Hierarchical Federated Learning With Momentum Acceleration in Multi-Tier Networks

Zhengjie Yang®, Student Member, IEEE, Sen Fu®, Wei Bao®, Member, IEEE, Dong Yuan®, Member, IEEE, and Albert Y. Zomaya®, Fellow, IEEE

Abstract—In this article, we propose Hierarchical Federated Learning with Momentum Acceleration (HierMo), a three-tier worker-edge-cloud federated learning algorithm that applies momentum for training acceleration. Momentum is calculated and aggregated in the three tiers. We provide convergence analysis for HierMo, showing a convergence rate of $O(\frac{1}{T})$. In the analysis, we develop a new approach to characterize model aggregation, momentum aggregation, and their interactions. Based on this result, we prove that HierMo achieves a tighter convergence upper bound compared with HierFAVG without momentum. We also propose HierOPT, which optimizes the aggregation periods (worker-edge and edge-cloud aggregation periods) to minimize the loss given a limited training time. By conducting the experiment, we verify that HierMo outperforms existing mainstream benchmarks under a wide range of settings. In addition, HierOPT can achieve a near-optimal performance when we test HierMo under different aggregation periods.

Index Terms—Federated learning, momentum, convergence analysis, edge computing.

I. INTRODUCTION

W ith the advancement of Industry 4.0, Internet of Things (IoT), and Artificial Intelligence, machine learning applications such as image classification [1], automatic driving [2], and automatic speech recognition [3] are rapidly developed. Since the machine learning dataset is distributed in individual users and in many situations they are not willing to share these sensitive raw data, Federated Learning (FL) emerges [4]. It allows workers to participate in the model training without sharing their raw data. Typically, FL is implemented in two tiers, where multiple devices (workers) are distributed and connected to a remote aggregator (usually located in the cloud). A potential issue of the two-tier FL setting is its scalability. The communication overhead between workers and the cloud is proportional to the number of workers, which causes problems when there are a large number of geo-distributed workers connecting to the remote cloud via the public Internet.

With the development of edge computing [5], a more effective solution is adding the edge tier between local workers and the remote cloud to address the scalability issue. Different from the typical two-tier architecture, in the three-tier hierarchical architecture as shown in Fig. 1, workers can first communicate with the edge node for edge-level aggregation, and then the edge nodes communicate with the remote cloud for cloud-level aggregation. Each edge node is closer to the workers and is usually connected with them in the local/edge network, so that the communication cost is much cheaper compared with the two-tier case when the workers directly communicate with the cloud. In Fig. 1, we can see that much of the traffic through the public Internet (left subfigure) is restrained in the local edge networks (right subfigure) due to the existence of the edge nodes. Therefore, the three-tier architecture is a good fit for larger-scale FL, and has attracted attentions from researchers in recent years [6], [7], [8].

Although the three-tier FL can improve the communication efficiency in one training iteration by replacing worker-cloud communication with worker-edge communication, there is also a need to accelerate its convergence performance to reduce the number of iterations. One obstacle in the three-tier FL is that
each edge node can only aggregate the updates of its local workers, and there is a discrepancy among edge nodes. The edge nodes are to be synchronized in the cloud-level aggregation. The two-level aggregation causes delayed synchronization, leading to less training efficiency. Therefore, it is a strong motivation for us to develop a more efficient algorithm to accelerate the convergence, reducing the number of training iterations in the three-tier hierarchical architecture, and finally improve the overall training efficiency (considering both per-iteration cost and the number of iterations).

Moment is proved to be an effective mechanism to accelerate model training. Many studies have demonstrated its advantage in both centralized machine learning environment [9], [10], [11], [12] and two-tier FL environment [13], [14], [15], [16]. Apart from the conventional gradient descent step, the momentum method conducts additional momentum steps [17] to accelerate convergence. In this paper, we propose Hierarchical Federated Learning with Momentum Acceleration (HierMo), which leverages momentums to accelerate three-tier FL. HierMo is operated as follows: (1) In each iteration, each worker locally updates its own model and worker momentum; (2) In every $\tau$ iterations (\(\tau\) is called the worker-edge aggregation\footnote{Each edge node also calculates another momentum for its own usage to further accelerate convergence. See Section III for the detailed algorithm.} period), each edge node receives, averages, and sends back the models and momentum values with its connected workers, (3) In every $\tau \cdot \pi$ iterations (\(\pi\) is called the edge-cloud aggregation period), the cloud receives, averages, and sends back the models and momentum values with edge nodes. The edge nodes will then distribute them to connected workers. The above (1)–(3) steps are repeated for multiple rounds until the loss is sufficiently small.

Theoretically, we prove that HierMo is convergent and has an $\mathcal{O}(1/\tau)$ convergence rate for smooth non-convex problems for a given $T$ iterations. In this step, we need to address substantial new challenges, compared with two-tier FL. In particular, we develop a new method to characterize the multi-time cross-two-tier momentum interaction and cross-three-tier momentum interaction, which do not exist in the two-tier FL. After we theoretically prove the convergence, we observe that the worker-edge and edge-cloud aggregation periods $\tau$ and $\pi$ are key design variables we aim to optimize. Based on the result of the convergence analysis, we propose HierOPT algorithm, which can find a local optimal ($\tau, \pi$) value pair.

In the experiment, we demonstrate the performance of HierMo compared with various mainstream hierarchical FL and momentum-based FL algorithms, including hierarchical FL without momentum (HierFAVG [18] and CFL [19]), two-tier FL with momentum (FedMom [20], SlowMo [21], FedNAG [22], Mime [23], FastSlowMo [24], DOMO [25], and FedADC [26]), and two-tier FL without momentum (FedAvg [4]). The experiment is implemented on different kinds of models (linear regression, logistic regress, CNN [27], VGG16 [28], and ResNet18 [29]) based on various real-world datasets (MNIST [30], CIFAR-10 [31], ImageNet [29], [32] for image classification, and UCI-HAR [33] for human activity recognition). The experimental results illustrate that HierMo drastically outperforms benchmarks under a wide range of settings. We also verify HierOPT can output a near-optimal ($\tau, \pi$) in the real-world settings. All these results match our expectations by the theoretical analysis.

The contributions of this paper are summarized as follows.

- We have proved that HierMo is convergent and has an $\mathcal{O}(1/\tau)$ convergence rate for smooth non-convex problems for a given $T$ iterations under non-i.i.d. data.
- We have proved that as long as learning step size $\eta$ is sufficiently small, HierMo (with momentum acceleration) achieves the tighter convergence upper bound than HierFAVG (without momentum acceleration).
- We have proposed the new HierOPT algorithm which can find a local optimal pair of ($\tau^*, \pi^*$) when total training time is constrained.
- HierMo is efficient and decreases the total training time by 5%–73% compared with the mainstream two-tier momentum-based algorithms and three-tier algorithms.
- HierOPT generates the near-optimal pair of ($\tau^*, \pi^*$) when the total training time is constrained. HierOPT achieves the near-optimal accuracy with only 0.01–0.07% (CNN on MNIST) and 0.24–0.41% (CNN on CIFAR10) gap from the real-world optimum.

The rest of the paper is organized as follows. In Section II, we introduce related works. The HierMo algorithm design is described in Section III. In Section IV, we provide theoretical results including the convergence analysis of HierMo and the performance gain of momentum. The algorithm to optimize the aggregation periods, i.e., HierOPT, is proposed in Section V. Section VI provides our experimental results and the conclusion is made in Section VII.

II. RELATED WORK

A. Momentum in Machine Learning and Federated Learning

Momentum [34] is a method that enhances the speed of gradient descent in the relevant direction by incorporating a fraction, denoted as $\gamma$, of the difference between past and current model vectors. In the classical centralized setting, the update rule for momentum (Polyak’s momentum) is defined as follows:

$$m(t) = \gamma m(t-1) - \eta \nabla F(w(t-1)), \quad (1)$$
$$w(t) = w(t-1) + m(t), \quad (2)$$

with $\gamma \in [0, 1], t = 1, 2, \ldots$, $m(0) = 0$, where $\gamma$ represents the momentum factor (weight of momentum), $t$ denotes the update iteration, $m(t)$ is the momentum term at iteration $t$, and $w(t)$ represents the model parameter at iteration $t$. By utilizing this method, the momentum term increases for dimensions in which the gradients point consistently in the same direction, while it decreases the updates for dimensions in which the gradients change direction. Consequently, momentum facilitates faster convergence and reduces oscillation [17], [35].

Momentum has been explored in both centralized machine learning and FL. In the centralized environment, another variant of momentum known as Nesterov Accelerated Gradient
(NAG) [17], [36] is proposed. NAG\(^2\) calculates the gradient based on an approximation of the next parameter position, given by \(\nabla F(w(t - 1) + \gamma m(t - 1))\) instead of \(\nabla F(w(t - 1))\) in Polyak’s momentum. This modification leads to improved convergence performance. The utilization of momentum in over-parameterized models is studied in [11], while [9] provides a unified convergence analysis for both Polyak’s momentum and NAG. Additionally, [12] investigates NAG in stochastic settings.

All of the aforementioned studies highlight the benefits of momentum in accelerating centralized training, which has prompted researchers to explore its application in the FL setting. Depending on where the momentum is incorporated, it can be categorized into three types: worker momentum, aggregator momentum, and combination momentum. In the case of worker momentum (e.g., FedNAG [22] and Mime [23]), momentum acceleration is applied at the workers during each local iteration. However, this approach is susceptible to data heterogeneity among workers, which can negatively impact long-term performance. On the other hand, aggregator momentum (e.g., FedMom [20] and SlowMo [21]) applies momentum acceleration only at the aggregator, based on the global model. It exhibits similar acceleration properties as in the centralized setting and helps dampen oscillations [17]. Nevertheless, this method is applied less frequently (every \(\tau\) iterations where \(\tau\) represents the aggregation period), compared to worker momentum (every iteration). Consequently, the performance gain may not be as pronounced, particularly when \(\tau\) is large. To address the aforementioned limitations, recent approaches such as FastSlowMo [24], DOMO [25], and FedADC [26] combine worker and aggregator momenta, demonstrating improved convergence performance compared to using either worker or aggregator momentum alone. Please note that the forms of momentum discussed above are classified according to an approximation of the next parameter position, giving rise to multi-time momentum interaction and cross-three-tier momentum interaction. This is completely different from existing two-tier and three-tier scenarios. We devise new mathematical methods to bound the on-tier momentum acceleration, and two-level virtual update (edge and cloud) method to bound the momentum interactions, so that the convergence of HierMo still holds.

### III. HierMo Problem Formulation

#### A. Overview

We consider a three-tier hierarchical FL system, which comprises a cloud server, \(L\) edge nodes, and \(N\) workers. Each edge node \(\ell\) serves \(C_\ell\) workers, resulting in a total of \(N = \sum_{\ell=1}^{L} C_\ell\) workers across all edge nodes. We denote the \(i\)th worker served by edge node \(\ell\) as worker \((i, \ell)\), where \(i = 1, 2, \ldots, C_\ell, \ell = 1, 2, \ldots, L\). Each worker contains its own local dataset, with the number of data samples denoted by \(D_i, \ell\). The total training dataset in the cluster of workers served by edge node \(\ell\) is \(D_\ell = \sum_{i=1}^{C_\ell} D_i, \ell\), and the total training dataset \(D = \sum_{\ell=1}^{L} D_\ell\). The objective of the three-tier hierarchical FL is to find the stationary point \(w^*\) that minimizes the global loss function \(F(w)\) which is the weighted average of the loss functions of all workers. The problem can be formulated as follows:

\[
\min_{w \in \mathbb{R}^d} F(w) = \sum_{\ell=1}^{L} \sum_{i=1}^{C_\ell} F_i, \ell(w) = \frac{1}{D} \sum_{\ell=1}^{L} \sum_{i=1}^{C_\ell} D_i, \ell F_i, \ell(w)
\]

where \(d\) is the dimension of \(w\), \(F(w)\) is the global loss function at the cloud server, and \(F_i, \ell(w)\) is the local loss function at worker \((i, \ell)\). (4) is the mathematical transformation from (3) by adding \(D_\ell\). We also define the edge loss function at edge node \(\ell\) as \(F_\ell(w) = \sum_{i=1}^{C_\ell} D_i, \ell F_i, \ell(w)\), which is the weighted average of edge node \(\ell\)’s connected workers’ local loss functions \(F_i, \ell(w)\). Therefore, by replacing \(\sum_{i=1}^{C_\ell} D_i, \ell F_i, \ell(w)\) with \(F_\ell(w)\) in (4), we can directly derive (5), demonstrating that the global loss function is the weighted average of all edge loss functions as \(F(w) = \sum_{\ell=1}^{L} \frac{1}{D_\ell} F_\ell(w)\). We assume the problem is within the scope of cross-silo federated learning [40] where all workers are required to participate in the training with siloed data. Each worker represents a repository of data, and data are sensitive and non-i.i.d. The key notations are summarized in Table I.

#### B. Worker Momentum and Edge Momentum

We notice that there are two types of momentum in two-tier FL: One type (i.e., worker momentum) is calculated at each worker and is aggregated; The other type (i.e., aggregator momentum) is calculated at the aggregator. Since both types can accelerate the convergence, we adopt both of them in our work.
In the three-tier case in our paper, the worker momentum is individually computed in each worker and aggregated in the edge node (worker momentum edge aggregation) and the cloud (worker momentum cloud aggregation). We still call it worker momentum throughout the paper. For the aggregator momentum, we apply it at each edge node. Each edge node computes its own momentum and it is not shared with the workers or the cloud. We call it edge momentum throughout this paper.

C. HierMo Algorithm

In Algorithm 1, we propose a momentum-based three-tier hierarchical FL algorithm, named as HierMo, which applies both worker momentum and edge momentum. HierMo aims to find the final cloud model \( \mathbf{w}^T \) to solve the formula (3). It conducts \( T \) local iterations, \( K \) edge aggregations, and \( P \) cloud aggregations, where \( T = K\tau = P\tau\pi, \tau \) is the worker-edge aggregation period, and \( \pi \) is the edge-cloud aggregation period.

1) Worker Update: In each local iteration \( t \), each worker \((i,\ell)\) computes its worker update, which includes two things: (1) worker momentum update \( m^t_{i,\ell} \) (Line 5) and (2) worker model update \( \mathbf{w}^t_{i,\ell} \) (Line 6). ① and ② follow the Nesterov Accelerated Gradient (NAG) [36] momentum update and are conducted every iteration. Through this way, each worker can utilize its own worker momentum acceleration.

2) Edge Update: When \( t = k\tau, k = 1,2,\ldots,K \), each edge node \( \ell \) receives workers’ momenta and models in \( C_\ell \) and performs edge update, which includes two operations: (1) Worker momentum edge aggregation \( m^{k\tau}_{\ell} \) (Line 9) with re-distribution (Line 12). Through this way, some straggler workers with high data-heterogeneity whose local momenta \( m^t_{i,\ell} \) pointing to an inappropriate direction can be refined from \( m^{k\tau}_{\ell} \). (2) Edge momentum \( m^{k\tau}_{\ell} \) and model \( \mathbf{w}^{k\tau}_{\ell} \) update (Lines 10–11) with model re-distribution (Line 13). Since the computation of edge momentum and model update is based on the edge model, it is equivalent to perform it in edge setting involving all workers’ dataset under edge node \( \ell \) (\( D_\ell = \sum_{i=1}^{C_\ell} D_{i,\ell} \)). By doing so, it damps oscillations [17] within the edge node. Please note that ① and ② are two operations on the same edge node, so that we use subscript “−” and “+” to label the momentum/model right after operations ① and ② respectively. Finally, both ① and ② are conducted in each edge node every \( \tau \) iterations.

3) Cloud Update: When \( t = p\tau\pi, p = 1,2,\ldots,P \), the cloud receives edge aggregated worker momentum \( m^{p\tau\pi}_{\ell} \) and...
edge model $w_i^{(p+1)\pi}$ for all $\ell \in L$ and performs cloud update, which includes two things: (1) Worker momentum cloud aggregation $m_i^{(p+1)\pi}$ (Line 16) and re-distribution (Lines 18 and 20). Through this way, all edge nodes and workers receive the cloud aggregated worker momentum and mitigate the disadvantage caused by non-i.i.d. data heterogeneity. (2) Edge model cloud aggregation $w_i^{(p+1)\pi}$ (Line 17) and cloud model re-distribution (Lines 19 and 21). Please note that the cloud will re-distribute the momentum and model to all edge nodes and all edge nodes will then distribute them to all workers when $t$ is a multiple of $\tau\pi$.

D. Extension to Asynchronized FL

In HierMo, the settings of worker-edge aggregation period $\tau$ and edge-cloud aggregation period $\pi$ are the same among workers and edge nodes, which is under synchronized FL scenario. It is also possible to extend HierMo to an asynchronized scenario, where different $\tau$ and $\pi$ values can be set for workers and edge nodes respectively. To ensure convergence under asynchronous settings, a possible solution is to bound the worst-case values of $\tau$ and $\pi$ (the largest $\tau$ and $\pi$ in the synchronous case). There are no existing works proving that the workers with different $\tau$ can perform faster convergence (compared with the worst case $\tau$). In the theoretical aspects, using smaller $\tau$ for some workers but larger $\tau$ for other workers will not generate any acceleration in convergence (tighter bound). The same applies to edge nodes with different $\pi$ values. Therefore, in the following section, we analyze the convergence of synchronous case (same $\tau$ and $\pi$), which is also the common focus in the literature [4], [20], [41].

IV. CONVERGENCE ANALYSIS OF HIERMO

In this section, we present the theoretical analysis of HierMo. We first provide preliminaries. Then, we introduce the concept of virtual update which is a significant intermediate step to conduct convergence analysis. Afterward, we show the convergence guarantee of HierMo. Finally, we compare the convergence upper bound of HierMo and HierFAVG to analyze the performance gain of momentum.

A. Preliminaries

We assume $F_i,\ell(\cdot)$ satisfies the following standard conditions that are commonly adopted in the literature [13], [22], [41].

Assumption 1: $F_i,\ell(w)$ is $\rho$-Lipschitz, i.e., $\|F_i,\ell(w_1) - F_i,\ell(w_2)\| \leq \rho \|w_1 - w_2\|$ for any $w_1, w_2, i, \ell$.

Assumption 2: $F_i,\ell(w)$ is $\beta$-smooth, i.e., $\|\nabla F_i,\ell(w_1) - \nabla F_i,\ell(w_2)\| \leq \beta \|w_1 - w_2\|$ for any $w_1, w_2, i, \ell$.

Assumption 3: (Bounded diversity) The variance of local gradient to edge gradient is bounded. i.e., $\|\nabla F_i,\ell(w) - \nabla F_i(w)\| \leq \delta_i,\ell$ for all $i, \ell, \forall w$. We also define $\delta_i$ as the weighted average of $\delta_i,\ell$ and $\delta$ as the weighted average of $\delta_i$, i.e., $\delta_i = \sum_{\ell \in C_i} \frac{D_i,\ell}{\sum_{\ell' \in C_i} \rho_i,\ell'} \delta_i,\ell$ and $\delta = \sum_{i \in L} D_i \delta_i,\ell$.

According to Assumptions 1 and 2, and applying the Triangle Inequality to $F_i,\ell(w)$, it is straightforward to show that $F_i,\ell(w)$ is $\rho$-Lipschitz and $\beta$-smooth. Assumptions 1 and 2 indicate that the function and the gradient of the function are not changing too fast. Assumption 3 indicates that the data distributed to all workers are heterogeneous and non-i.i.d.. $\delta_i,\ell$ is used to quantify the level of gradient divergence and is different at different workers.

B. Virtual Update

In order to index the edge aggregation and cloud aggregation, we divide the total $T$ local iterations into $K$ edge intervals and $P$ cloud intervals. $T = K\tau = PT\pi$. We use $[k]$ to denote the edge interval $t \in [(k-1)\tau, k\tau]$ for $k = 1, 2, \ldots, K$, and $\{p\}$ to denote the cloud interval $t \in [(p-1)\tau\pi, p\tau\pi]$ for $p = 1, 2, \ldots, P$. Please note that the edge aggregation occurs at the end of each edge interval and the cloud aggregation occurs at the end of each cloud interval. Therefore, each edge interval $[k]$ contains $\tau$ local iterations with one edge aggregation, and each cloud interval $\{p\}$ contains $\pi$ edge intervals with one cloud aggregation, i.e., $\{p\} = \{k\}$ for $k = (p-1)\pi + 1, (p-1)\pi + 2, \ldots, p\pi$.

At the beginning of edge interval $[k]$ when $t = (k-1)\tau$, we set edge virtual update

$$m_i^{(k-1)\tau} \leftarrow m_i^{(k-1)\tau},$$

(6)

$$w_i^{(k-1)\tau} \leftarrow w_i^{(k-1)\tau},$$

(7)

for each edge node $\ell$, where $m_i^{(k-1)\tau}$ and $w_i^{(k-1)\tau}$ are set as the virtual aggregated values right after the edge aggregation occurs. Then, we further conduct edge virtual update as if model and momentum updates are conducted in the edge node. When $t \in ((k-1)\tau, k\tau)$, we conduct edge virtual update as

$$m_i^{(k-1)\tau} \leftarrow m_i^{(k-1)\tau} - \eta \nabla F_i(w_i^{(k-1)\tau}),$$

(8)

$$w_i^{(k-1)\tau} \leftarrow w_i^{(k-1)\tau} + \gamma (m_i^{(k-1)\tau} - m_i^{(k-1)\tau}).$$

(9)

We repeat (6)–(9) for each edge interval $[k]$ where $k = 1, 2, \ldots, K$. Please note that only if $t = k\tau$, $k = 1, \ldots, K$, $m_i^{(k-1)\tau}$ and $w_i^{(k-1)\tau}$ are computed. For ease of analysis, we define intermediate value $w_i^{(t)} = \sum_{i=1}^{C_i} \frac{D_i,\ell}{\sum_{\ell' \in C_i} \rho_i,\ell'} w_i^{(t)},$ and $m_i^{(t)} = \sum_{i=1}^{C_i} \frac{D_i,\ell}{\sum_{\ell' \in C_i} \rho_i,\ell'} m_i^{(t)}$ that are meaningful at any iteration $t$.

Same as edge intervals, for each cloud interval $\{p\}$ where $p = 1, 2, \ldots, P$, the cloud virtual update is also conducted:

$$m_i^{(p-1)\pi} \leftarrow m_i^{(p-1)\pi},$$

(10)

$$w_i^{(p-1)\pi} \leftarrow w_i^{(p-1)\pi},$$

(11)

when $t = (p-1)\tau\pi$, and

$$m_i^{(p-1)\pi} \leftarrow m_i^{(p-1)\pi} - \eta \nabla F(w_i^{(p-1)\pi}),$$

(12)

$$w_i^{(p-1)\pi} \leftarrow w_i^{(p-1)\pi} + \gamma (m_i^{(p-1)\pi} - m_i^{(p-1)\pi}),$$

(13)

when $t \in ((p-1)\tau\pi, p\tau\pi)$.

By applying virtual updates on edge nodes and the cloud, we can bound the gap between real updates and these virtual updates that can be used to prove the convergence. Since in HierMo, the momenta and the models are aggregated on both edge nodes and the cloud, it brings much more challenges to conduct convergence analysis. The virtual update is an important
intermediate process for convergence analysis and is one of our contributions in this paper.

Fig. 2 illustrates the evolution of \(w^t_{[\cdot]_i}, \ w^t_{[\cdot]_e}, \ w^t_{[k]_i}, \ w^t_{[\cdot]_i}, \) and \(w^t\) when \(\tau = 2, \pi = 2\). There are 2 edge nodes and each edge node serves 2 workers (in total 4 workers in Fig. 2). After every 2 local updates, there is an edge aggregation, and after every 2 edge aggregations (4 local updates), there is a cloud aggregation. Please note (1) \(w^t_{[\cdot]_i} \) and \(w^t_{[\cdot]_e}\) are different. \(w^t_{[k]_i}\) is calculated from \(w^{(k-1)}_{[k]_i}\) after \(\tau\) edge virtual updates, while \(w^t_{[\cdot]_i}\) is directly given by \(w^t_{[\cdot]_e}\). (2) \(w^t_{[k]_i}\) and \(w^t_{[\cdot]_i}\) are different. \(w^t_{[k]_i}\) is the intermediate value that is used for edge model/momentum update, while \(w^t_{[\cdot]_i}\) is calculated from \(w^t_{[k]_i}\) during edge model/momentum update. (3) \(w^t_{[\cdot]_i}\) and \(w^t_{[\cdot]_i}\) are different. \(w^t_{[\cdot]_i}\) is calculated from \(w^t_{[\cdot]_i}\) after \(\tau \cdot \pi\) cloud virtual updates, while \(w^t_{[\cdot]_i}\) is directly given by \(w^t_{[\cdot]_i}\).

C. Convergence Analysis

In this section, we provide the convergence analysis of HiErMo. In Theorem 1, we first focus on worker models under each edge node \(i\) to bound the distance between edge intermediate value \(w_{\ell}^t\) and edge virtual update \(w_{[\cdot]_i}^t\) within interval \([k]\).

**Theorem 1:** For any edge interval \([k]\), \(\forall \ell \in ((k-1)\tau, k\tau]\) and \(\forall i \in L\), we have\
\[
\|w_{\ell_i}^t - w_{[\cdot]_i}^t\| \leq h(t - (k-1)\tau, \delta_i),
\]
where \(h(x, \delta_i)\) is\
\[
h(x, \delta_i) = \eta \delta_i \left( I(\gamma A)^x + J(\gamma B)^x - \frac{1}{\eta \beta} \right) - \frac{\gamma^2 (\gamma^t - 1) - (\gamma - 1) x}{(1 - \gamma^t)^2},
\]
and \(A, B, I, J\) are constants defined in Appendix A, available online, for the complete proof.

Please note that when \(t = (k-1)\tau\) for all \([k]\), we have \(\|w_{\ell_i}^t - w_{[\cdot]_i}^t\| = 0 = h(0, \delta_i)\), which also satisfies (15). Also, \(F_i(w)\) is \(\rho\)-Lipschitz, so that we also have\
\[
F_i(w_{\ell_i}) - F_i(w_{[\cdot]_i}) \leq \rho h(t - (k-1)\tau, \delta_i).
\]

**Proof Sketch:** We first obtain the worker momentum upper bound \(\|m_{[\cdot]_i}^t - m_{[\cdot]_i}^t\|\) for each worker \(i, \ell\). Based on it and worker momentum update rules in Lines 5–6 in Algorithm 1, we bound the worker model parameter gap \(\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\|\). Then, we extend above two bounds to obtain edge aggregated worker momentum upper bound \(\|m_{[\cdot]_i}^t - m_{[\cdot]_i}^t\|\). Finally, the gap of edge model parameter \(\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\|\) is obtained. See Appendix A, available online, for the complete proof.

In Theorem 2, we then bound the edge momentum update between \(w_{[\cdot]_i}^t\) and \(w_{[\cdot]_i}^t\) within interval \([k]\).

**Theorem 2:** For any edge interval \([k]\) in any edge node \(i \in L\), suppose \(0 < \gamma < 1, 0 < \gamma_i < 1\), and any \(\tau = 1, 2, \ldots\), we have\
\[
\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\| \leq s(\tau),
\]
where \(s(\tau)\) is\
\[
s(\tau) = \gamma \tau \eta \rho (\gamma \mu + \gamma + 1)
\]
and constant \(\mu\) is defined in Appendix E, available online.

**Proof Sketch:** Based on the edge momentum update rules in Lines 10–11 in Algorithm 1, we can derive \(w_{[\cdot]_i}^t - w_{[\cdot]_i}^t = \gamma (w_{[\cdot]_i}^t - w_{[\cdot]_i}^{(k-1)})\). Then we prove the bound of \(\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\|\) based on the definition of intermediate value where \(w_{[\cdot]_i}^t = \sum_{t=1}^{k\pi} D_t \ell_{t_i}^t w_{\ell_i}^t\), and then the result is obtained. See Appendix E, available online, for the complete proof.

By combining the results of Theorems 1 and 2, we can telescope the bound within edge interval \([k]\) to the cloud interval \([p]\) where \(k = (p-1)\pi + 1, (p-1)\pi + 2, \ldots, p\pi\). Then, we are ready to bound the gap between weighted average of edge virtual update \(\sum_{\ell=1}^{L} D_{\ell} w_{[\cdot]_i}^t\) and cloud virtual update \(w_{[\cdot]_i}^t\) in Theorem 3.

**Theorem 3:** For any cloud interval \([p]\), \(0 < \gamma < 1, 0 < \gamma_i < 1\), and any \(\tau = 1, 2, \ldots\), we have\
\[
\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\| \leq h(\tau, \delta) + \sum_{\ell=1}^{L} D_{\ell} \beta (h(\tau, \delta) + s(\tau)),
\]
where we define \(w_{[\cdot]_i}^t = \sum_{\ell=1}^{L} D_{\ell} w_{[\cdot]_i}^t\), \(\forall \ell \in L\).

**Proof Sketch:** We propose an intermediate sequence of edge virtual update on the cloud \(w_{[\cdot]_i}^t\). Then we bound \(\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\|\) and \(\|w_{[\cdot]_i}^t - w_{[\cdot]_i}^t\|\) respectively to obtain the final result. See Appendix F, available online, for the complete proof.

**Theorem 4:** Under the following conditions: (1) \(0 < \beta \eta (\gamma + 1) \leq 1, 0 < \gamma < 1, 0 < \gamma_i < 1,\) and \(\forall \tau, \pi \in \{1, 2, \ldots\}\); (2) \(\exists \varepsilon > 0, (2.1) \omega \alpha \sigma^2 - \beta(\pi,\tau,\delta_i) > 0; (2.2) F(w_{[\cdot]_i}^t) - F(w^*) \geq \varepsilon, \forall p; (2.3) F(w^t) - F(w^*) \geq \varepsilon\) are satisfied;
Algorithm 1 gives
\[
F(w^T) - F(w^*) \leq \frac{1}{T} \frac{1}{\omega \sigma^2 - \rho j(\tau, \pi, \delta, \ell; \delta)}.
\]  
(20)

where \( j(\tau, \pi, \delta, \ell; \delta) \) is
\[
j(\tau, \pi, \delta, \ell; \delta) = h(\tau, \pi, \delta) + (\pi + 1) \sum_{\ell=1}^{L} \frac{D_{\ell}}{D} \left( \left( h(\tau, \delta) + s(\tau) \right) \right).
\]  
(21)

We define \( F(w^*) \) as the minimum value, if there exists some \( \varphi > 0 \) such that \( F(w^*) \leq F(w) \) for all \( w \) within distance \( \varphi \) of \( w^* \). Constant \( \mu \) is defined in Appendix E, available online, and constants \( \omega, \sigma, \) and \( \alpha \) are defined in Appendix H, available online.

**Proof Sketch:** We first analyze the convergence of \( F(w^{t+1}) - F(w^t) \) within cloud interval \( \{p\} \) when \( t \in [p - 1; p, p + 1] \). Then, we merge \( h(\tau, \delta_i), s(\tau) \), and result of Theorem 3 to handle the overall effects and telescope the gap of overall effects to all \( P \) cloud intervals, and then the final result is obtained. See Appendix H, available online, for the complete proof.

Please note in the proof of Theorems 1 and 2, we have characterized the multi-time tier-one and tier-two momentum acceleration respectively. In the proof of Theorems 2, 3, and 4, we have characterized the multi-time cross-tier momentum interaction and cross-three-tier momentum interaction brought by the three-tier FL. To analyze \( \pi \) times cross-two-tier momentum interactions, we devise a new telescope form to bound these new deviations (49)–(51) and (58). To analyze cross-three-tier momentum interaction, we devise a new mechanism to analyze such momentum interactions across multi-tiers (52)–(58) and (64)–(67).

We have demonstrated that the gap between the global loss function value \( F(w^T) \) and the stationary point \( F(w^*) \) is upper bounded by a function of \( T = K_{\tau} = P_{\pi} \) which is inversely proportional to \( T \). It converges with the convergence rate \( O(\frac{1}{T}) \) for smooth non-convex problems under non-i.i.d. data distribution. We also give the following observations based on the above theorems.

**Observation 1:** The overall gap in Theorem 4, \( F(w^T) - F(w^*) \) decreases when \( T \) is larger. From Appendix G, available online, we have \( h(x) \geq 0 \) for any \( x = 1, 2, \ldots \), and it increases with \( x \). According to (18), \( s(\pi) \) increases with \( \pi \). According to (21), \( j(\tau, \pi) \) increases with \( \tau \) and \( \pi \). Therefore, the value of \( \rho j(\tau, \pi, \delta, \ell; \delta) \) increases with \( \tau \) and \( \pi \) so as to increase the overall bound \( F(w^T) - F(w^*) \). However, in order to let the Condition (2.1) in Theorem 4 hold, we cannot set a very large \( \tau \) and \( \pi \), implying that convergence is guaranteed when \( j(\tau, \pi) \) is below a certain threshold. Experiments on the effects of \( \tau \) and \( \pi \) further verify that larger \( \tau \) and \( \pi \) decreases the convergence performance.

In Theorem 5, we further eliminate the value \( \varepsilon \) in Theorem 4 and further demonstrate the bound between the final loss function value that the algorithm can obtain \( F(w^f) \) and the stationary point \( F(w^*) \), where we define
\[
w^f = \arg \min_{w \in \{w_{\alpha}, \alpha; p = 1, 2, \ldots, P\}} F(w).
\]  
(22)

**Theorem 5:** Under the following condition: \( 0 < \beta \eta(\gamma + 1) \leq 1, 0 < \gamma < 1, 0 < \gamma_e < 1, \) and \( \forall \tau, \pi \in \{1, 2, \ldots\} \), we have
\[
F(w^f) - F(w^*) \leq \frac{1}{2T\omega \alpha^2} + \frac{\rho j(\tau, \pi, \delta, \ell; \delta)}{\omega \alpha^2 \tau \pi} \leq f_{\text{HierMo}}(T).
\]  
(23)

**Proof:** See Appendix I, available online, for the complete proof.

Theorem 5 will be used in Sections IV-D and V to compare the convergence upper bound and formulate the optimization problem respectively.

**D. Comparison Between HierMo and HierFAVG**

In this section, we theoretically quantify the performance gain brought by HierMo compared with HierFAVG (without momentum). The convergence upper bound of HierFAVG can be derived from [18] as follows:
\[
F(w^f) - F(w^*) \leq \frac{1}{2T\omega \alpha^2} + \frac{\rho j^*(\tau, \pi, \delta, \ell; \delta)}{\omega \alpha^2 \tau \pi} \leq f_{\text{HierFAVG}}(T).
\]  
(24)

The definitions of \( \hat{\alpha} \) and \( \hat{j}(\cdot) \) can be found in [18].

To prevent the gradient descent from overshooting [42], it is common to choose a very small \( \eta \). The following theorem is made when \( \eta \to 0^+ \).

**Theorem 6:** When \( 0 < \beta \eta(\gamma + 1) \leq 1, 0 < \gamma < 1, 0 < \gamma_e < 1, \) and \( \forall \tau, \pi \in \{1, 2, \ldots\} \), HierMo outperforms HierFAVG, i.e.,
\[
f_{\text{HierFAVG}}(T) - f_{\text{HierMo}}(T) > 0
\]
for any \( T \) and \( \eta \to 0^+ \).

**Proof:** See Appendix J, available online, for detailed proof.

The above theorem indicates that HierMo leads to a tighter convergence upper bound compared with HierFAVG, showing that HierMo theoretically outperforms HierFAVG.

V. AGGREGATION PERIOD OPTIMIZATION BY HierOPT

We have proved that HierMo is convergent in Section IV. We observe that the worker-edge and edge-cloud aggregation periods \( \tau \) and \( \pi \) are two key design variables that will influence the convergence performance. The values of \( \tau \) and \( \pi \) will also influence the usage of communication and computation resources in the real-world training process. Therefore, we aim to optimize these two variables and formulate an optimization problem: Under a given total training time denoted as \( T_{\text{total}} \), how the HierMo algorithm achieves the best performance (min global model loss).

We denote the worker computation delay for one iteration as \( T_{\text{comp}}^w \), edge computation delay for one edge aggregation as
\(T_{\text{comp}}\), and cloud computation delay for one cloud aggregation as \(T_{\text{comp}}\). We also denote the worker communication delay to the edge as \(T_{\text{w2e}}\) and edge communication delay to the cloud as \(T_{\text{e2c}}\). All the above values are assumed to be given as they can be measured in the real world. We assume each worker \(\{i, \ell\}\) communicates with connected edge node \(\ell\) in parallel and each edge node \(\ell\) communicates with cloud in parallel [8], [18], [43]. The above assumptions are commonly adopted in the literature [41], [43]. As a result, the total training time for HierMo is calculated as follows

\[
T_{\text{total}} \triangleq P \cdot (\tau T_{\text{w}} + \pi T_{\text{comp}} + T_{\text{comp}} + \pi T_{\text{w2e}} + T_{\text{e2c}}),
\]

(25)

where \(P\) is the total number of cloud aggregations (\(P = \frac{L}{T}\)).

In order to find the optimal pair of \((\tau, \pi)\), we target to minimize (23), where (23) demonstrates the bound between the global loss and the stationary point [18], [41]. By incorporating the constraints, the optimization problem can be formulated as follows

\[
\min_{\tau, \pi} \frac{1}{2T\omega^2\sigma^2} \rho_j(\tau, \pi, \delta, \ell, \delta) + \sqrt{\frac{1}{4T^2\omega^2\alpha^2\sigma^4}} + \rho_j(\tau, \pi, \delta, \ell, \delta),
\]

\[
\text{subject to } P \cdot (\tau T_{\text{w}} + \pi T_{\text{comp}} + T_{\text{comp}} + \pi T_{\text{w2e}} + T_{\text{e2c}}) = T_{\text{total}},
\]

(26a)

\[
T = P_{\tau\pi},
\]

(26b)

\[
\tau \geq 1,
\]

(26c)

\[
\pi \geq 1.
\]

(26d)

From constraints (26a) and (26b), we obtain

\[
1 = T_{\text{total}} \cdot \frac{1}{T} + \frac{1}{\tau} + \frac{1}{\pi} + \frac{T_{\text{comp}}}{T_{\text{total}}},
\]

(27)

Substituting (27) into (26), we can eliminate the equation constraints. We also define

\[
q(\tau, \pi) \triangleq \frac{1}{2T\omega^2\sigma^2} = \frac{T_{\text{comp}} + T_{\text{w2e}}}{2T_{\text{total}}\omega^2\sigma^2} \cdot \frac{1}{\tau} + \frac{T_{\text{comp}} + T_{\text{e2c}}}{2T_{\text{total}}\omega^2\sigma^2} \cdot \frac{1}{\pi},
\]

(28)

The problem (26) can be re-formulated as

\[
\min_{\tau, \pi} q(\tau, \pi) + \rho_j(\tau, \pi, \delta, \ell, \delta) + \sqrt{q^2(\tau, \pi) + \frac{\rho_j(\tau, \pi, \delta, \ell, \delta)}{\omega^2\alpha^2\tau^2}},
\]

(29)

\[
\text{subject to } \tau \geq 1,
\]

(29a)

\[
\pi \geq 1.
\]

(29b)

It is non-trivial to find a closed-form optimal pair of \((\tau, \pi)\) in the three-tier hierarchical FL because problem (29) includes both polynomial and exponential terms of \(\tau\) and \(\pi\), where the exponential term is nested-embedded in \(h(\cdot)\) that is embedded in \(j(\cdot)\). Even if for a two-tier FL problem, the objective function of the bound is complicated, and it is still infeasible to find an optimal solution in closed form [41], [43]. In what follows, we propose the Hierarchical Optimizing Periods (HierOPT) algorithm to find a local optimal solution to problem (29).

In Algorithm 2, for convenience, we define the objective function (29) as \(R(\tau, \pi)\) with respect to \(\tau\) and \(\pi\). We also define the partial derivative of \(\tau\) and \(\pi\) as \(R'(\tau)\) and \(R'(\pi)\) respectively. Since \(R(\tau, \pi)\) is in closed-form, \(R'(\tau)\) and \(R'(\pi)\) are also in closed-form and can be calculated numerically given any \(\pi\) and \(\tau\) respectively. Algorithm 2 is operated as follows: \(1\) We take turns to calculate \(R'(\tau)\) (Lines 3–8) and \(R'(\pi)\) (Lines 9–14). When the gradient is greater than zero, implying that the objective function has the trend to increase, we decrease the value by 1 (Lines 5 and 11). When the gradient is less than zero, implying that the objective function has the trend to decrease, we increase the value by 1 (Lines 7 and 13). Due to constraints (29a) and (29b), we restrict the values of \(\tau\) and \(\pi\) to be equal or greater than 1. \(2\) If the pair of value \((\tau, \pi)\) is visited before (Lines 16–19), it means Algorithm 2 converges and \((\tau, \pi)\) oscillates within a number of feasible value pairs (because \(\tau\) and \(\pi\) can only be integers). In this case, we find a local optimal pair of \((\tau^*, \pi^*)\) and we can exit the algorithm.

VI. EXPERIMENTAL RESULTS

In this section, we evaluate the convergence performance of HierMo compared with three typical categories of benchmark algorithms: \(1\) three-tier FL without momentum (Hier-FAVG [18] and CFL [19]), \(2\) two-tier FL with momentum (FastSlowMo [24], DOMO [25], FedADC [26], FedMom [20],

Algorithm 2: HierOPT Algorithm.

**Input:** \(T_{\text{total}}, T_{\text{w}}, T_{\text{comp}}, T_{\text{w2e}}, T_{\text{e2c}}, \omega, \alpha, \sigma, \nu, j, h, R, \rho, q\)

**Output:** \(\tau^*\) and \(\pi^*\)

1: Initialize \(\tau_0\) and \(\pi_0\) as random positive integers, \(i = 0\) as the index of search iteration.
2: while \(\text{true} \) do
3: Calculate \(R'(\tau_i)\)
4: if \(R'(\tau_i) > 0\) then
5: \(\tau_{i+1} \leftarrow \max\{\tau_i - 1, 1\}\)
6: else if \(R'(\tau_i) < 0\) then
7: \(\tau_{i+1} \leftarrow \tau_i + 1\)
8: end if
9: if \(R'(\pi_i)\) then
10: \(\pi_{i+1} \leftarrow \max\{\pi_i - 1, 1\}\)
11: else if \(R'(\pi_i) < 0\) then
12: \(\pi_{i+1} \leftarrow \pi_i + 1\)
13: end if
14: end if
15: Record \((\tau_i, \pi_i)\)
16: if the pair of values \((\tau, \pi)\) is visited before then
17: \(\tau^* \leftarrow \tau_i\) and \(\pi^* \leftarrow \pi_i\)
18: BREAK
19: end if
20: \(i \leftarrow i + 1\)
21: end while

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
SlowMo [21], FedNAG [22], and Mime [23]), and (3) two-tier FL without momentum (FedAvg [4]). We use two different KPIs to compare the convergence performance. We first evaluate the training accuracy under a given number of training iterations. We then compare the total training time required by each algorithm to achieve a given training accuracy (e.g., 95%). For the two-tier benchmarks, we assume that the edge nodes do not exist and the workers are directly connected to the cloud. We then discuss the effects of \( \tau \) and \( \pi \) respectively and their joint effects. Afterwards, we explicitly quantify different levels of non-i.i.d. data and analyze their effects. Finally, we implement a real-world three-tier hierarchical FL system to test the overall training time. Through this way, we verify that \((\tau^*, \pi^*)\) derived in Section V leads to near-optimal performance in the realistic scenario.

A. Experiment on Convergence of HierMo

1) Experimental Setup: We employ four real-world datasets including MNIST [30], CIFAR-10 [31], and ImageNet [29], [32] for image classification, and UCI-HAR [33] for human activity recognition. All training and testing samples are randomly shuffled and distributed to workers. Please note there is no restriction on how the data is distributed at different workers, therefore, the level of non-i.i.d. data distribution captured by \( \delta_{i,\ell} \) is different for each worker \( \{i, \ell\} \).

In this experiment, we focus on the convergence performance (i.e., accuracy comparison given the number of iterations) of different algorithms. We do not consider the real-world delay for now. The results do not depend on hardware but on the algorithm itself. Therefore, we use a GPU tower server with 4 NVIDIA GeForce RTX 2080Ti GPUs to create several virtual machines within a single server to carry out the experiment. (Even if real-world hardware is used in the experiment, it will still give the same results.) The experiment on the optimization considering real-world delay will be discussed in Section VI-B.

We use five models including linear regression, logistic regression, CNN, VGG16, and ResNet18. The CNN model’s structure is the classic one in [27], which has two \( 5 \times 5 \) convolutional layers with 32 and 64 channels respectively. In each convolutional layer, \( 2 \times 2 \) max pooling is used. The last three layers are fully connected layers with ReLu activation and softmax. The structure of VGG16 and ResNet18 can be found in [28], [29] respectively. We use mini-batch in all experiments, and the batch size is 64. We set the learning rate \( \eta = 0.01 \). The specific settings for each experiment are as follows.

- **Performance comparison (Table II):** The experiment is conducted on linear regression, logistic regression, CNN, VGG16, and ResNet18. We set \( T = 1000 \) (MNIST), \( T = 4000 \) (UCI-HAR), or \( T = 10000 \) (CIFAR10 and ImageNet), \( \gamma = 0.5, \gamma_e = 0.5 \). There are 4 workers and 2 edge nodes with each edge node serving 2 workers (three-tier algorithm). There are 4 workers directly served by the cloud (two-tier algorithm). For two-tier algorithms, we set \( \tau = 20 \) (convex model) or \( \tau = 40 \) (non-convex model). For three-tier algorithms, we set \( \tau = 10, \pi = 2 \) (convex model) or \( \tau = 20, \pi = 2 \) (non-convex model). Please note that since \( \tau \) does not exist for two-tier algorithms, we set \( \tau \) value for two-tier algorithms equal to \( \tau \pi \) value for three-tier algorithms for a fair comparison. These hyper-parameters are typically used in existing works [8], [13], [14], [18], [22].

- **Effects of \( \tau \) and \( \pi \) (Fig. 3(a)–(c)):** The experiment is conducted when CNN is trained on MNIST. We set \( T = 1000, \gamma = 0.5, \gamma_e = 0.5 \). There are 16 workers and 4 edge nodes with each edge node serving 4 workers.

- **Effects of non-i.i.d. data (Fig. 3(e)–(g)):** The experiment is conducted when CNN is trained on MNIST. We set \( \tau = 40 \) (two-tier) or \( \tau = 20, \pi = 2 \) (three-tier), and \( T = 1000 \). There are 4 workers and 2 edge nodes with each edge node serving 2 workers (three-tier algorithm). There are 4 workers directly served by the cloud (two-tier algorithm).

2) Performance Comparison: In Table II, we compare the convergence performance of HierMo with benchmark algorithms. The numbers show the accuracy when different algorithms for a fair comparison. These hyper-parameters are typically used in existing works [8], [13], [14], [18], [22].

| Algorithm | MNIST | MNIST | MNIST | CIFAR10 | CIFAR10 | ImageNet | UCI-HAR |
|-----------|-------|-------|-------|---------|---------|----------|---------|
| Linear    | 85.97 | 89.25 | 96.13 | 64.18   | 90.06   | 69.64    | 88.36   |
| Logistic  | 83.62 | 87.00 | 93.40 | 38.46   | 89.46   | 68.63    | 54.56   |
| CNN       | 83.36 | 86.98 | 93.58 | 38.79   | 89.80   | 68.77    | 59.19   |
| CFL [19]  | 85.79 | 89.02 | 95.90 | 59.39   | 88.53   | 67.05    | 81.15   |
| FastSlowMo [24] | 85.79 | 89.02 | 95.90 | 59.39   | 88.53   | 67.05    | 81.15   |
| DOMO [25] | 83.74 | 88.83 | 95.59 | 63.49   | 89.16   | 67.58    | 86.05   |
| FedADC    | 85.51 | 88.18 | 95.09 | 56.60   | 89.38   | 76.76    | 85.14   |
| FedMom [20] | 84.84 | 88.05 | 94.74 | 54.87   | 88.03   | 66.91    | 84.69   |
| SlowMo [21] | 84.82 | 88.00 | 94.88 | 54.43   | 88.47   | 66.84    | 83.03   |
| FedNAG    | 84.97 | 88.14 | 95.04 | 55.54   | 88.33   | 68.11    | 84.69   |
| Mime [23] | 84.41 | 86.73 | 93.89 | 48.24   | 81.76   | 64.33    | 76.75   |
| FedAvg [4] | 83.57 | 86.89 | 93.34 | 37.79   | 88.27   | 66.59    | 53.31   |

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
This verifies that when two types of momentum are applied, the three-tier architecture outperforms the two-tier architecture. This is because the additional edge aggregation can decrease the effect of data heterogeneity among workers under the same edge node, so as to improve the performance.

Comparing FastSlowMo, DOMO and FedADC with FedMom, SlowMo, FedNAG, and Mime, we observe that FastSlowMo ≈ DOMO > FedADC > FedNAG > FedMom ≈ SlowMo > Mime. This confirms that using combined worker momentum and aggregator momentum can accelerate the convergence compared with those using momentum only on workers or only on the aggregator. For worker momentum only or aggregator momentum only algorithms, we can still observe their acceleration compared with FedAvg. We also observe Mime may not perform well. Sometimes, it is even worse than FedAvg. This is because Mime uses the fixed momentum value in worker momentum update, where such value can be refreshed only in the global aggregation phase. As a result, the momentum value may be stale, especially when τ is as large as 40.

Comparing HierFAVG and CFL with two-tier momentum-based algorithms (FastSlowMo, DOMO, FedADC, FedMom, SlowMo, FedNAG, and Mime), we observe that for DNN, HierFAVG and CFL outperform two-tier momentum-based algorithms, while for convex model and CNN, the later is better. This shows that for complicated models, the three-tier architecture plays a more significant role to accelerate the convergence while for less complicated models, the momentum plays a more significant role to accelerate the convergence.

We also compare the training accuracy when more workers (N = 50 and N = 100) participate the training to demonstrate the cross-siloed FL [40] (typically up to one hundred participants). The results in Fig. 3(d) and (h) show the same trend as results in Table II.

3) Effects of τ and π: In Fig. 3(a)–(c), we evaluate the effects of τ and π, and their joint effects. The curves in the figure show the accuracy when CNN is trained on MNIST. When π and τ are fixed in Fig. 3(a) and (b) respectively, we observe that larger τ or π lowers the performance. This observation matches our expectation and verifies the result of Theorem 4 showing that the larger τ or π leads to larger convergence upper bound. When τ · π (the product of τ and π) is fixed in Fig. 3(c), we observe that smaller τ (larger π) leads to better performance. This shows...
that more frequent edge aggregation is more effective compared with more frequent cloud aggregation.

4) Effects of Non-i.i.d. Data Distribution: In Fig. 3(e)–(g), we evaluate the effects of different levels of non-i.i.d. data distribution. The curves show the training accuracy. To quantify the level of non-i.i.d. data distribution, we explicitly assign only $x < 10$ out of 10 classes of data for each worker. (Each worker has data samples from a subset of classes.) The class of data is randomly allocated to each worker. Smaller $x$ represents higher level of non-i.i.d. setting. We use 3-class non-i.i.d., 6-class non-i.i.d., and 9-class non-i.i.d. to represent high, middle and low level of non-i.i.d. data respectively.

We observe that HierMo > HierFAVG > DOMO ≈ FastSlowMo > FedADC > FedNAG > CFL > FedMom > SlowMo > Mime ≈ FedAvg in most cases. This is consistent with the results in Table II, showing that HierMo outperforms all benchmarks under any levels of non-i.i.d. data distribution. We also observe higher level of non-i.i.d. setting decreases convergence performance for all algorithms. Specifically, HierMo achieves 66.11% accuracy for high level non-i.i.d. data, while achieving 92.21% accuracy and 94.70% accuracy for middle and low level non-i.i.d. data respectively. This matches our expectations where higher level of non-i.i.d. setting causes more data divergence that is denoted by larger $\delta$, and therefore lowers the accuracy.

B. Experiment on Real-World Hierarchical FL System

1) Experimental Setup: We build up a real-world hierarchical FL system to implement image classification application. We evaluate the performance of HierMo and HierOPT in the following aspects. 1) To reach a target training accuracy (0.95), we compare the total training time of HierMo and benchmarks. 2) For a given total training time $T_{\text{total}}$, we compare the performance of HierMo under different $(\tau, \pi)$ and verify that $(\tau^*, \pi^*)$ derived by HierOPT is near optimal.

We use four mini-PCs with Intel Core i3 10105f CPU as workers, two laptops (Macbook Pro 2018 with Intel Core i7-8750H CPU and Dell Inspiron 15 with Intel Core i5 7300HQ CPU) as edge nodes, and one GPU Tower Server (Intel Xeon Bronze 3104 CPU, 4 NVIDIA GeForce RTX 2080Ti GPUs) as the cloud server. The workers are connected to their corresponding edge nodes with edge nodes’ 5 GHz WIFI hotspots. The edge nodes are connected to Redmi AC2100 router with wired cables (1 Gbps Ethernet). The router is then connected to the public Internet. The cloud server is also connected to the public Internet via another ISP’s access network. Please note that in two-tier FL system, since workers directly communicate with the cloud server, the edge nodes are only responsible for forwarding data. The system architecture is shown in Fig. 4. The calculation of total training time starts when the cloud server triggers four workers to start training, and ends when the cloud server evaluates that the training accuracy reaches a given threshold (e.g., 0.95). Since HierMo follows the classical synchronized cross-silo FL [40]. Only after the aggregator receives all the workers’ local models, it can then proceed the global model aggregation. Therefore, for inputs of HierOPT, the computation and communication delays are sampled from the device with the longest delays. Please note that the delays we sample may exhibit deviations, but these deviations are typically stable within an acceptable range. To address this, we gather multiple delay samples and compute their average to obtain the final delays (i.e., the averaged maximum delays). Such sampling method is commonly used in the assumptions for theoretical analysis in the literature [8], [18], [43].

For total training time comparison (Fig. 5), the experiment is conducted under two settings when CNN is trained on MNIST: $\gamma$ = 0.5, $\tau$ = 20 (two-tier) and $\tau$ = 10, $\pi$ = 2 (three-tier) and $\gamma$ = 0.5, $\tau$ = 40 (two-tier) or $\tau$ = 20, $\pi$ = 2 (three-tier). For performance of HierOPT (Fig. 6), the experiment is conducted when CNN is trained on MNIST and CIFAR10. We set $\gamma$ = 0.5, $\tau$ = 0.5, $T_{\text{total}}$ = 200 s or $T_{\text{total}}$ = 400 s (MNIST), and $T_{\text{total}}$ = 2500 s or $T_{\text{total}}$ = 5000 s (CIFAR10).

2) Total Training Time Comparison: In Fig. 5, we compare the total training time of HierMo and benchmarks when CNN is trained on MNIST. We observe that to reach the accuracy 0.95, HierMo spends 522.64 s under setting 1 and 462.58 s under setting 2 while other benchmarks spend 551.24s–1582.16 s under setting 1 and 617.98s–1722.08 s under setting 2 respectively. This demonstrates that HierMo is efficient and decreases the total training time by 5–73% compared with the benchmarks.

3) Performance of HierOPT: In Fig. 6, we illustrate the performance of HierOPT. All constants in the objective function (29) can be sampled in advance of the training process [41], [43]. We show the accuracy under different pairs of $(\tau, \pi)$ and flag $(\tau^*, \pi^*)$ derived by HierOPT. The darker color in the chromatography indicates a higher training accuracy. The red cross indicates the derived $(\tau^*, \pi^*)$ by HierOPT. We observe that in all
figures, HierOPT can find near-optimal solutions. In Fig. 6(a), when $T^{total} = 200$ s, the optimal accuracy is 92.66%, with optimal $(\tau, \pi) = (21, 2)$, while HierOPT finds $(\tau^*, \pi^*) = (22, 2)$, with accuracy 92.65%, only a 0.01% gap from the optimum. In Fig. 6(b), when $T^{total} = 400$ s, the optimal accuracy is 94.83%, with optimal $(\tau, \pi) = (29, 2)$, while HierOPT finds $(\tau^*, \pi^*) = (32, 2)$, with accuracy 94.76%, only a 0.07% gap from the optimum. For CIFAR10, HierOPT can still find the near-optimal $(\tau^*, \pi^*)$, with only 0.41% (51.98% to 51.57%) and 0.24% (65.60% to 65.36%) gap from the real-world optimum, when $T^{total} = 2500$ s and $T^{total} = 5000$ s respectively.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose HierMo, a three-tier hierarchical FL algorithm that applies momentum to accelerate convergence. We provide convergence analysis for HierMo, showing that it converges with a rate of $O\left(\frac{1}{s}\right)$ for smooth non-convex problems under non-i.i.d. data. In the analysis, we develop a new two-level virtual update (edge and cloud) method to characterize the multi-time cross-two-tier momentum interaction and the cross-three-tier momentum interaction. The performance gain of momentum is also quantified. We also propose HierOPT to derive a near-optimal setting of worker-edge and edge-cloud aggregation periods $(\tau, \pi)$ under a limited total training time. We verify that HierMo outperforms existing mainstream benchmarks under a wide range of settings. In addition, HierOPT can achieve a near-optimal performance when we test HierMo under different values of $(\tau, \pi)$.

There are still some works that can be studied in the future. In this paper, we only consider the typical three-tier FL scenario in a cross-siloed distributed environment. First, we can extend our algorithm to cross-device [40] environments by selecting partial workers to participate in the training. It brings substantial analytical challenges, but is a promising investigation scenario. Second, we can use quantization [44] techniques to apply joint quantization on worker-edge-cloud momenta. This helps reduce the size of transmission payloads, thereby reducing communication overhead. Third, we can explore extending the fixed hyper-parameters setting of momentum to adaptive and personalized momentum to obtain a more personalized FL model [45].

REFERENCES

[1] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” Int. J. Remote Sens., vol. 28, no. 5, pp. 823–870, 2007.

[2] J. E. Naranjo, C. González, R. García, T. de Pedro, and R. E. Haber, “Power-steering control architecture for automatic driving.” IEEE Trans. Intell. Transp. Syst., vol. 6, no. 4, pp. 406–415, Dec. 2005.

[3] D. Yu and L. Deng, Automatic Speech Recognition, vol. 1, Berlin, Germany: Springer, 2016.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agiera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282.

[5] W. Y. B. Lim et al., “Federated learning in mobile edge networks: A comprehensive survey,” IEEE Commun. Surveys Tuts., vol. 22, no. 3, pp. 1333–1363, Third Quarter 2020.

[6] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, “Hierarchical federated learning across heterogeneous cellular networks,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2020, pp. 8866–8870.

[7] C. Briggs, Z. Fan, and P. Andras, “Federated learning with hierarchical clustering of local updates to improve training on non-IID data,” in Proc. Int. Joint Conf. Neural Netw., 2020, pp. 1–9.

[8] L. Liu, J. Zhang, S. Song, and K. B. Letaief, “Hierarchical federated learning with quantization: Convergence analysis and system design,” IEEE Trans. Wireless Commun., vol. 22, no. 1, pp. 2–18, Jan. 2023.

[9] Y. Yan, T. Yang, Z. Li, Q. Lin, and Y. Yang, “A unified analysis of stochastic momentum methods for deep learning,” in Proc. 27th Int. Joint Conf. Artif. Intell., 2018, pp. 2955–2961.

[10] C. Liu and M. Belkin, “Accelerating SGD with momentum for over-parameterized learning,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–10.

[11] S. Vaswani, F. Bach, and M. Schmidt, “Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron,” in Proc. 22nd Int. Conf. Artif. Intell. Statist., 2019, pp. 1195–1204.

[12] M. Assran and M. Rabbat, “On the convergence of Nesterov’s accelerated gradient method in stochastic settings,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 410–420.

[13] W. Liu, L. Chen, Y. Chen, and W. Zhang, “Accelerating federated learning via momentum gradient descent,” IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 8, pp. 1754–1766, Aug. 2020.

[14] H. Yu, R. Jin, and S. Yang, “On the linear speedup analysis of communication-efficient momentum SGD for distributed non-convex optimization,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 7184–7193.

[15] H. Yang, M. Fang, and J. Liu, “Achieving linear speedup with partial worker participation in non-IID federated learning,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–11.

[16] H. Gao, A. Xu, and H. Huang, “On the convergence of communication-efficient local SGD for federated learning,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 7510–7518.

[17] S. Ruder, “An overview of gradient descent optimization algorithms,” 2016, arXiv:1609.04747.

[18] L. Liu, J. Zhang, S. Song, and K. B. Letaief, “Client-edge-cloud hierarchical federated learning,” in Proc. IEEE Int. Conf. Mach. Learn., 2020, pp. 6–6.

[19] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, “Resource-efficient federated learning with hierarchical aggregation in edge computing,” in Proc. IEEE Conf. Comput. Commun., 2021, pp. 1–10.

[20] Z. Huo et al., “Faster-on-device training using new momentum algorithm,” 2020, arXiv:2002.02990.

[21] T. Wang, V. Tantia, N. Ballas, and M. Rabbat, “SlowMo: Improving communication-efficient distributed SGD with slow momentum,” in Proc. Int. Conf. Learn. Representations, 2020, pp. 1–12.

[22] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, “Federated learning with Nesterov accelerated gradient,” IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 12, pp. 4863–4873, Dec. 2022.

[23] S. P. Karimikty et al., “Mime: Mimicking centralized stochastic algorithms in federated learning,” 2020, arXiv:2008.03606.

[24] Z. Yang, S. Fu, W. Bao, D. Yuan, and A. Y. Zomaya, “FastSlowMo: Federated learning with combined worker and aggregator momentum,” IEEE Trans. Artif. Intell., to be published, doi: 10.1109/TAI.2022.3187678.

[25] A. Xu and H. Huang, “Coordinating momentum for cross-silo federated learning,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 8735–8743.

[26] E. Ozfatura, K. Ozfatura, and D. Gündüz, “FedADC: Accelerated federated learning with drift control,” in Proc. IEEE Int. Symp. Inf. Theory, 2021, pp. 467–472.

[27] M. Jay et al., FL on MNIST Using a CNN, 2023. [Online]. Available: https://github.com/OpenMined/Pysyft/tree/dev/notebooks/tutorials

[28] S. Gross et al., Pytorch-VGG, May 2021. [Online]. Available: https://github.com/pytorch/vision/blob/master/torchvision/models/vgg.py

[29] T. Moon and T. Ryffel, Pytorch-TinyImageNet, Jun. 2020. [Online]. Available: https://github.com/tjmoon0104/pytorch-tiny-imagenet

[30] Y. Lecun, Y. Bengio, and G. Hinton, “Gradient-based learning applied to document recognition,” in Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[31] N. Srivastava et al., “Dropout: A simple way to prevent neural networks from overfitting,” 2014, arXiv:1404.9832.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2009, pp. 248–255.

[33] D. Anguita, A. Ghio, L. Oneto, X. P. Perez, and J. L. R. Ortiz, “A public domain dataset for human activity recognition using smartphones,” in Proc. 21st Int. Eur. Symp. Artif. Neural Netw. Comput. Intell. Mach. Learn., 2013, pp. 437–442.
[34] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” USSR Comput. Math. Math. Phys., vol. 4, no. 5, pp. 1–17, 1964.

[35] G. Goh, “Why momentum really works,” Distill, 2017. [Online]. Available: http://distill.pub/2017/momentum

[36] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence o(1/k²),” Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543–547, 1983.

[37] S. Bubeck, “Convex optimization: Algorithms and complexity,” Foundations Trends Mach. Learn., vol. 8, no. 3–4, pp. 231–357, 2015.

[38] J. Wang, S. Wang, R.-R. Chen, and M. Ji, “Demystifying why local aggregation helps: Convergence analysis of hierarchical SGD,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 8548–8556.

[39] T. Castiglia, A. Das, and S. Patterson, “Multi-level local SGD: Distributed SGD for heterogeneous hierarchical networks,” in Proc. Int. Conf. Learn. Representations, 2021, pp. 1–11.

[40] P. Kairouz et al., “Advances and open problems in federated learning,” Found. Trends Mach. Learn., vol. 14, no. 1/2, pp. 1–210, 2021.

[41] S. Wang et al., “Adaptive federated learning in resource constrained edge computing systems,” IEEE J. Sel. Areas Commun., vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[42] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Berlin, Germany: MIT press, 2016.

[43] C. T. Dinh, N. H. Tran, T. D. Nguyen, W. Bao, A. Y. Zomaya, and B. B. Zhou, “Federated learning with proximal stochastic variance reduced gradient algorithms,” in Proc. 49th Int. Conf. Parallel Process., 2020, pp. 1–11.

[44] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, “FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization,” in Proc. Int. Conf. Artif. Intell. Statist., 2020, pp. 2021–2031.

[45] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, “Towards personalized federated learning,” IEEE Trans. Neural Netw. Learn. Syst., to be published, doi: 10.1109/TNNLS.2022.3160699.

Zhengjie Yang (Student Member, IEEE) received the master of information technology degree from the University of Sydney, in 2017. He is currently working toward the PhD degree with the School of Computer Science, University of Sydney, since 2019. In 2017–2018, he worked as a software engineer with the Garvan Institute of Medical Research and Link Group Pty Ltd in Sydney. His research focuses on the edge computing and distributed machine learning.

Wei Bao (Member, IEEE) received the BE degree in communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, the MASc degree in electrical and computer engineering from the University of British Columbia, Vancouver, Canada, in 2011, and the PhD degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2016. He is currently a senior lecturer with the School of Computer Science, University of Sydney, Sydney, Australia. His research covers the area of network science, with particular emphasis on Internet of things, mobile computing, edge computing, and distributed learning. He received the Best Paper Awards in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM) in 2013 and 2019 and IEEE International Symposium on Network Computing and Applications (NCA) in 2016.

Dong Yuan (Member, IEEE) received the BEng and MEng degrees in computer science from Shandong University, Jinan, China, in 2005 and 2008, respectively, and the PhD degree in computer science from the Swinburne University of Technology, Melbourne, Australia, in 2012. He is a senior lecturer from the School of Electrical and Information Engineering, University of Sydney. His research interests include cloud computing, parallel and distributed systems, scheduling and resource management, deep learning, data management, and Internet of Things.

Sen Fu received the bachelor of computer science and technology (Advanced) (Honours) degree from the University of Sydney, in 2021. He is currently working toward the PhD degree with the School of Computer Science, University of Sydney. His research focuses on the distributed machine learning.

Wei Bao (Member, IEEE) received the BE degree in communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, the MASc degree in electrical and computer engineering from the University of British Columbia, Vancouver, Canada, in 2011, and the PhD degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2016. He is currently a senior lecturer with the School of Computer Science, University of Sydney, Sydney, Australia. His research covers the area of network science, with particular emphasis on Internet of things, mobile computing, edge computing, and distributed learning. He received the Best Paper Awards in ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM) in 2013 and 2019 and IEEE International Symposium on Network Computing and Applications (NCA) in 2016.

Dong Yuan (Member, IEEE) received the BEng and MEng degrees in computer science from Shandong University, Jinan, China, in 2005 and 2008, respectively, and the PhD degree in computer science from the Swinburne University of Technology, Melbourne, Australia, in 2012. He is a senior lecturer from the School of Electrical and Information Engineering, University of Sydney. His research interests include cloud computing, parallel and distributed systems, scheduling and resource management, deep learning, data management, and Internet of Things.

Albert Y. Zomaya (Fellow, IEEE) is Peter Nicol Russell chair professor of computer science and director of the Centre for Distributed and High-Performance Computing, University of Sydney. Till date, he has published more than 700 scientific papers and articles and is (co-)author/editor of more than 30 books. A sought-after speaker, he has delivered more than 250 keynote addresses, invited seminars, and media briefings. He was editor in chief of the IEEE Transactions on Computers (2010-2014) and the IEEE Transactions on Sustainable Computing (2016-2020). He is a decorated scholar with numerous accolades including fellowship of the IEEE, the American Association for the Advancement of Science, and the Institution of Engineering and Technology. Also, he is a fellow of the Australian Academy of Science, Royal Society of New South Wales, foreign member of Academia Europaea, and member of the European Academy of Sciences and Arts. Some of him recent awards include the Research Innovation Award, IEEE Technical Committee on Cloud Computing (2021), the Technical Achievement and Recognition Award, IEEE Communications Society’s IoT, Ad Hoc and Sensor Networks Technical Committee (2022). He is a Clarivate 2022 highly cited researcher, and his research interests lie in parallel and distributed computing, networking, and complex systems.