DOUBLE MOMENTUM SGD FOR FEDERATED LEARNING

A PREPRINT

An Xu, Heng Huang
Department of Electrical and Computer Engineering
University of Pittsburgh
{an.xu, heng.huang}@pitt.edu

Feb 5, 2021

ABSTRACT

Communication efficiency is crucial in federated learning. Conducting many local training steps in clients to reduce the communication frequency between clients and the server is a common method to address this issue. However, the client drift problem arises as the non-i.i.d. data distributions in different clients can severely deteriorate the performance of federated learning. In this work, we propose a new SGD variant named as DOMO to improve the model performance in federated learning, where double momentum buffers are maintained. One momentum buffer tracks the server update direction, while the other tracks the local update direction. We introduce a novel server momentum fusion technique to coordinate the server and local momentum SGD. We also provide the first theoretical analysis involving both the server and local momentum SGD. Extensive experimental results show a better model performance of DOMO than FedAvg and existing momentum SGD variants in federated learning tasks.

1 Introduction

With deep learning models becoming prevalent but data-hungry, data privacy emerges as an important issue. To preserve data privacy without losing access to massive data, federated learning [15] was proposed. To avoid prohibitively gathering data to the data center, the server only communicates the model weights and its update with the participating clients. Therefore, the participating clients can keep their data private, locally train its model, and then send the local model update back to update the server model at the server node and conclude one training round.

However, training a deep learning model requires many training iterations to converge. Unlike workers in data-center distributed training with large network bandwidth and relatively low communication delay, the clients participating in the collaborative federated learning system can be faced with much more unstable conditions and slower network links due to geo-distribution. Typically, [1] showed that one communication round in federated learning could take about 2 to 3 minutes in practice. To address the communication inefficiency, FedAvg [19] was proposed and acknowledged as the most basic method in federated learning. In FedAvg, the server randomly selects some clients and sends the server model to them. Each client conducts many local training steps using SGD and local data based on the server model and sends back the updated model to the server. The server then averages the models received from clients and finishes one round of training. Therefore, the number of communication rounds is greatly reduced because communication is not conducted every training iteration. Here we also refer to the idea of FedAvg as periodic averaging. When we have the full participation of the clients in each training round, FedAvg reduces to local SGD [26][31]. Another parallel line of works is to compress the communication in federated learning to reduce the volume of the message [23]. But in this paper, we do not consider communication compression.

Although periodic averaging methods such as FedAvg greatly improve the training efficiency in federated learning, a new problem named client drift arises. Because we cannot gather and randomly shuffle the client data as data-center distributed training does, the data distributions of different clients are non-i.i.d. Therefore, the gradients computed at different clients can be highly skewed. Given that we do many local training steps in each training round, skewed gradients will lead to update directions gradually diverging and over-fitting local data in different clients. This client
drift issue can deteriorate the performance of FedAvg drastically [33, 8, 9], especially with a low similarity of the data distribution on different clients and a large number of local training steps.

Many efforts have been made to tackle the critical client drift problem in federated learning, and momentum is one of them. As a method to reduce variance shown in recent work [3] and smooth the model update direction, momentum SGD has shown its power in training various deep learning models in various tasks [27]. [9] proposed to maintain the momentum for the average local model update in a training round and named it as server momentum (or global momentum) SGD. The server momentum SGD method has been proposed in [2] for training speech models and in [29] for distributed training, but [29] did not apply it to federated learning. Vanilla momentum SGD maintains momentum for the gradient in each training step. To distinguish it from the server momentum SGD method, we refer to vanilla momentum SGD as local momentum SGD in federated learning throughout this paper. [9] empirically showed the ability of server momentum SGD to tackle client drift in federated learning, while [29] provided a theoretical analysis of server momentum SGD. However, there has been a lack of understanding in the connection between the server and local momentum SGD in federated learning. Besides, whether we can further improve momentum-based method in federated learning remains another question.

In this paper, we answer the above questions by proposing double momentum SGD (DOMO). we consider cross-silo federated learning [12] scenario in this work, where the number of participating clients is comparatively small, so it is reasonable to require the full participation of all the clients in each training round. But each client may possess a large amount of local data, making it prohibitive to compute the full local gradient. Practical scenarios include collaborative learning of hospitals in health care, financial institutes, etc. In contrast to cross-silo federated learning, cross-device federated learning [12] has an extremely large number of participating clients such that we can only sample a fraction of the clients for training in each round, but each client tends to possess a comparatively small number of local data. Practical scenarios include collaborative learning with mobile devices. We summarize our contributions as follows.

- We propose a new double momentum SGD (DOMO) method with a novel server momentum fusion technique.
- We provide the first theoretical analysis involving both server and local momentum SGD in non-convex settings and new insights into their connection.
- Extensive deep federated learning experiments show that DOMO can improve the test accuracy by up to 5% compared with the state-of-the-art momentum-based method when training VGG-16 on CIFAR-10.

2 Background & Related Work

| Training round (total) | r (R) |
| Local training step (total) | p (P) |
| Client (total) | k (K) |
| Server, local learning rate | α, η |
| Server momentum fusion constant | β |
| Server, local momentum constant | µs, µl |
| Server, local momentum buffer | m_r, m_p |
| Server, (average) local model | x_r, x_r,k |
| Stochastic gradient | ∇F(k)(x_r,p,ξ(k)) |
| Full gradient | ∇f(k)(x_r,p) |

To begin with, consider federated learning as an optimization problem of

\[
\min_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{K} \sum_{k=0}^{K-1} f^{(k)}(\mathbf{x}),
\]

where \( f^{(k)} \) is the local loss function on client \( k \), \( \mathbf{x} \) is the model weights and \( K \) is the number of clients. All the basic notations throughout this paper are listed in Table 1. In FedAvg, the client trains the local model for \( P \) steps using SGD with gradient \( \nabla F^{(k)}(x_r,p,\xi^{(k)}) \) and sends local model update \( x_r^p - x_r^0 \) to server. Server then takes an average and update the server model via \( x_{r+1} = x_r - \frac{\alpha}{K} \sum_{k=0}^{K-1} (x_r^{(k)} - x_r^0) \).
We focus on improving momentum-based methods in federated learning and describe our new double momentum SGD (DOMO) algorithm in this section.

### Algorithm 1: Double Momentum SGD (DOMO)

1. **Input:** period $P \geq 1$, number of rounds $R$, number of clients $K$, server learning rate $\alpha$, local learning rate $\eta$, server momentum constant $\mu_s$, local momentum constant $\mu_l$, server momentum fusion constant $\beta$

2. **Initialize:** Server momentum buffer $m_0 = 0$. $\forall k \in [K]$, local model $x^{(k)}_0 = x_0$ and local momentum buffer $m^{(k)}_{0,0} = 0$.

3. **for** $r = 0, 1, \ldots, R - 1$ **do**
4.  **Client $k$:**
5.   $(r \geq 1)$ Receive $x_r$ and $\frac{1}{P} \sum_{p=0}^{P-1} m^{(k)}_{r-1,p}$ to initialize $x^{(k)}_{r,0}$ and $m^{(k)}_{r,0}$. Receive $m_r$ from server.
6.   **for** $p = 0, 1, \ldots, P - 1$ **do**
7.     **Option I:** $x^{(k)}_{r,p} \leftarrow x^{(k)}_{r-1,p} - \eta \beta P m_r \cdot 1_{p=0}
8.     m^{(k)}_{r,p+1} = \mu_l m^{(k)}_{r,p} + \nabla F(x^{(k)}_{r,p}, \zeta^{(k)}_{r,p})
9.     **Option I:** $x^{(k)}_{r,p+1} = x^{(k)}_{r,p} - \eta m^{(k)}_{r,p+1} - \eta \beta m_r$
10. **Option II:** $x^{(k)}_{r,p+1} = x^{(k)}_{r,p} - \eta m^{(k)}_{r,p+1} - \eta \beta m_r$
11. **end for**
12. Send $d^{(k)}_{r} = \frac{1}{P} \sum_{p=0}^{P-1} m^{(k)}_{r,p+1}$ and $m^{(k)}_{r,p+1}$ to server.
13. **Server:**
14. Receive $d^{(k)}_{r}$ and $m^{(k)}_{r,p}$ from client.
15. $m_{r+1} = \mu_s m_r + \frac{1}{K} \sum_{k=1}^{K} d^{(k)}_{r}$
16. $x_{r+1} = x_r - \alpha \nabla F m_{r+1}$
17. Send $x^{(k)}_{r+1}, \frac{1}{K} \sum_{k=1}^{K} m^{(k)}_{r+1}$, and $m^{(k)}_{r+1}$ to client.
18. **end for**
19. **Output:**

**Momentum-based.** State-of-the-art method server momentum SGD maintains a server momentum buffer with the local model update $\frac{\alpha}{\mu_s} \sum_{k=0}^{K-1} (x^{(k)}_{r,P} - x^{(k)}_{r,0})$ and is used to update the server model. While local momentum SGD maintains a local momentum buffer with $\nabla F^{(k)}(x^{(k)}_{r,P}, \zeta^{(k)}_{r,P})$ and is used to update the local model. [29] empirically showed that the server learning rate $\alpha = 1$ in server momentum SGD.

**Adaptive.** [22] applied the idea of using server statistics as in server momentum SGD to adaptive optimizers including Adam [14], AdaGrad [6], and Yogi [32]. [22] showed that server learning rate should be smaller than $\mathcal{O}(1)$ in terms of complexity, but the exact value was unknown. [22] also showed that convergence analysis with full participation could be easily generalized to partial participation as in cross-device federated learning.

**Inter-client Variance Reduction.** Variance reduction in federated learning refers to correct client drift caused by different data distribution on different clients following the variance reduction convention. In contrast, traditional stochastic methods based on variance reduction [11, 14] and popular in convex optimization can be seen as intra-client variance reduction. Scaffold [13] proposed to maintain a control variate $c_k$ on each client $k$ and add $\frac{1}{K} \sum_{k=0}^{K-1} c_k - c_k$ to gradient $\nabla F^{(k)}(x^{(k)}_{r,P}, \zeta^{(k)}_{r,P})$ when conducting local training. A prior work VRL-SGD [18] was built on a similar idea with $c_k$ equal to the average local gradients in the last training round. Both Scaffold and VRL-SGD have to maintain local statistics and make the clients stateful, so they are more suitable for cross-silo federated learning. Mimeo proposes to apply server statistics locally to address this issue and can be seen as a combination of server statistics and variance reduction. However, Mime has to compute the full local gradient which can be prohibitive in cross-silo federated learning. Besides, Mimo’s theoretical results are based on Storm [2] but their algorithm is based on standard Polyak’s momentum. Though theoretically appealing, variance reduction technique has shown to be ineffective in practical neural networks’ optimization [5], [5] argued that common tricks such as data augmentation, batch normalization, and dropout broke the transformation locking and deviated practice from theory.

**Other.** There are some other settings of federated learning including heterogeneous optimization [17, 28], fairness [20], etc. These different settings, variance reduction techniques, and server statistics can sometimes be combined.

## 3 New Double Momentum SGD (DOMO)

We focus on improving momentum-based methods in federated learning and describe our new double momentum SGD (DOMO) algorithm in this section.
We illustrate the general form of DOMO in Algorithm 1. Different from all existing works, we maintain both the server Aggregate Local Model Updates without Server Momentum.

Algorithm 1: Double Momentum SGD (DOMO, $\beta = \mu_s$).

1: **Input:** period $P \geq 1$, number of rounds $R$, number of clients $K$, server learning rate $\alpha$, local learning rate $\eta$, server and local momentum constant $\mu_s$ and $\mu_l$.

2: **Initialize:** Server momentum buffer $m_0 \equiv 0$. \(\forall k \in [K]\), local model $x_{0,0}^{(k)} = x_0$ and local momentum buffer $m_{0,0}^{(k)} = 0$.

3: for $r = 0, 1, \ldots, R - 1$ do

4: **Client** $k$:

5: \((r \geq 1)\) Receive $x_r$ and \(\frac{1}{K} \sum_{k=0}^{K-1} m_{r-1,0}^{(k)}\) to initialize $x_{r,0}^{(k)}$ and $m_{r,0}^{(k)}$. Receive $m_r$ from server.

6: for $p = 0, 1, \ldots, P - 1$ do

7: Option I: $x_{r,p}^{(k)} \leftarrow x_{r,p}^{(k)} - \eta \mu_s P m_r \cdot 1_{p=0}$

8: $m_{r,p+1}^{(k)} = \mu_l m_{r,p}^{(k)} + \nabla F(x_{r,p}^{(k)}; \theta_{r,p})$

9: Option II: $x_{r,p+1}^{(k)} = x_{r,p}^{(k)} - \eta m_{r,p}^{(k)} - \eta \mu_s m_r$

10: Option III: $x_{r,p+1}^{(k)} = x_{r,p}^{(k)} - \eta m_{r,p+1}^{(k)} - \eta \mu_s m_r \cdot 1_{p=P-1}$ 

11: end for

12: Send $d_{r}^{(k)} = \frac{1}{\eta P} (x_{r,p}^{(k)} - x_{r,p}^{(k)})$ and $m_{r,p}^{(k)}$ to server.

13: **Server**:

14: Receive $d_{r}^{(k)}$ and $m_{r,p}^{(k)}$ from client.

15: $m_{r+1} = \frac{1}{K} \sum_{k=1}^{K} d_{r}^{(k)}$, $x_{r+1} = x_r - \alpha \eta P m_{r+1}$

16: Send $x_{r+1}$, \(\frac{1}{K} \sum_{k=1}^{K} m_{r+1}^{(k)}\), and $m_{r+1}$ to client.

17: end for

18: **Output:**

3. Double Momentum Buffers (DOMO & DOMO-S)

We illustrate the general form of DOMO in Algorithm 1. Different from all existing works, we maintain both the server and local statistics (momentum buffers). However, the local momentum buffer does not make the clients in DOMO stateful. At the end of each training round, we will average the local momentum buffer such that the clients start from the same local momentum buffer in the next training round. In the first place, we briefly summarize the idea of DOMO in the following steps.

1. Receive initial model and statistics from server in the beginning of the training round.
2. Fuse the server momentum buffer in local training steps.
3. Remove the effect of the server momentum buffer in the local model update before sending to the server.
4. Aggregate local model updates from clients to update model and statistics at the server node.

**Server Momentum Fusion in Local Training Steps.** For the corresponding part in Algorithm 1, client $k$ first initialize its local model $x_{r,0}^{(k)}$, local momentum buffer $m_{r,0}^{(k)}$, and server momentum buffer $m_r$ at the start of training round $r$ in Algorithm 1 line 5. In each local training step, apart from standard local momentum SGD (Algorithm 1 line 8), we propose two options to fuse server momentum into local training steps (Algorithm 1 lines 9, 10). For simplicity, we denote option I as DOMO and option II as DOMO-S with “S” standing for “scatter”. In DOMO, we apply server momentum buffer $m_r$ with coefficient $\beta P$ and learning rate $\eta$ to the local model only at the first local training step ($p = 0$). $\beta$ is the server momentum fusion constant. While in DOMO-S, we evenly scatter this process to all the ($P$) local training steps, therefore the coefficient becomes $\beta$ instead of $\beta P$ in Algorithm 1 line 10.

Intuitively, the motivation behind DOMO is that the local model update direction should be adjusted by the direction of server momentum buffer to tackle the client drift issue, especially when the data distribution across clients is highly skewed. Furthermore, DOMO-S follows this motivation in a more fine-grained way and adjusts each local momentum SGD training step by server momentum buffer.

**Aggregate Local Model Updates without Server Momentum.** We propose to remove the effect of server momentum $m_r$ in local model updates (Algorithm 1 line 12) before aggregating it to server. The remaining part (Algorithm 1 lines 14 to 16) in the server follows standard server momentum SGD. Mathematically speaking, the equivalent server momentum constant would have been deviated to $\mu_s + \beta$ if we would not remove it. Besides, the range of $\beta$ would...
have been narrowed down to $[0, 1 - \mu_s)$. Intuitively speaking, momentum buffer serves as a smoothed update direction of the stochastic gradient to reduce variance (or sampling noise). Smoothing itself is not necessary.

### 3.2 Pre-Momentum, Intra-Momentum, & Post-Momentum

To improve the understanding of the connection between server momentum SGD and local momentum SGD, we propose new concepts called pre-momentum, intra-momentum, and post-momentum. We will show that the naive combination of server momentum and local momentum SGD works like post-momentum, while our proposed DOMO and DOMO-S work like pre-momentum and intra-momentum respectively.

To illustrate these concepts, we turn Algorithm 1 into Algorithm 2 as an equivalent form when 

$$m_p = \text{FedAvgLM}.$$  

Consequently, we provided new insight into the connection between server momentum and local momentum SGD by looking at the order of applying server momentum buffer and local momentum buffer. Considering that server and local momentum buffers can be regarded as the smoothed server and local update direction, the order of applying which one first shall not make much difference when the similarity of data distribution across clients is high, i.e., the client drift issue is not severe. However, when the data similarity is low, it becomes more significant to provide the information of server update direction during the local training as pre-momentum and intra-momentum do.

### 4 Convergence of DOMO

In this section, we interpret the motivation behind DOMO from a theoretical perspective. It is the first convergence analysis involving both server and local momentum SGD to the best of our knowledge. There has been little theoretical analysis even for the naive combination of server and local momentum SGD (FedAvgSLM). We consider non-convex smooth objective function satisfying Assumption 1. We also assume that the local stochastic gradient is an unbiased estimation of local full gradient and has a bounded variance in Assumption 2. Furthermore, we bound the non-i.i.d. data distribution across clients in Assumption 3 which is looser than when $B = 0$. We note that for i.i.d. data distribution as in data-center distributed training, $G = 0$ and $B = 0$. For non-i.i.d. data distribution in federated learning, $G$ measures the data similarity in different clients. Specifically, a low data similarity will lead to a larger $G^2$. Note that some basic notations are listed in Table 1.

**Assumption 1** (L-Lipschitz Smoothness) The global objective function $f(\cdot)$ and local objective function $f^{(k)}$ are $L$-smooth, i.e.,

$$\|\nabla f^{(k)}(x) - \nabla f^{(k)}(y)\|_2 \leq L\|x - y\|_2, \quad (2)$$

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^d, k \in [K].$$

**Assumption 2** (Unbiased Gradient and Bounded Variance) The stochastic gradient $\nabla F^{(k)}(x, \xi)$ is an unbiased estimation of the full gradient $\nabla f^{(k)}(x)$, i.e.,

$$\mathbb{E}_\xi \nabla F(x, \xi) = \nabla f(x), \quad \forall x \in \mathbb{R}^d.$$  

Its variance is also bounded, i.e.,

$$\mathbb{E}_\xi \|\nabla F(x, \xi) - \nabla f(x)\|_2^2 \leq \sigma^2, \quad \forall x \in \mathbb{R}^d.$$  

**Assumption 3** (Bounded Non-i.i.d. Distribution) For any client $k \in [K]$ and $x \in \mathbb{R}^d$, there exists $B \geq 0$ and $G \geq 0$, the variance of the local full gradient in each client is upper bounded so that:

$$\frac{1}{K} \sum_{k=0}^{K-1} \|\nabla f^{(k)}(x) - \nabla f(x)\|_2^2 \leq G^2 + B^2\|\nabla f(x)\|_2^2.$$  

(5)
Theorem 1 (Convergence of DOMO) Assume Assumptions 1, 2, and 3 exist. Let \( P \leq \min\{\frac{1 - \mu_s}{6\eta L}, \frac{1 - \mu_s}{6\eta L}\} \) and \( 1 - 2\eta L - \frac{4\mu_s^2\eta^2 L^2}{(1 - \mu_s)^2} \leq 0 \). When \( \alpha = 1 - \mu_s \) and \( \beta = \mu_s \), we have

\[
\frac{1}{RP} \sum_{rP + p = 0}^{RP - 1} E\|\nabla f(x_{r,p})\|^2 \leq \frac{8(1 - \mu_l)(f_s - f(x_0))}{3\eta R P} + \frac{4\eta L \sigma^2}{3(1 - \mu_l)}(1 - \frac{1}{K}) + \frac{3\eta L P}{2(1 - \mu_l)} + \frac{\mu_0^2 \eta L}{(1 - \mu_l)^3 K} + \frac{12\eta^2 L^2 P^2 G^2}{(1 - \mu_l)^2}.
\]

(6)

According to Theorem 1 let \( \eta = O(K^{\frac{1}{2}} R^{-\frac{1}{2}} P^{-\frac{1}{2}}) \) and \( P = O(K^{-\frac{1}{2}} R^{\frac{1}{2}}) \), then we have a convergence rate \( \frac{1}{RP} \sum_{rP + p = 0}^{RP - 1} E\|\nabla f(x_{r,p})\|^2 = O(K^{-\frac{1}{2}} R^{-\frac{1}{2}} P^{-\frac{1}{2}}) \) which achieves a linear speedup regarding the number of clients \( K \) and the same iteration complexity as SGD.

Lemma 1 (DOMO update rule) Suppose \( 0 \leq r' \leq r \) and \( 0 \leq p' \leq p \). Let

\[
\hat{y}_{r,p} = x_0 - \frac{\alpha \eta}{1 - \mu_s} K \sum_{k=0}^{K-1} rP + p' - 1 \sum m_{r', p'} + 1 \text{ and } z_{r,p} = \frac{1}{\mu_l - 1} \hat{y}_{r,p} - \frac{1}{\mu_l - 1} \hat{y}_{r,p-1}
\]

where \( \hat{y}_{0,0} = y_{0,0} = x_0 \), then

\[
z_{r,p+1} - z_{r,p} = -\frac{\alpha \eta}{(1 - \mu_l)(1 - \mu_s)} K \sum_{k=0}^{K-1} rP + p' - 1 \sum m_{r', p'} + 1 \nabla F^{(k)}(x_{r,p}, x_{r', p'})
\]

(7)

Sketch of Proof. The key of the proof is to find a novel auxiliary sequence \( \{z_{r,p}\} \) that not only has a concise update rule than the mixture of server and local momentum, but also is close to the average local model \( \{x_{r,p}\} \). Moreover, \( z_{r,p} \) should equal \( z_{r+1,0} \) to facilitate the analysis of the update between \( z_{r,p-1} \) and \( z_{r+1,0} \). Lemma 1 gives the update rule of such an auxiliary sequence. In contrast, the analysis between \( x_{r,p-1} \) and \( x_{r+1,0} \) is more tricky due to the server momentum buffer applied at the end of the training round. Before to analyze the convergence of \( \{\hat{y}_{r,p}\} \) with the help of \( \{z_{r,p}\} \), we only have to bound \( \|x_{r,p} - \hat{x}_{r,p}\|_2 \) (inconsistency bound) and \( \frac{1}{K} \sum_{k=0}^{K-1} \|x_{r,p} - x_{r,p}^{(k)}\|_2 \) (divergence bound). The divergence bound measures how the local models on different clients diverges due to non-i.i.d. distribution and is more straightforward to analyze since it is only affected by local momentum SGD. The inconsistency bound measures the inconsistency between the auxiliary variable and the average local model as a trade-off for a more concise update rule.

Lemma 2 (Inconsistency Bound) Let \( h_1 = \frac{\mu_1}{1 - \mu_s} - (1 - \frac{\alpha}{1 - \mu_s}) \frac{\mu_1}{1 - \mu_s} \), \( \alpha \geq (1 - \mu_s)(1 - \mu_l) \), and \( \beta = \frac{\mu_s - \mu_1}{\mu_1} \alpha \), we have

\[
\sum_{t=0}^{RP - 1} \|z_{r,p} - \hat{x}_{r,p}\|^2 \leq \frac{\eta^2}{1 - \mu_l} \left( \sum_{p=0}^{P-1} h_1^2 \mu_l \right) \cdot \sum_{t=0}^{RP - 1} \|\nabla F^{(k)}(x_{r', p'}, x_{r', p'})\|^2.
\]

(9)

Tighten Inconsistency Bound. In Lemma 2 we show that server momentum buffer in DOMO can help tighten the inconsistency bound to improve the bound in Eq. (6). Note that server momentum buffer does not affect the divergence bound. Specifically, setting \( \alpha = (1 - \mu_s)(1 - \mu_l) \) and DOMO can scale the inconsistency bound done to about \( \frac{1}{\mu_1 + \mu_1^2} \) of that in local momentum SGD (\( \alpha = 1, \mu_s = 0 \)). It is reasonable considering server momentum buffer carries historical local momentum information. Therefore, we connect the server and local momentum by showing the benefit from the theoretical perspective. The server momentum fusion technique is critical in the proof of Lemma 2. The improvement of the inconsistency bound is a constant factor. It does not affect the overall convergence rate but can accelerate the initial training when the learning rate is large, which is crucial in federated learning. Note that this improvement analysis has not reached the optimal due to inequality scaling.

DOMO with \( \beta = \mu_s \). From Lemma 2 we can see that by setting \( \beta = \mu_s \), DOMO preserves the same inconsistency bound as local momentum SGD. There is good reason to make this choice and turn Algorithm 1 to Algorithm 2. Consider momentum buffer as a smoothed update rule. Suppose the update of server momentum buffer \( m_{r+1} = \mu_s m_r + \Delta_r \) becomes steady with \( \Delta_r \to \Delta \), then \( m_r \) becomes an estimation of \( \frac{\Delta}{1 - \mu_s} \). With the coefficient \( \frac{1}{1 - \mu_s} \), the local momentum SGD is consistent with server momentum SGD in terms of the magnitude of the update. Setting \( \alpha = 1 - \mu_s \) balance the inconsistency and lead to \( \beta = \mu_s \) in Lemma 2. We note that in practice, \( \alpha \) still needs tuning. For reference, (29) showed that the bound in convergence analysis is minimized when \( \alpha = 1 - \mu_s \), but \( \alpha = 1 \) works best in their experiments. We also note that there is no \( \mu_s \) in Theorem 1 because it is removed from Lemma 2 by setting \( \beta = \mu_s \).
Figure 1: Training curves using the VGG-16 model with various data similarity $s$. Best viewed in color.

Figure 2: Test accuracy (%) with various sever momentum constant $\mu_s$ and local momentum constant $\mu_l$. $\mu_s = 0$ corresponds to FedAvgLM, $\mu_l = 0$ corresponds to FedAvgLM, $\mu_s = 0$ & $\mu_l = 0$ corresponds to FedAvg, and $\mu_s \neq 0$ & $\mu_l \neq 0$ corresponds to DOMO. Best viewed in color.

5 Experimental Results

5.1 Settings

All experiments are implemented using PyTorch [21] and run on a cluster where each node is equipped with 4 Tesla P40 GPUs and 64 Intel(R) Xeon(R) CPU E5-2683 v4 cores @ 2.10GHz. We compare the following momentum-based methods: 1) FedAvg, 2) FedAvgSM (i.e., server momentum SGD), 3) FedAvgLM (i.e., local momentum SGD), 4) FedAvgLM-Z (i.e., local momentum SGD with local momentum buffer reset to zero [24] after each training round), 5) FedAvgSLM (i.e., FedAvgSM + FedAvgLM), 6) FedAvgSLM-Z (i.e., FedAvgSM + FedAvgLM-Z), 7) DOMO (i.e., option I), and 8) DOMO-S (i.e., option II). By default, we set $\alpha = 1$ as suggested in [29] and $\beta = \mu_s$ unless specified otherwise. The local momentum constant $\mu_l$ is tuned from $\{0.9, 0.8, 0.6, 0.4, 0.2\}$. We tune the server momentum constant $\mu_s$ from $\{0.9, 0.6, 0.3\}$ which is more coarse-grained because $\mu_s = 0.9$ already works best for all methods in all our experiments. $\mu_s = 0.9$ and $\mu_l = 0.6$ by default unless specified otherwise. The base learning rate is tuned from $\{4 \times 10^{-3}, 2 \times 10^{-1}, 1 \times 10^{-1}, 5 \times 10^{-2}, 1 \times 10^{-2}, 5 \times 10^{-3}, ...\}$. We test local epoch $E \in \{0.5, 1, 2\}$ and $E = 1$ by default.

Data Similarity $s$. We follow the practice in [13] to simulate the non-i.i.d. data distribution. Specifically, fraction $s$ of the data are randomly selected and allocated to clients, while the remaining fraction $1 - s$ are allocated by sorting.
We illustrate the experimental results in Figures 1, 2, 3, and 4 with test accuracy \((\text{worsens the client drift issue.})\) The number of groups in group normalization is set to 8. The local batch size \(b\) \(\leq 32\). For VGG-16, there is no batch normalization \([10]\) layer. For ResNet-56, we replace the batch normalization layer with the group normalization \([30]\) layer because \(\text{non-i.i.d. data distribution causes inaccurate batch statistics estimation and worsens the client drift issue.}\) The number of groups in group normalization is set to 8. The local batch size \(b\) \(= 32\) and the total batch size \(B = Kb = 512.\) In this setting, the number of local training step \(P = 98\) when the local epoch \(E = 1.\) The weight decay is \(5 \times 10^{-4}.\) The model is trained for 200 epochs with a learning rate decay of 0.1 at epoch 120 and 160. Random cropping, random flipping, and standardization are applied as data augmentation techniques.

Dataset. We train the VGG-16 \([25]\) and ResNet-56 \([7]\) models on CIFAR-10 \([16]\) image classification task. For VGG-16, there is no batch normalization \([10]\) layer. For ResNet-56, we replace the batch normalization layer with the group normalization \([30]\) layer because \(\text{non-i.i.d. data distribution causes inaccurate batch statistics estimation and worsens the client drift issue.}\) The number of groups in group normalization is set to 8. The local batch size \(b\) \(= 32\) and the total batch size \(B = Kb = 512.\) In this setting, the number of local training step \(P = 98\) when the local epoch \(E = 1.\) The weight decay is \(5 \times 10^{-4}.\) The model is trained for 200 epochs with a learning rate decay of 0.1 at epoch 120 and 160. Random cropping, random flipping, and standardization are applied as data augmentation techniques.

5.2 Performance

We illustrate the experimental results in Figures 1, 2, 3, and 4 with test accuracy \((\text{mean \pm std})\) reported in the brackets of the legend. Testing performance is the main metric for comparison in federated learning because local training metrics become less meaningful with clients tending to overfit their local data during local training. In overall, DOMO \(\gg\) DOMO-S \(>\) FedAvgSLM-Z \(\geq\) FedAvgSLM \(>\) FedAvgSM \(>\) FedAvgLM-Z \(\geq\) FedAvgLM \(>\) FedAvg regarding the test accuracy. DOMO and DOMO-S consistently achieve the fastest empirical convergence rate and best test accuracy in all experiments. On the contrary, the initial convergence rate of FedAvgSLM and FedAvgSLM-Z can even be worse than FedAvgSM. Besides, using server statistics is much better than without it (FedAvgSM \(\gg\) FedAvg and FedAvgSLM(-Z) \(\gg\) FedAvgLM(-Z)), in consist with the result in 9.

1https://www.cs.toronto.edu/ kriz/cifar.html
We can see that DOMO improves the test accuracy over the best counterpart by varying model, varying the local epoch, varying hyper-parameters, and varying data similarity. Table 2: Test accuracy (%) when training VGG-16 using DOMO with various hyper-parameters.

Reference:

Varying Data Similarity $s$. We plot the training curves under different data similarity settings in Figure 1. We can see that the improvement of DOMO and DOMO-S over other momentum-based methods increases with the data similarity $s$ decreasing. This property makes our proposed method favorable in federated learning where the data heterogeneity can be complicated. In particular, DOMO improves FedAvgSLM-Z, FedAvgSM, and FedAvg by 5.00%, 5.68%, and 23.14% respectively regarding the test accuracy when $s = 5\%$. When $s = 10\%$ and $s = 20\%$, DOMO improves over the best counterpart by 2.13% and 1.02% respectively, while DOMO-S improves by 1.41% and 0.85% respectively.

Varying the Server and Local Momentum Constant $\mu_s$, $\mu_l$. We explore the various combinations of server and local momentum constant $\mu_s$ and $\mu_l$ of DOMO and report the test accuracy in Figure 2. $\mu_s = 0.9$ and $\mu_l = 0.6$ work best regardless of the data similarity $s$ and the algorithm we use. Deviating from $\mu_s = 0.9$ and $\mu_l = 0.6$ leads to gradually lower test accuracy.

Varying the Local Epoch $E$. We plot the training curves of VGG-16 under different local epoch $E$ settings in Figure 3 with data similarity $s = 10\%$. The number of local training steps $P = 49$ and 196 respectively when $E = 0.5$ and 2. We can see that DOMO improves the test accuracy over the best counterpart by 1.25% and 1.70% respectively when $E = 0.5$ and 2.

Varying Hyper-parameters $\alpha$ and $\beta$. We explore the combinations of hyper-parameters $\alpha$ and $\beta$ and report the corresponding test accuracy in Table 2 to verify the default choice of $\alpha = 1.0$ and $\beta = 0.9$ in our experiments.

Varying Model. We also plot the training curves of ResNet-56 in Figure 4 which exhibit a similar pattern. DOMO improves the best counterpart by 1.35% when data similarity $s = 10\%$ and local epoch $E = 1$. In particular, we find that FedAvgSLM and FedAvgSLM-Z are inferior to FedAvgSM which implies that a naive combination of server and local momentum SGD may even hurt the performance. In contrast, DOMO and DOMO-S improve FedAvgSLM by 2.45% and 2.03%.

6 Conclusion

In this work, we have presented a new double momentum SGD (DOMO) method with a novel server momentum fusion technique to improve the state-of-the-art momentum-based federated learning algorithm. We provided new insights into the connection between the server and local momentum with the new concepts of pre-momentum, intra-momentum, and post-momentum in DOMO. We also provided the first convergence analysis involving both the server and local momentum SGD. From a theoretical perspective, we elaborated this connection by showing that server momentum could lead to a tighter inconsistency bound in DOMO. Future works may include incorporating the inter-client variance reduction technique to tighten the divergence bound as well. Deep federated learning experimental results verify the effectiveness of DOMO. DOMO can achieve an improvement of up to 5% regarding the test accuracy compared with the state-of-the-art momentum-based method when training VGG-16 on CIFAR-10.

References

[1] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, et al. Towards federated learning at scale: System design. arXiv preprint arXiv:1902.01046, 2019.

[2] K. Chen and Q. Huo. Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering. In 2016 ieee international conference on acoustics, speech and signal processing (icassp), pages 5880–5884. IEEE, 2016.
[3] A. Cutkosky and F. Orabona. Momentum-based variance reduction in non-convex sgd. In *Advances in Neural Information Processing Systems*, pages 15236–15245, 2019.

[4] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27:1646–1654, 2014.

[5] A. Defazio and L. Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv preprint arXiv:1812.04529*, 2018.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pages 4387–4398. PMLR, 2020.

[9] T.-M. H. Hsu, H. Qi, and M. Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[11] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

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

[15] J. Konećný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[16] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1912.06127*, 2018.

[18] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng. Variance reduced local sgd with lower communication complexity. *arXiv preprint arXiv:1912.12844*, 2019.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[20] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625, 2019.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[22] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[23] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

[24] F. Seide and A. Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2135–2135, 2016.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] S. U. Stich. Local sgd converges fast and communicates little. In *International Conference on Learning Representations*, 2018.

[27] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.
[28] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

[29] J. Wang, V. Tantia, N. Ballas, and M. Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. In *International Conference on Learning Representations*, 2019.

[30] Y. Wu and K. He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[31] H. Yu, R. Jin, and S. Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization. In *International Conference on Machine Learning*, pages 7184–7193, 2019.

[32] M. Zaheer, S. Reddi, D. Sachan, S. Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in neural information processing systems*, pages 9793–9803, 2018.

[33] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.