNet101 and AmoebaNet-D (4,512)). Note that the implementation of DSP involves some inefficient copy operations due to limited supported features of the deep learning framework, which means DSP could achieve a potentially even faster speedup.

**Stronger robustness** We slow down each GPU by a certain percentage with a probability of $\frac{1}{3}$ and run the experiments on ResNet164 (Table 2). The performance of FR degrades a lot because it does not break the forward locking nor completely decouple the backward pass. DSP is very robust with the best slow down percentage less than $\frac{1}{3}$ of the GPU slow down percentage. Longer queues improve DSP's resilience to random stragglers.

Table 2: Slowdown (CIFAR-10, ResNet164, K=3)

| | Slow down percentage | | |
|---|---|---|---|
| GPU | 20% | 50% | 100% |
| FR | 8.977% | 28.52% | 97.06% |
| DSP(1,1,0;4,2,0) | **6.017%** | 16.14% | 37.44% |
| DSP(2,2,0;6,3,0) | 7.465% | **16.01%** | 36.57% |
| DSP(3,3,0;10,5,0) | 7.391% | 18.15% | **32.10%** |

**Better generalization** Table 3 shows the best top-1 test accuracy. The test performance of DSP is better than BP and FR. From Lemma 1 we know that the DSP gradient deviates from the BP gradient. This difference becomes small as the training proceeds, but could impose small noise and help find a better local minimum.

Table 3: Best Top-1 Test Accuracy (K=3)

| | ResNet164 | | ResNet98 | |
|---|---|---|---|---|
| | CIFAR-10 | CIFAR-100 | CIFAR-10 | CIFAR-100 |
| BP | 92.84% | 70.74% | 92.91% | 69.48% |
| FR | 92.99% | 71.25% | 93.10% | 69.87% |
| DSP(1,1,0;4,2,0) | **93.05%** | 71.05% | 92.86% | 69.59% |
| DSP(2,2,0;6,3,0) | 92.76% | 71.00% | **93.18%** | **70.30%** |
| DSP(3,3,0;10,5,0) | 92.79% | **71.29%** | 92.78% | 69.98% |

**Difference of DSP and BP gradient** We attest our theoretical analysis of Lemma 1 via checking the difference between DSP and BP gradient. From the first figure of Figure 5 we can see that the difference drops very fast as the training proceeds and it drops faster for upper blocks, which confirms the rationality of the DSP gradient. Moreover, the lower blocks suffer from a larger difference, and as the Layer-wise Staleness increases the difference will also increase, which matches Lemma 1 well. As the learning rate drops, the difference drops a lot. This verifies that a smaller learning rate can deal with a larger number of blocks and Layer-wise Staleness in Theorem 1 and 2.

## 6 Conclusion

In this paper, we have proposed Layer-wise Staleness and DSP, a novel way to train neural networks. DSP is proved to converge to critical points for non-convex problems. We apply DSP to train CNNs in parallel and the experiment results confirm our theoretical analysis. Our proposed method achieves significant speedup, resilience to random stragglers, and better generalization. The speedup can exceed $K$ compared with model parallel BP.

## REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 16)*, pp. 265–283, 2016.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. *arXiv preprint arXiv:1901.08164*, 2019.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta (eds.), *Proceedings of COMPSTAT'2010*, pp. 177–186, Heidelberg, 2010. Physica-Verlag HD. ISBN 978-3-7908-2604-3.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192–204, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Andreas Griewank. An implementation of checkpointing for the reverse or adjoint model of differentiation. *ACM Trans. Math. Software*, 26(1):1–19, 1999.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *Advances in neural information processing systems*, pp. 1223–1231, 2013.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *arXiv preprint arXiv:1811.06965*, 2018.

Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. In *Advances in Neural Information Processing Systems*, pp. 6659–6668, 2018a.

Zhouyuan Huo, Bin Gu, qian Yang, and Heng Huang. Decoupled parallel backpropagation with convergence guarantee. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2098–2106, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL `http://proceedings.mlr.press/v80/huo18a.html`.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1627–1635. JMLR. org, 2017.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pp. 586–594, 2016.

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

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Seunghak Lee, Jin Kyu Kim, Xun Zheng, Qirong Ho, Garth A Gibson, and Eric P Xing. On model parallelization and scheduling strategies for distributed machine learning. In *Advances in neural information processing systems*, pp. 2834–2842, 2014.

Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2014.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pp. 1037–1045, 2016.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8): 103–111, 1990.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, 2017.

Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*, 2016.

# A BASIC LEMMAS

**Lemma 1.** *If Assumptions 2 and 3 hold, the difference between DSP gradient and BP gradient regarding the parameters of block $k$ satisfies:*

$$\left\|\nabla_{x_k}\mathcal{L}(F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}}),y)-\mathcal{G}_{x_k}(x_0^{t_{2K-1}};...;x_{K-1}^{t_K})\right\|\leq LM\sum_{i=k}^{K-1}\left\|x_i^{t_{2K-1-i}}-x_i^{t_i}\right\|.$$

*Proof.* We gradually move the DSP gradient of the block $k$ towards the BP gradient by replacing one block's backward parameters with its forward parameters at a time. $K-k$ steps in total are needed, and each step will introduce an error. After all the replacement is done, it becomes the BP gradient at the forward parameters. Firstly we replace $x_k^{t_{2K-1-k}}$ with $x_k^{t_k}$, and calculate the error introduced as follows,

$$\begin{aligned}
\|\Delta_k\| = &\left\|\left(\frac{\partial F(h_0;x_0^{t_0};...;x_{k-1}^{t_{k-1}};x_k^{t_{2K-1-k}})}{\partial x_k^{t_{2K-1-k}}}-\frac{\partial F(h_0;x_0^{t_0};...;x_{k-1}^{t_{k-1}};x_k^{t_k})}{\partial x_k^{t_k}}\right)\cdot\right.\\
&\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{2K-2-k}})}{\partial F(h_0;x_0^{t_0};...;x_k^{t_k})}\cdots\frac{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}};x_{K-1}^{t_K})}{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}})}\cdot\\
&\left.\frac{\partial\mathcal{L}\left(F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}}),l\right)}{\partial F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}})}\right\|\\
\leq &\left\|\frac{\partial F(h_0;x_0^{t_0};...;x_{k-1}^{t_{k-1}};x_k^{t_{2K-1-k}})}{\partial x_k^{t_{2K-1-k}}}-\frac{\partial F(h_0;x_0^{t_0};...;x_{k-1}^{t_{k-1}};x_k^{t_k})}{\partial x_k^{t_k}}\right\|\cdot\\
&\left\|\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{2K-2-k}})}{\partial F(h_0;x_0^{t_0};...;x_k^{t_k})}\cdots\frac{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}};x_{K-1}^{t_K})}{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}})}\cdot\right.\\
&\left.\frac{\partial\mathcal{L}\left(F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}}),l\right)}{\partial F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}})}\right\|\\
\leq &LM\left\|x_k^{t_{2K-1-k}}-x_k^{t_k}\right\|.
\end{aligned}$$

Secondly we replace $x_{k+1}^{t_{2K-2-k}}$ with $x_{k+1}^{t_{k+1}}$, and calculate the error introduced,

$$\begin{aligned}
\|\Delta_{k+1}\| = &\left\|\left(\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{2K-2-k}})}{\partial x_k^{t_k}}-\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{k+1}})}{\partial x_k^{t_k}}\right)\cdot\right.\\
&\frac{\partial F(h_0;x_0^{t_0};...;x_{k+1}^{t_{k+1}};x_{k+2}^{t_{2K-3-k}})}{\partial F(h_0;x_0^{t_0};...;x_{k+1}^{t_{k+1}})}\cdots\frac{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}};x_{K-1}^{t_K})}{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}})}\cdot\\
&\left.\frac{\partial\mathcal{L}\left(F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}}),l\right)}{\partial F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}})}\right\|\\
\leq &\left\|\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{2K-2-k}})}{\partial x_k^{t_k}}-\frac{\partial F(h_0;x_0^{t_0};...;x_k^{t_k};x_{k+1}^{t_{k+1}})}{\partial x_k^{t_k}}\right\|\cdot\\
&\left\|\frac{\partial F(h_0;x_0^{t_0};...;x_{k+1}^{t_{k+1}};x_{k+2}^{t_{2K-3-k}})}{\partial F(h_0;x_0^{t_0};...;x_{k+1}^{t_{k+1}})}\cdots\frac{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}};x_{K-1}^{t_K})}{\partial F(h_0;x_0^{t_0};...;x_{K-2}^{t_{K-2}})}\cdot\right.\\
&\left.\frac{\partial\mathcal{L}\left(F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}}),l\right)}{\partial F(h_0;x_0^{t_0};...;x_{K-1}^{t_{K-1}})}\right\|\\
\leq &LM\left\|x_{k+1}^{t_{2K-2-k}}-x_{k+1}^{t_{k+1}}\right\|.
\end{aligned}$$

We repeatedly perform the above procedure, until we get the error in the last step,

$$\begin{aligned}
\|\Delta_{K-1}\| = & \left\| \left( \frac{\partial F(h_0; x_0^{t_0}; ...; x_{K-2}^{t_{K-2}}; x_{K-1}^{t_K})}{\partial x_k^{t_k}} - \frac{\partial F(h_0; x_0^{t_0}; ...; x_{K-2}^{t_{K-2}}; x_{K-1}^{t_{K-1}})}{\partial x_k^{t_k}} \right) \cdot \right. \\
& \left. \frac{\partial \mathcal{L}\left(F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}}), l\right)}{\partial F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}})} \right\| \\
\leq & \left\| \frac{\partial F(h_0; x_0^{t_0}; ...; x_{K-2}^{t_{K-2}}; x_{K-1}^{t_K})}{\partial x_k^{t_k}} - \frac{\partial F(h_0; x_0^{t_0}; ...; x_{K-2}^{t_{K-2}}; x_{K-1}^{t_{K-1}})}{\partial x_k^{t_k}} \right\| \cdot \\
& \left\| \frac{\partial \mathcal{L}\left(F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}}), l\right)}{\partial F(h_0; x_0^{t_0}; ...; x_{K-1}^{t_{K-1}})} \right\| \\
\leq & LM \left\| x_{K-1}^{t_K} - x_{K-1}^{t_{K-1}} \right\|.
\end{aligned}$$

Add them together and we will have

$$\begin{aligned}
& \left\| \nabla_{x_k} \mathcal{L}(F(h_0; x_0^{t_0}; x_1^{t_1}; ...; x_{K-1}^{t_{K-1}}), l) - \mathcal{G}_{x_k}(x_0^{t_{2K-1}}; x_1^{t_{2K-2}}; ...; x_{K-1}^{t_K}) \right\| \\
= & \|\Delta_k + \Delta_{k+1} + ... + \Delta_{K-1}\| \\
\leq & \|\Delta_k\| + \|\Delta_{k+1}\| + ... + \|\Delta_{K-1}\| \\
\leq & LM \sum_{i=k}^{K-1} \left\| x_i^{t_{2K-1-i}} - x_i^{t_i} \right\|.
\end{aligned}$$

□

**Lemma 2.** *Assume Assumption 2 and 3 exist. The second moment of the difference between DSP and BP gradient satisfies,*

$$\left\| \nabla f(x_0^{t_0}; ...; x_{K-1}^{t_{K-1}}) - \mathcal{G}(x_0^{t_{2K-1}}; ...; x_{K-1}^{t_K}) \right\|^2 \leq \frac{1}{2} L^2 c_0 \sum_{k=0}^{K-1} \frac{k+1}{K+1} \left\| x_k^{t_{2K-1-k}} - x_k^{t_k} \right\|^2.$$

*Proof.* Via summation of Lemma 1 we can get,

$$\left\| \nabla f(x_0^{t_0}; x_1^{t_1}; ...; x_{K-1}^{t_{K-1}}) - \mathcal{G}(x_0^{t_{2K-1}}; x_1^{t_{2K-2}}; ...; x_{K-1}^{t_K}) \right\| \leq LM \sum_{k=0}^{K-1} (k+1) \left\| x_k^{t_{2K-1-k}} - x_k^{t_k} \right\|.$$

Then we have,

$$
\begin{aligned}
&\left\|\nabla f(x_0^{t_0}; x_1^{t_1}; ...; x_{K-1}^{t_{K-1}}) - \mathcal{G}(x_0^{t_{2K-1}}; x_1^{t_{2K-2}}; ...; x_{K-1}^{t_K})\right\|^2 \\
&\leq L^2 M^2 \left(\sum_{k=0}^{K-1} (k+1) \left\|x_k^{t_{2K-1-k}} - x_k^{t_k}\right\|\right)^2 \\
&= L^2 M^2 \left(\sum_{k=0}^{K-1} (k+1)\right)^2 \left(\sum_{k=0}^{K-1} \frac{k+1}{\sum_{k=0}^{K-1}(k+1)} \left\|x_k^{t_{2K-1-k}} - x_k^{t_k}\right\|\right)^2 \\
&\leq L^2 M^2 \left(\sum_{k=0}^{K-1} (k+1)\right)^2 \sum_{k=0}^{K-1} \frac{k+1}{\sum_{k=0}^{K-1}(k+1)} \left\|x_k^{t_{2K-1-k}} - x_k^{t_k}\right\|^2 \\
&= \frac{1}{2} L^2 M^2 K(K+1) \sum_{k=0}^{K-1} (k+1) \left\|x_k^{t_{2K-1-k}} - x_k^{t_k}\right\|^2.
\end{aligned}
$$

□

## B DSP WITH SGD

**Theorem 1.** *Assume Assumptions 1, 2 and 3 hold. Let* $c_0 = M^2K(K+1)^2$, *and* $c_1 = -(\Delta t^2 + 2) + \sqrt{(\Delta t^2+2)^2 + 2c_0\Delta t^2}$. *If the learning rate* $\alpha_n \leq \frac{c_1}{Lc_0\Delta t^2}$, *then*

$$
\frac{\sum_{n=0}^{N-1} \alpha_n \mathbb{E}\left\|\nabla f(x^{n'})\right\|^2}{\sum_{n=0}^{N-1} \alpha_n} \leq \frac{2\left[f(x^0) - f^*\right]}{\sum_{n=0}^{N-1} \alpha_n} + \frac{L\sigma^2(2 + K\Delta t^2 + \frac{1}{4}Kc_1)\sum_{n=0}^{N-1} \alpha_n^2}{\sum_{n=0}^{N-1} \alpha_n}.
$$

*Proof.* According to Lipschitz continuous, we have

$$
\begin{aligned}
&f(x^{n+1}) - f(x^n) \leq \left\langle \nabla f(x^n), x^{n+1} - x^n \right\rangle + \frac{L}{2}\left\|x^{n+1} - x^n\right\|^2 \\
&= -\alpha_n \left\langle \nabla f(x^n), \mathcal{G}(x^n;\xi) \right\rangle + \frac{L\alpha_n^2}{2}\left\|\mathcal{G}(x^n;\xi)\right\|^2 \\
&= -\alpha_n \left\langle \nabla f(x^n) - \nabla f(x^{n'}), \mathcal{G}(x^n;\xi) \right\rangle - \alpha_n \left\langle \nabla f(x^{n'}), \mathcal{G}(x^n;\xi) \right\rangle + \frac{L\alpha_n^2}{2}\left\|\mathcal{G}(x^n;\xi)\right\|^2 \\
&\leq \frac{1}{2L}\left\|\nabla f(x^n) - \nabla f(x^{n'})\right\|^2 + \frac{L\alpha_n^2}{2}\left\|\mathcal{G}(x^n;\xi)\right\|^2 - \alpha_n \left\langle \nabla f(x^{n'}), \mathcal{G}(x^n;\xi) \right\rangle \\
&\quad + \frac{L\alpha_n^2}{2}\left\|\mathcal{G}(x^n;\xi)\right\|^2 \\
&\leq \frac{L}{2}\left\|x^n - x^{n'}\right\|^2 - \alpha_n \left\langle \nabla f(x^{n'}), \mathcal{G}(x^n;\xi) \right\rangle + L\alpha_n^2 \left\|\mathcal{G}(x^n;\xi)\right\|^2.
\end{aligned}
$$

Take expectation regarding $\xi$ on both sides,

$$
\begin{aligned}
&\mathbb{E}\left[f(x^{n+1})\right] - f(x^n) \leq \frac{L}{2}\left\|x^n - x^{n'}\right\|^2 - \alpha_n\left\langle \nabla f(x^{n'}), \mathcal{G}(x^n)\right\rangle + L\alpha_n^2 \mathbb{E}\left\|\mathcal{G}(x^n;\xi)\right\|^2 \\
&= \frac{L}{2}\left\|x^n - x^{n'}\right\|^2 + \frac{\alpha_n}{2}\left(\left\|\nabla f(x^{n'}) - \mathcal{G}(x^n)\right\|^2 - \left\|\nabla f(x^{n'})\right\|^2 - \left\|\mathcal{G}(x^n)\right\|^2\right) \\
&\quad + L\alpha_n^2\left(\left\|\mathcal{G}(x^n)\right\|^2 + \mathrm{Var}\left[\mathcal{G}(x^n;\xi)\right]\right) \\
&\leq \frac{L}{2}\left\|x^n - x^{n'}\right\|^2 + \frac{\alpha_n}{2}\left\|\nabla f(x^{n'}) - \mathcal{G}(x^n)\right\|^2 - \left(\frac{\alpha_n}{2} - L\alpha_n^2\right)\left\|\mathcal{G}(x^n)\right\|^2 \\
&\quad - \frac{\alpha_n}{2}\left\|\nabla f(x^{n'})\right\|^2 + L\alpha_n^2\sigma^2 \\
&\leq \sum_{k=0}^{K-1}\left[\frac{L}{2} + \frac{1}{4}\alpha_n L^2 M^2 K(K+1)(k+1)\right]\left\|x_k^n - x_k^{n'}\right\|^2 - \left(\frac{\alpha_n}{2} - L\alpha_n^2\right)\left\|\mathcal{G}(x^n)\right\|^2 \\
&\quad - \frac{\alpha_n}{2}\left\|\nabla f(x^{n'})\right\|^2 + L\alpha_n^2\sigma^2.
\end{aligned}
$$

The last inequality utilizes Lemma 2. Consider the first term and take expectation,

$$
\begin{aligned}
\mathbb{E}\left\|x_k^n - x_k^{n'}\right\|^2 &= \mathbb{E}\left\|\sum_{i=n-\Delta t_k}^{n-1} -\alpha_i \mathcal{G}_{x_k}(x^i;\xi)\right\|^2 \\
&\leq \Delta t_k \sum_{i=n-\Delta t_k}^{n-1} \alpha_i^2 \mathbb{E}\left\|\mathcal{G}_{x^k}(x^i;\xi)\right\|^2 \\
&\leq \Delta t \sum_{i=n-\Delta t}^{n-1} \alpha_i^2\left(\left\|\mathcal{G}_{x_k}(x^i)\right\|^2 + \sigma^2\right).
\end{aligned}
$$

Take the total expectation and perform summation for it,

$$
\begin{aligned}
&\sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\frac{L}{2} + \frac{1}{4}\alpha_n L^2 M^2 K(K+1)(k+1)\right)\mathbb{E}\left\|x_k^n - x_k^{n'}\right\|^2 \\
&\leq \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\frac{L}{2} + \frac{1}{4}\alpha_n L^2 M^2 K(K+1)(k+1)\right)\Delta t \sum_{i=n-\Delta t}^{n-1}\alpha_i^2\left(\mathbb{E}\left\|\mathcal{G}_{x_k}(x^i)\right\|^2 + \sigma^2\right) \\
&\leq \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\frac{L}{2} + \frac{1}{4}\alpha_n L^2 M^2 K(K+1)(k+1)\right)\Delta t \cdot \Delta t \cdot \alpha_n^2\left(\mathbb{E}\left\|\mathcal{G}_{x_k}(x^n)\right\|^2 + \sigma^2\right).
\end{aligned}
$$

Take the total expectation and perform summation for all the terms,

$$
\begin{aligned}
&\mathbb{E}\left[f(x^N)\right] - f(x^0) \\
&\leq \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\frac{L}{2}+\frac{1}{4}\alpha_n L^2M^2K(K+1)(k+1)\right)\Delta t^2\alpha_n^2\left(\mathbb{E}\left\|\mathcal{G}_{x_k}(x^n)\right\|^2+\sigma^2\right) \\
&\quad - \sum_{n=0}^{N-1}\left(\frac{\alpha_n}{2}-L\alpha_n^2\right)\mathbb{E}\sum_{k=0}^{K-1}\left\|\mathcal{G}_{x_k}(x^n)\right\|^2 - \sum_{n=0}^{N-1}\frac{\alpha_n}{2}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2 + L\sigma^2\sum_{n=0}^{N-1}\alpha_n^2 \\
&= \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\left(\frac{L}{2}+\frac{1}{4}\alpha_n L^2M^2K(K+1)(k+1)\right)\Delta t^2\alpha_n^2 - \frac{\alpha_n}{2}+L\alpha_n^2\right)\mathbb{E}\left\|\mathcal{G}_{x_k}(x^n)\right\|^2 \\
&\quad + \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\left(\frac{L}{2}+\frac{1}{4}\alpha_n L^2M^2K(K+1)(k+1)\right)\Delta t^2\alpha_n^2\sigma^2 - \sum_{n=0}^{N-1}\frac{\alpha_n}{2}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2 \\
&\quad + L\sigma^2\sum_{n=0}^{N-1}\alpha_n^2 \\
&\leq \sum_{n=0}^{N-1}\sum_{k=0}^{K-1}\frac{1}{4}\alpha_n\left(L^2M^2K(K+1)^2\Delta t^2\alpha_n^2 + \left(2\Delta t^2+4\right)L\alpha_n - 2\right)\mathbb{E}\left\|\mathcal{G}_{x^k}(x^n)\right\|^2 \\
&\quad + \sum_{n=0}^{N-1}\left(\frac{1}{2}LK+\frac{1}{8}\alpha_n L^2M^2K^2(K+1)^2\right)\Delta t^2\alpha_n^2\sigma^2 - \sum_{n=0}^{N-1}\frac{\alpha_n}{2}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2 + L\sigma^2\sum_{n=0}^{N-1}\alpha_n^2 \\
&\leq \sum_{n=0}^{N-1}\left(\frac{1}{2}LK+\frac{1}{8}\alpha_n L^2M^2K^2(K+1)^2\right)\Delta t^2\alpha_n^2\sigma^2 - \sum_{n=0}^{N-1}\frac{\alpha_n}{2}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2 + L\sigma^2\sum_{n=0}^{N-1}\alpha_n^2.
\end{aligned}
$$

The last inequality utilizes the restriction on the learning rate. Then we have

$$
\begin{aligned}
&\frac{\sum_{n=0}^{N-1}\alpha_n\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2}{\sum_{n=0}^{N-1}\alpha_n} \\
&\leq \frac{2\left[f(x^0)-f^*\right]}{\sum_{n=0}^{N-1}\alpha_n} + \frac{L\sigma^2\sum_{n=0}^{N-1}\alpha_n^2\left[2+K\Delta t^2+\frac{1}{4}\alpha_n LM^2K^2(K+1)^2\Delta t^2\right]}{\sum_{n=0}^{N-1}\alpha_n}.
\end{aligned}
$$

□

## C DSP WITH MOMENTUM SGD

The SUM method also implies the following recursions,

$$
\begin{aligned}
x^{n+1}+\frac{\beta}{1-\beta}v^{n+1} &= x^n+\frac{\beta}{1-\beta}v^n-\frac{\alpha}{1-\beta}\mathcal{G}(x^n;\xi), \quad n\geq 0 \\
v^{n+1} &= \beta v^n+((1-\beta)s-1)\alpha\mathcal{G}(x^n;\xi), \quad n\geq 0.
\end{aligned} \tag{7}
$$

where $v^n$ is given by

$$
v^n=\begin{cases} x^n-x^{n-1}+s\alpha\mathcal{G}(x^{n-1};\xi), & n\geq 1 \\ 0, & n=0. \end{cases} \tag{8}
$$

Let $z^n = x^n + \frac{\beta}{1-\beta}v^n$.

**Lemma 3.** *Assume Assumption 1 exists. Let* $c_2 = \frac{((1-\beta)s-1)^2}{(1-\beta)^2}$*, then*

$$\sum_{n=0}^{N-1} \mathbb{E} \left\| v^n \right\|^2 \leq c_2 \alpha^2 \sum_{n=0}^{N-1} \mathbb{E} \left\| \mathcal{G}(x^n) \right\|^2 + c_2 \sigma^2 \alpha^2 N.$$

*Proof.* Let $\hat{\alpha} = ((1-\beta)s - 1)\alpha$. From Eq. (7),

$$v^{n+1} = \beta v^n + \hat{\alpha} \mathcal{G}(x^n; \xi).$$

Note that $v^0 = 0$. Then

$$v^n = \hat{\alpha} \sum_{i=0}^{n-1} \beta^{n-1-i} \mathcal{G}(x^i; \xi).$$

Then we have,

$$\begin{aligned}
\mathbb{E} \left\| v^n \right\|^2 &= \hat{\alpha}^2 \mathbb{E} \left\| \sum_{i=0}^{n-1} \beta^{n-1-i} \mathcal{G}(x^i; \xi) \right\|^2 = \hat{\alpha}^2 \left( \sum_{i=0}^{n-1} \beta^{n-1-i} \right)^2 \mathbb{E} \left\| \sum_{i=0}^{n-1} \frac{\beta^{n-1-i}}{\sum_{i=0}^{n-1} \beta^{n-1-i}} \mathcal{G}(x^i; \xi) \right\|^2 \\
&\leq \hat{\alpha}^2 \left( \sum_{i=0}^{n-1} \beta^{n-1-i} \right)^2 \sum_{i=0}^{n-1} \frac{\beta^{n-1-i}}{\sum_{i=0}^{n-1} \beta^{n-1-i}} \mathbb{E} \left\| \mathcal{G}(x^i; \xi) \right\|^2 \\
&= \hat{\alpha}^2 \sum_{i=0}^{n-1} \beta^{n-1-i} \sum_{i=0}^{n-1} \beta^{n-1-i} \left\| \mathcal{G}(x^i) \right\|^2 + \hat{\alpha}^2 \sigma^2 \left( \sum_{i=0}^{n-1} \beta^{n-1-i} \right)^2 \\
&\leq \frac{\hat{\alpha}^2}{1-\beta} \sum_{i=0}^{n-1} \beta^{n-1-i} \left\| \mathcal{G}(x^i) \right\|^2 + \frac{\hat{\alpha}^2 \sigma^2}{(1-\beta)^2} \\
&= (1-\beta) c_2 \alpha^2 \sum_{i=0}^{n-1} \beta^{n-1-i} \left\| \mathcal{G}(x^i) \right\|^2 + c_2 \alpha^2 \sigma^2.
\end{aligned}$$

Take the total expectation and perform summation,

$$\begin{aligned}
\sum_{n=0}^{N-1} \mathbb{E} \left[ \left\| v^n \right\|^2 \right] &\leq (1-\beta) c_2 \alpha^2 \sum_{n=0}^{N-1} \sum_{i=0}^{n-1} \beta^{n-1-i} \mathbb{E} \left\| \mathcal{G}(x^i) \right\|^2 + c_2 \alpha^2 \sigma^2 N \\
&= (1-\beta) c_2 \alpha^2 \sum_{i=0}^{N-2} \sum_{n=i+1}^{N-1} \beta^{n-1-i} \mathbb{E} \left\| \mathcal{G}(x^i) \right\|^2 + c_2 \alpha^2 \sigma^2 N \\
&= (1-\beta) c_2 \alpha^2 \sum_{i=0}^{N-2} \frac{1-\beta^{N-1-i}}{1-\beta} \mathbb{E} \left\| \mathcal{G}(x^i) \right\|^2 + c_2 \alpha^2 \sigma^2 N \\
&\leq c_2 \alpha^2 \sum_{n=0}^{N-2} \mathbb{E} \left\| \mathcal{G}(x^n) \right\|^2 + c_2 \sigma^2 \alpha^2 N \leq c_2 \alpha^2 \sum_{n=0}^{N-1} \mathbb{E} \left\| \mathcal{G}(x^n) \right\|^2 + c_2 \sigma^2 \alpha^2 N.
\end{aligned}$$

□

**Lemma 4.** *Assume Assumption 1 exists, then*

$$\sum_{n=0}^{N-1} \mathbb{E} \left\| x^n - x^{n'} \right\|^2 \leq 2\Delta t^2 (c_2 + s^2) \alpha^2 \sum_{n=0}^{N-1} \mathbb{E} \left\| \mathcal{G}(x^n) \right\|^2 + 2\Delta t^2 \sigma^2 (c_2 + s^2) \alpha^2 N.$$

*Proof.* First take expectation regarding $\xi$,

$$\begin{aligned}
\mathbb{E}\left\|x^n - x^{n'}\right\|^2 &= \sum_{k=0}^{K-1} \mathbb{E}\left\|x_k^n - x_k^{n'}\right\|^2 = \sum_{k=0}^{K-1} \mathbb{E}\left\|\sum_{i=n-\Delta t_k}^{n-1} v_k^{i+1} - s\alpha \mathcal{G}_{x_k}(x^i;\xi)\right\|^2 \\
&\leq \sum_{k=0}^{K-1} \Delta t_k \sum_{i=n-\Delta t_k}^{n-1} \mathbb{E}\left\|v_k^{i+1} - s\alpha \mathcal{G}_{x_k}(x^i;\xi)\right\|^2 \\
&\leq \sum_{k=0}^{K-1} 2\Delta t_k \sum_{i=n-\Delta t_k}^{n-1} \left(\mathbb{E}\left\|v_k^{i+1}\right\|^2 + s^2\alpha^2 \mathbb{E}\left\|\mathcal{G}_{x_k}(x^i;\xi)\right\|^2\right) \\
&\leq \sum_{k=0}^{K-1} 2\Delta t \sum_{i=n-\Delta t}^{n-1} \left(\mathbb{E}\left\|v_k^{i+1}\right\|^2 + s^2\alpha^2 \mathbb{E}\left\|\mathcal{G}_{x_k}(x^i;\xi)\right\|^2\right) \\
&= 2\Delta t \sum_{i=n-\Delta t}^{n-1} \left(\mathbb{E}\left\|v^{i+1}\right\|^2 + s^2\alpha^2 \mathbb{E}\left\|\mathcal{G}(x^i;\xi)\right\|^2\right) \\
&\leq 2\Delta t \sum_{i=n-\Delta t}^{n-1} \left(\mathbb{E}\left\|v^{i+1}\right\|^2 + s^2\alpha^2 \left\|\mathcal{G}(x^i)\right\|^2 + s^2\alpha^2\sigma^2\right).
\end{aligned}$$

Take total expectation on both sides and perform summation,

$$\begin{aligned}
\sum_{n=0}^{N-1} \mathbb{E}\left\|x^n - x^{n'}\right\|^2 &\leq 2\Delta t \sum_{n=0}^{N-1} \sum_{i=n-\Delta t}^{n-1} \left(\mathbb{E}\left\|v^{i+1}\right\|^2 + s^2\alpha^2 \mathbb{E}\left\|\mathcal{G}(x^i)\right\|^2 + s^2\alpha^2\sigma^2\right) \\
&\leq 2\Delta t^2 \sum_{n=0}^{N-2} \left(\mathbb{E}\left\|v^{n+1}\right\|^2 + s^2\alpha^2 \mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2 + s^2\alpha^2\sigma^2\right) \\
&\leq 2\Delta t^2 \sum_{n=0}^{N-1} \mathbb{E}\left\|v^n\right\|^2 + 2\Delta t^2 s^2\alpha^2 \sum_{n=0}^{N-1} \mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2 + 2\Delta t^2 s^2\alpha^2\sigma^2 N \\
&\leq 2\Delta t^2 (c_2 + s^2)\alpha^2 \sum_{n=0}^{N-1} \mathbb{E}\left[\left\|\mathcal{G}(x^n)\right\|^2\right] + 2\Delta t^2\sigma^2(c_2+s^2)\alpha^2 N.
\end{aligned}$$

□

**Theorem 2.** *Assume Assumption 1, 2 and 3 hold. Let* $c_2 = \frac{((1-\beta)s-1)^2}{(1-\beta)^2}$, $c_3 = M^2K(K+1)^2\Delta t^2(c_2+s^2)$, $c_4 = 3 + \beta^2 c_2 + 2(1-\beta)^2\Delta t^2(c_2+s^2)$, *and* $c_5 = \frac{2+\beta^2 c_2}{1-\beta} + 2(1-\beta)\Delta t^2(c_2 + s^2) + \frac{-c_4+\sqrt{c_4^2+4(1-\beta)^2 c_3}}{2(1-\beta)}$. *If the learning rate* $\alpha$ *is fixed and satisfies* $\alpha \leq \frac{-c_4+\sqrt{c_4^2+4(1-\beta)^2 c_3}}{2(1-\beta)c_3 L}$, *then*

$$\frac{1}{N}\sum_{n=0}^{N-1} \mathbb{E}\left\|\nabla f(x^{n'})\right\|^2 \leq \frac{2(1-\beta)(f(x^0) - f^*)}{N\alpha} + c_5\sigma^2 L\alpha.$$

*Proof.* According to Lipschitz continuous gradient,

$$
\begin{aligned}
& f(z^{n+1}) - f(z^n) \\
& \leq \left\langle \nabla f(z^n), z^{n+1} - z^n \right\rangle + \frac{L}{2} \left\| z^{n+1} - z^n \right\|^2 \\
& = -\frac{\alpha}{1-\beta} \left\langle \nabla f(z^n), \mathcal{G}(x^n;\xi) \right\rangle + \frac{L\alpha^2}{2(1-\beta)^2} \left\| \mathcal{G}(x^n;\xi) \right\|^2 \\
& = -\frac{\alpha}{1-\beta} \left\langle \nabla f(z^n) - \nabla f(x^n), \mathcal{G}(x^n;\xi) \right\rangle - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^n), \mathcal{G}(x^n;\xi) \right\rangle \\
& \quad + \frac{L\alpha^2}{2(1-\beta)^2} \left\| \mathcal{G}(x^n;\xi) \right\|^2 \\
& \leq \frac{1}{2} \left( \frac{1}{L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n;\xi) \right\|^2 \right) \\
& \quad - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^n), \mathcal{G}(x^n;\xi) \right\rangle + \frac{L\alpha^2}{2(1-\beta)^2} \left\| \mathcal{G}(x^n;\xi) \right\|^2 \\
& = \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^n), \mathcal{G}(x^n;\xi) \right\rangle + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n;\xi) \right\|^2 .
\end{aligned}
$$

Take expectation regarding $\xi$ on both sides,

$$
\begin{aligned}
& \mathbb{E}\left[ f(z^{n+1}) \right] - f(z^n) \\
& \leq \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^n), \mathcal{G}(x^n) \right\rangle + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \sigma^2 \\
& = \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^n) - \nabla f(x^{n'}), \mathcal{G}(x^n) \right\rangle \\
& \quad - \frac{\alpha}{1-\beta} \left\langle \nabla f(x^{n'}), \mathcal{G}(x^n) \right\rangle + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \sigma^2 \\
& \leq \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 + \frac{1}{2} \left( \frac{1}{L} \left\| \nabla f(x^n) - \nabla f(x^{n'}) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n) \right\|^2 \right) \\
& \quad + \frac{\alpha}{2(1-\beta)} \left( \left\| \nabla f(x^{n'}) - \mathcal{G}(x^n) \right\|^2 - \left\| \nabla f(x^{n'}) \right\|^2 - \left\| \mathcal{G}(x^n) \right\|^2 \right) \\
& \quad + \frac{L\alpha^2}{(1-\beta)^2} \left\| \mathcal{G}(x^n) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \sigma^2 \\
& = -\frac{\alpha}{2(1-\beta)} \left\| \nabla f(x^{n'}) \right\|^2 + \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 + \frac{1}{2L} \left\| \nabla f(x^n) - \nabla f(x^{n'}) \right\|^2 \\
& \quad + \frac{\alpha}{2(1-\beta)} \left\| \nabla f(x^{n'}) - \mathcal{G}(x^n) \right\|^2 - \left( \frac{\alpha}{2(1-\beta)} - \frac{3L\alpha^2}{2(1-\beta)^2} \right) \left\| \mathcal{G}(x^n) \right\|^2 + \frac{L\alpha^2}{(1-\beta)^2} \sigma^2 .
\end{aligned}
$$

Take the total expectation and perform summation,

$$
\sum_{n=0}^{N-1} \mathbb{E} \left[ \frac{1}{2L} \left\| \nabla f(z^n) - \nabla f(x^n) \right\|^2 \right] \leq \sum_{n=0}^{N-1} \frac{L}{2} \mathbb{E} \left\| z^n - x^n \right\|^2 = \sum_{n=0}^{N-1} \frac{L\beta^2}{2(1-\beta)^2} \mathbb{E} \left\| v^n \right\|^2 .
$$

$$
\begin{aligned}
&\sum_{n=0}^{N-1}\mathbb{E}\left[\frac{1}{2L}\left\|\nabla f(x^n)-\nabla f(x^{n'})\right\|^2+\frac{\alpha}{2(1-\beta)}\left\|\nabla f(x^{n'})-\mathcal{G}(x^n)\right\|^2\right]\\
&\le\sum_{n=0}^{N-1}\frac{L}{2}\mathbb{E}\left\|x^n-x^{n'}\right\|^2+\frac{\alpha}{4(1-\beta)}L^2M^2K(K+1)\sum_{k=0}^{K-1}(k+1)\sum_{n=0}^{N-1}\mathbb{E}\left\|x_k^n-x_k^{n'}\right\|^2\\
&\le\sum_{n=0}^{N-1}\frac{L}{2}\mathbb{E}\left\|x^n-x^{n'}\right\|^2+\frac{\alpha}{4(1-\beta)}L^2M^2K(K+1)^2\sum_{n=0}^{N-1}\mathbb{E}\left\|x^n-x^{n'}\right\|^2\\
&\le\sum_{n=0}^{N-1}\frac{L}{2}\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\mathbb{E}\left\|x^n-x^{n'}\right\|^2.
\end{aligned}
$$

Then we have,

$$
\begin{aligned}
&\mathbb{E}\left[f(z^N)\right]-f(z^0)\\
&\le-\frac{\alpha}{2(1-\beta)}\sum_{n=0}^{N-1}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2-\left(\frac{\alpha}{2(1-\beta)}-\frac{3L\alpha^2}{2(1-\beta)^2}\right)\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2+\frac{L\sigma^2\alpha^2}{(1-\beta)^2}N\\
&\quad+\sum_{n=0}^{N-1}\frac{L\beta^2}{2(1-\beta)^2}\mathbb{E}\left\|v^n\right\|^2+\sum_{n=0}^{N-1}\frac{L}{2}\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\mathbb{E}\left\|x^n-x^{n'}\right\|^2\\
&\le-\frac{\alpha}{2(1-\beta)}\sum_{n=0}^{N-1}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2-\left(\frac{\alpha}{2(1-\beta)}-\frac{3L\alpha^2}{2(1-\beta)^2}\right)\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2+\frac{L\sigma^2\alpha^2}{(1-\beta)^2}N\\
&\quad+\frac{L\beta^2}{2(1-\beta)^2}\left(c_2\alpha^2\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2+c_2\sigma^2\alpha^2N\right)\\
&\quad+\frac{L}{2}\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\cdot\\
&\qquad\left[2\Delta t^2(c_2+s^2)\alpha^2\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2+2\Delta t^2\sigma^2(c_2+s^2)\alpha^2N\right]\\
&=-\frac{\alpha}{2(1-\beta)}\sum_{n=0}^{N-1}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2-\left[\frac{\alpha}{2(1-\beta)}-\alpha^2\left(\frac{3L}{2(1-\beta)^2}+\frac{L\beta^2c_2}{2(1-\beta)^2}+\right.\right.\\
&\qquad\left.\left.L\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\Delta t^2(c_2+s^2)\right)\right]\cdot\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2\\
&\quad+\sigma^2\alpha^2N\left[\frac{L}{(1-\beta)^2}+\frac{L\beta^2c_2}{2(1-\beta)^2}+L\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\Delta t^2(c_2+s^2)\right]\\
&=-\frac{\alpha}{2(1-\beta)}\sum_{n=0}^{N-1}\mathbb{E}\left\|\nabla f(x^{n'})\right\|^2+\frac{\alpha}{2(1-\beta)^2}[(1-\beta)M^2K(K+1)^2\Delta t^2(c_2+s^2)L^2\alpha^2+\\
&\qquad\left(3+\beta^2c_2+2(1-\beta)^2\Delta t^2(c_2+s^2)\right)L\alpha-(1-\beta)]\cdot\sum_{n=0}^{N-1}\mathbb{E}\left\|\mathcal{G}(x^n)\right\|^2\\
&\quad+\sigma^2\alpha^2N\left[\frac{L}{(1-\beta)^2}+\frac{L\beta^2c_2}{2(1-\beta)^2}+L\left(1+\frac{\alpha}{2(1-\beta)}LM^2K(K+1)^2\right)\Delta t^2(c_2+s^2)\right].
\end{aligned}
$$

The second inequality utilizes Lemma 3 and 4. According to the restriction on the learning rate, we can remove the second term in the last equality,

$$f_* - f(x^0) \leq -\frac{\alpha}{2(1-\beta)} \sum_{n=0}^{N-1} \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2 + \sigma^2 L \alpha^2 N \left[ \frac{1}{(1-\beta)^2} + \frac{\beta^2 c_2}{2(1-\beta)^2} + \left( 1 + \frac{\alpha}{2(1-\beta)} LM^2 K (K+1)^2 \right) \Delta t^2 (c_2 + s^2) \right].$$

Therefore we have,

$$\frac{1}{N} \sum_{n=0}^{N-1} \mathbb{E} \left\| \nabla f(x^{n'}) \right\|^2 \leq \frac{2(1-\beta)(f^* - f(x^0))}{N\alpha} + \sigma^2 L \alpha \left[ \frac{2 + \beta^2 c_2}{1-\beta} + \left( 2(1-\beta) + \alpha L M^2 K (K+1)^2 \right) \Delta t^2 (c_2 + s^2) \right].$$

□