Communication-Efficient Federated Learning With Compensated Overlap-FedAvg

Yuhao Zhou, Qing Ye, and Jiancheng Lv, *Member, IEEE*

Abstract—While petabytes of data are generated each day by a number of independent computing devices, only a few of them can be finally collected and used for deep learning (DL) due to the apprehension of data security and privacy leakage, thus seriously retarding the extension of DL. In such a circumstance, federated learning (FL) was proposed to perform model training by multiple clients' combined data without the dataset sharing within the cluster. Nevertheless, federated learning with periodic model averaging (FedAvg) introduced massive communication overhead as the synchronized data in each iteration is about the same size as the model, and thereby leading to a low communication efficiency. Consequently, variant proposals focusing on the communication rounds reduction and data compression were proposed to decrease the communication overhead of FL. In this article, we propose Overlap-FedAvg, an innovative framework that loosed the chain-like constraint of federated learning and paralleled the model training phase with the model communication phase (i.e., uploading local models and downloading the global model), so that the latter phase could be totally covered by the former phase. Compared to vanilla FedAvg, Overlap-FedAvg was further developed with a hierarchical computing strategy, a data compensation mechanism, and a nesterov accelerated gradients (NAG) algorithm. In Particular, Overlap-FedAvg is orthogonal to many other compression methods so that they could be applied together to maximize the utilization of the cluster. Besides, the theoretical analysis is provided to prove the convergence of the proposed framework. Extensive experiments conducting on both image classification and natural language processing tasks with multiple models and datasets also demonstrate that the proposed framework substantially reduced the communication overhead and boosted the federated learning process.

Index Terms—Distributed computing, federated learning, overlap, efficient communication

1 Introduction

With the rapid development of deep learning, tons of daily generated data that were previously considered useless can now be utilized to extract latent patterns through deep learning (DL) [1], and thereby retrieving valuable information. Similarly, owning abundant data resources is a prerequisite for many innovative breakthroughs in terms of academia across different fields, including Natural Language Processing [2], [3] and Computer Vision [4], [5]. On the other hand, thanks to the popularity of independent computing devices [6] (e.g., smartphone, laptop) and edge devices (e.g., router), each individual now can produce more data than ever before, with gigabytes or even terabytes expected to be generated every day. However, these fresh data are tied to different facilities, resulting in data fragmentation (i.e., isolated data islands problem). In other words, since separated facilities like smartphones are mostly equipped with lots of sensors (e.g., camera, microphone, GPS), and common users generally have no idea either what applications are collecting information through these sensors or how this collected information would be used for, these users will take considerable risks when they decide to share this information with others. As a result, while the demand for data and the speed of creating data have both been increased dramatically, it is still extremely difficult to gather these data together, processing, and making the most of them into a collection for learning, especially considering privacy is increasingly valued.

To tackle this problem, federated learning (FL) [7] was proposed to introduce a new promising centralized distributed training method [8] that allows multiple clients (e.g., mobile clients or servers from different enterprises) to coordinately train a deep neural network model on their combined data, without any of the participants having to reveal its local data to the central server or other participants. Specifically, the workflow of the federated learning with FederatedAveraging (i.e., FedAvg [7]), as Fig. 1 illustrated, has 4 steps: I. The central server initializes the global model and pushes the global model to every participating client. II. Clients train the received global model on its local data. After that, they return the essential information I_t^k that is beneficial to the evolving of the global model (i.e., mostly is model weights, but could be gradients) to the server, where k is the index of the client and the t indicates the iteration index, respectively. III. The central server utilizes all clients' information by $\sum_k I_t^k$ to update the global model. IV. Repeat step II and step III until convergence. Particularly, it can be seen that federated learning omits the data collection step, and conversely utilizing I_t^k that is not explicitly related to clients' local data to evolve the global model, which not only integrates fragmented data, but also offers as much privacy as possible.

Nonetheless, preserving data privacy by not revealing each client's local data in federated learning comes at a

• *The authors are with the College of Computer Science, Sichuan University, Chengdu 610017, China. E-mail: sooptq@gmail.com, fuyeking@stu.scu.edu.cn, lvjiancheng@scu.edu.cn.*

Manuscript received 2 Jan. 2021; revised 23 Mar. 2021; accepted 6 June 2021. Date of publication 17 June 2021; date of current version 8 July 2021.
(Corresponding author: Jiancheng Lv.)
Recommended for acceptance by P. Balaji.
Digital Object Identifier no. 10.1109/TPDS.2021.3090331

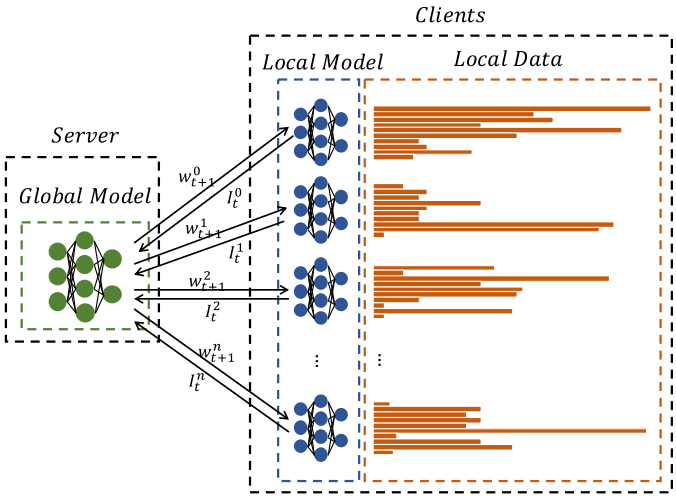

Fig. 1. A general federated learning workflow. Each client always keeps its data private and only shares essential information that could help improve the global model with others, be it gradients or model weights.

price, especially considering the fact that deep learning is basically built on top of huge chunks of data. In such a circumstance, the price for federated learning [9] is low communication efficiency due to overloaded bandwidth since the client-server communication occurs at the end of every training iteration. In detail, referring to the workflow of federated learning as shown in Fig. 1, we can see that at each iteration, every client has to communicate a I_t^k which commonly has the same size as the model. Given that nowadays the model size can be easily reached hundreds of megabytes [2], [10], [11] or even gigabytes, plus the model in the federated learning generally takes hundreds of thousands of iterations to converge, the overall data transfer size during the training process may exceed petabytes readily [12]. Hence, in comparison to non-privacy-preserving training, federated learning usually takes much more time to converge due to the time-consuming communication phase, and it may become even slower when the network condition is relatively poorer.

To reduce the communication cost of federated learning, various methods were proposed to explore the possible solutions, which can be mainly divided into two different categories in this paper. Specifically, the first category is the model averaging with a large communication interval. Some work [7], [13] tried to drastically decrease the communication overhead by enlarging the communication interval, yet it also degraded the model's final accuracy and the best communication step size is hard to capture. The second category is data compression. Some work [14], [15], [16], [17], [18] attempted to compress the I_t^k before transmitting it. However, the compressing phase needs to consume a lot of time as well, especially when performing federated learning on battery-sensitive (e.g., smartphone) or low-performance (e.g., netbook, gateway) devices.

In this paper, we propose a new novel framework named Overlap-FedAvg to alleviate this problem, which paralleled the model training phase with the model uploading and downloading phase, so that the latter phase could be totally covered by the former phase. Compared to the vanilla FedAvg, Overlap-FedAvg was first further developed with a hierarchical computing strategy to accomplish the parallelism. Nevertheless, this strategy also brought the gradient staleness problem so a gradient compensation mechanism was therefore employed to alleviate this issue and keep the convergence of the proposed Overlap-FedAvg framework. Additionally, a nesterov accelerated gradients (NAG) [19], [20] algorithm was also established to accelerate the convergence speed of the federated learning. Besides, Overlap-FedAvg is orthogonal to other compression methods so that it could collaborate with others to further improve the efficiency of the federated learning.

The contributions of this proposal are the followings:

- To improve the communication efficiency of federated learning, we propose a novel communication-efficient framework called Overlap-FedAvg by paralleling the communication with computation. To the best of our knowledge, it is the first work that focuses on improving the communication efficiency of federated learning by an overlapping approach, and it is totally compatible with many other data compression methods.
- In order to alleviate the gradients staleness problem brought by the parallelism in Overlap-FedAvg and keep the convergence speed of the federated learning, taylor series expansion was utilized as a compensation mechanism. Additionally, the essential theoretical analysis of the proposed Overlap-FedAvg algorithm is provided.
- To achieve higher accuracy, NAG was employed to further accelerate the federated learning. Compared with previous similar work, our proposal was not only effective but also intuitive and easy to implement. Note that theoretically, not only the NAG algorithm but all acceleration algorithms (e.g., Adam [21], RMSProp [22], etc.) could be applied to the Overlap-FedAvg framework readily.
- To investigate the effectiveness of the proposed Overlap-FedAvg, extensive experiments were conducted on four benchmark datasets with 8 DNN models to compare its potentiality with the baseline algorithm FedAvg. The results demonstrated that our proposed Overlap-FedAvg could effectively improve the accuracy of federated learning and decrease the communication cost. The source code and hyper-parameters of all experiments are open-sourced for reproducibility.[1] In addition, we discuss whether the proposed Overlap-FedAvg would damage the privacy security of federated learning.

The rest of this paper is organized as follows. A literature review is illustrated in Section 2, where some background information and related work are introduced. In Section 3, the implementations of the proposed Overlap-FedAvg algorithm are presented. Then the essential convergence analysis is described in Section 4. The experiment design and result analysis are detailedly documented in Section 5. Finally, the discussion and conclusion of this paper are drawn in Section 6.

1. https://github.com/Soptq/Overlap-FedAvg

2 Literature Review

In this section, we give a brief introduction of federated learning, and some related work that accelerates the training process of federated learning by improving communication efficiency.

Algorithm 1. `FederatedAveraging`. In the Cluster There are N Clients in Total, Each With a Learning Rate of η. The Set Containing All Clients is Denoted as S, the Communication Interval is Denoted as E, and the Fraction of Clients is Denoted as C

Central server do:
Initialization: global model w_0.
for each global iteration $t \in 1, \ldots, iteration$ **do**
  # Determine the number of participated clients.
  $m \leftarrow max(C \cdot N, 1)$
  # Randomly choose participated clients.
  $S_p = random.choice(S, m)$
  for all each client $k \in S_p$ **do in parallel**
    # Get clients improved model.
    $w_{t+1}^k \leftarrow TrainLocally(k, w_t)$
  end for
  # Update the global model.
  $w_{t+1} \leftarrow \sum_{k=0}^{N} p_k w_{t+1}^k$
end for
TrainLocally(k**,** w_0**):**
for each client iteration $e \in 1, \ldots, E$ **do**
  # Do local model training.
  $w_e \leftarrow w_{e-1} - \eta \nabla F(w_{e-1})$
end for
return w_E

2.1 Federated Learning

Federated learning generally describes a distributed framework that allows the cluster to perform model training on all clients' combined datasets without leaking their respective local data to others. This is accomplished by letting every participating client only sends information that could help evolve the global model, instead of its plainly local dataset, to the central server. Federated learning is originated from distributed deep learning [23], [24], [25], but with mainly four additional features. First, the training data of different clients used in federated learning is expected to be heavily unbalanced. Namely, each client's local data tend to be both Non-IID (Independent and Identically Distributed) and unequal in amounts, and will not be unveiled to others during the training process. As a result, any node in the cluster, including the central server, cannot determine the data distribution of any other nodes. Second, federated learning has a relatively small training batch size. Virtually, federated learning was initially designed to make use of the data on battery-sensitive or low-performance devices, which usually also have a relatively small memory size. Consequently, the batch size of the model needs to be small enough so that the memory would not run out. Third, the network bandwidth of the clients in federated learning is always considerably small compared to the vanilla distributed training, where gigabytes bandwidth is commonly used. Hence, numerous communications produce massive communication overhead and take much time, resulting in low communication efficiency. Moreover, network connections of clients can be even lost during the training, indicating the system has to be robust enough to handle these exceptions. Forth, clients in federated learning are unreliable, meaning it is very potential that malicious clients send carefully constructed information to the central server to poison the global model, and eventually making the global model unusable. Extensive work was proposed to address the issues aforementioned [26], [27], [28], [29], [30] while we focus on alleviating the massive communication overhead problem of federated learning in this article.

2.2 Related Work

Various work had been proposed to address the massive communication overhead problem during federated learning, which can be roughly divided into two groups: *reducing communication rounds* and *data compression*

(1) *Reducing communication rounds:* In vanilla federated learning, the communication phase happened at the end of every iteration (i.e., the communication interval is 1). Considering a typical federated DNN training process usually takes hundreds of thousands of iterations, one can easily think of enlarging the communication interval to significantly reduce the communication overhead. As a result, FederatedAveraging (FedAvg) [7] and its variants [13] were proposed to allow clients do multiple iterations of local training before making a global model update, as Algorithm 1 illustrated, where $p_k = \frac{n_k}{n}$ is the weight of the kth client. The experiments suggested that FedAvg massively increased the convergence speed with respect to wall-clock time due to the reduction of communication rounds. In addition, several work provided the theoretical analysis of FedAvg, advocating that it is not only convergent on both IID and Non-IID data with *decaying learning rate*, but also has linear speed-up [31], [32]. However, The communication interval in FedAvg is controlled by a hyper-parameter E, which is greatly influential to the final accuracy of the model and is shifty based on the characteristic of both the model and the dataset. In fact, the choice of E is a trade-off between the final accuracy of DNN and the training efficiency: the smaller the E is, the better the model's final accuracy generally would be, and conversely the bigger the E is, the faster the model generally would converge. Therefore, an experienced engineer needs to be employed to fine-tune the communication interval E in order to extract the best performance of the model.

(2) *Data compression:* Apart from decreasing communication rounds, another approach to reduce the communication overhead is to reduce the data transfer size, namely data compression. Following this direction, there are mainly two kinds of methods to effectively compress the data: quantization [14], [15], [16] and sparsification [17], [18], where the former one aims to represent the original data by a low-precision data type with a smaller size (e.g., `int8` or `bool`), and the latter one intentionally only transmits

essential values at each communication (about 1 percent of the total number of values). In detail, quantization with error feedback is experimentally and theoretically effective [33], but its compression rate is considerably low as the maximum compression ratio is limited to $\frac{1}{32}$ (the common data type used in deep learning is 32-bit). Moreover, with fewer bits carrying information, quantization tends to converge much slower. On the other hand, sparsification is capable of achieving a compression rate of $\frac{1}{100}$ easily without significantly damaging the model's convergence speed and final accuracy. Nevertheless, sparsification inevitably introduces extra phases during the training process, including sampling, compressing, encoding, decoding, and decompressing, which could potentially affect the overall training efficiency, especially on battery-sensitive (e.g., smartphone) or low-performance (e.g., netbook, gateway) devices.

In this paper, we propose Overlap-FedAvg that attempted to overlap the communication with computation in federated learning to boost the training from a structural perspective. In fact, the idea of communication-computation overlap had been explored extensively in High-Performance Computing (HPC) [34], [35], [36]. However, they generally assumed the data and the computation is independent of each other so that data can be transmitted as soon as it is available. Nonetheless, in the training process of a DNN, each iteration of training seriously related to the data of the last iteration, leading to a chain-like structure that is hard to split. Consequently, ideas in the HPC domain fail to be directly applied to federated learning. To accelerate the training of federated learning, the Overlap-FedAvg is proposed, which is the first overlapping work in federated learning to our best knowledge. Particularly, the dependency problem was addressed by relaxing the chain-like constraint and a data compensation mechanism, which will be detailedly discussed in the latter section.

Moreover, compared to above-mentioned related work and the corresponding shortcomings, Overlap-FedAvg first does not require experts to carefully design E. Contrarily, E in the Overlap-FedAvg is automatically determined by the environment (i.e., network condition, bandwidth, etc) with the hierarchical computing strategy. In other words, if the environment permits a more frequent communication, E will be automatically tuned smaller, and vise versa. Second, Overlap-FedAvg does not compress any data during the training process. That is, there is no reduced convergence rate or extra time-consuming phases brought by Overlap-FedAvg in terms of compression. Moreover, Overlap-FedAvg is compatible with those compression methods, so that they can be applied together to the training process to further accelerate the training speed.

3 Methodology

In this section, we first show an overview of the proposed Overlap-FedAvg framework, then we present its details as well as implementations, and finally we introduce the gradients compensation mechanism and the application of the NAG algorithm.

3.1 Overview

To alleviate the massive communication problem, we propose a novel framework named Overlap-FedAvg, which decoupled the model uploading and downloading phase with the model training phase by employing a separate process dedicating to data transmission on each client. In this way, clients' local model training phases can be freed from the interruptions of the frequent communication. However, while this decoupling made the training phase capable of doing continuous model training without any blocking, it also introduced staleness to the synchronized data, which will be detailedly explained in Section 3.3. Observed that the weights of the model are generally equivalent to the aggregation of the gradients, we decomposed the vanilla FedAvg's model updating rule, abstracting the gradients, utilizing taylor series expansion and fisher information matrix to analyze the gap between the stale gradients and the up-to-date gradients, and thereby properly compensating the stale model weights. Moreover, with the decomposition and the abstraction of the gradients, many parameters optimizing acceleration algorithms can be easily applied to further increase the converge speed of federated learning, for example, momentum accelerated SGD, NAG, or even Adam [21].

In general, Overlap-FedAvg's workflow consists of five steps. I. Initiate the global model on the central server and push it to every participating client. II. Each participating client does E iterations of local model training on their respective data. Note that E here is no longer a constant hyper-parameter that needs to be manually setup, but a dynamic variable that adapts to the changing circumstances. III. When the local training is finished, clients instantly continue the next iteration of local training, while commanding another process to push their local model to the central server. IV. When the central server receives clients' improved models, it first uses taylor series expansion and fisher information matrix to compensate the model weights, then with the compensated weights, the central server calculates the nesterov momentum and updates the global model. V. The central server pushes the latest global model to the participating clients. For the architecture, the biggest difference between the Overlap-FedAvg and vanilla FedAvg is step III, where we relaxed the chain-like requirements, meaning the central server is not guaranteed for receiving the latest model from each client, which unavoidably brought staleness problem to our setting. Consequently, step IV managed to address the issue of stale model weights.

3.2 Overlapping of the Communication

At the beginning of the federated learning in Overlap-FedAvg, each client will initialize two processes: a process focusing on local model training (denoted as P_{pro}) and a process dedicating to communicating (denoted as P_{com}). It is worth noting that the central server will also spawn two processes, where its P_{pro} is used to do global model updating, and its P_{com} is used to receive clients' improved models and pushes the latest global model to the clients. Since both feed-forward and back-propagation in model training is heavily depends on the solution of its previous step (i.e., w_t

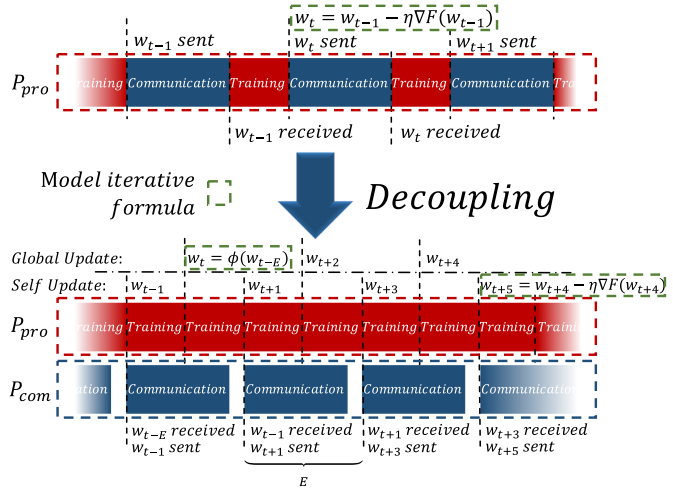

Fig. 2. Elementary SGD was used in vanilla FedAvg for model iterating, which in the meantime restricted the training phase from paralleling with the communication phase, as the training phase required the model weights that the communication phase is transmitting. To decouple the training phase and communication phase, Overlap-FedAvg loosed this restriction and employed two model iterative formulas to the training process: *self update* and *global update*, where the former one was used when the required model weights for a global model updating are not received yet, and the latter one was used when the required model weights are successfully received.

is requisite for $w_{t+1} = w_t - \eta\nabla F(w_t)$), we came to the conclusion that the model training process is strongly consecutive and real-time (i.e., chain-like). As a result, under vanilla FedAvg's setting, the model training at iteration $t+1$ will only begin after successfully receiving w_t, namely, after finishing the time-consuming communication phase at iteration t. Hence, in order to successfully decouple the model training phase from communicating, the real-time condition of vanilla FedAvg needs to be relaxed.

Algorithm 2. `Overlap-FedAvg`

Central server do:
1: Initialization: global model w_0.
2: **for** each global iteration $t \in 1, \ldots, iteration$ **do**
3: **for all** each client $k \in 1, \ldots, N$) **do in parallel**
4: # Get clients improved model.
5: $TrainLocally(k, w_t)$
6: **end for**
7: **if** p_{com} *received models* **then**
8: # Update the global model.
9: $w_{t+1} = \phi(w_t, p_0, w^0_{t-E}, p_1, w^1_{t-E}, \ldots, p_N, w^N_{t-E})$
10: **end if**
11: **end for**
TrainLocally(k, w_0):
12: **while** p_{com} *is communicating* **do**
13: # Do self update
14: $w_e \leftarrow w_{e-1} - \eta\nabla F(w_{e-1})$
15: **end while**
16: # Command p_{com} to send latest model
17: $p\_com\_send(w_E)$

The diagram of Overlap-FedAvg's model iterating process is presented in Fig. 2. In each iteration, P_{com} will fetch the model before P_{pro} has not even started training (i.e., predownload the global model), and it will also handle the model uploading task after the local model is improved, so that P_{pro} can instantly continue the next iteration of local model training without caring the matters of communicating. From the diagram, we can see that the communication interval E of Overlap-FedAvg is determined by the communication time and the training time. When P_{com} communicates, P_{pro}, like vanilla FedAvg, utilizes elementary SGD and its local training data to update the local model. On the other hand, when P_{com} finished communicating and received the latest model, unlike vanilla FedAvg that simply calculates the weighted aggregation as the new global model, Overlap-FedAvg use the function $\phi(\cdot)$ to update the global model, which will be detailedly discussed in the latter section. The pseudocode of Overlap-FedAvg is illustrated in Algorithm 2.

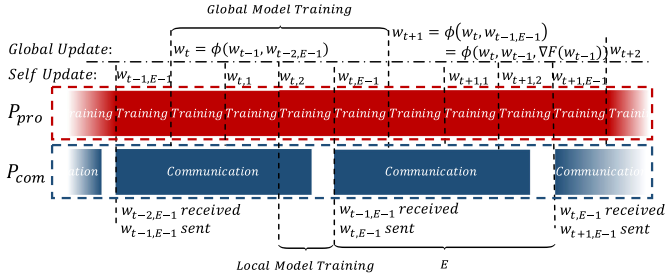

Fig. 3. Compared to Fig. 2, here the timestamp of w will only increase after performing a global update. In other words, from the central server's perspective, E iterations of clients' local model training are considered as one global model training in order to simplify the staleness analysis.

3.3 Gradients Compensation

The overlapping of the communication and training phase in the Overlap-FedAvg is promising because the communication is successfully hidden. However, the communication phase inevitably takes time, and from our observation, it is actually very likely to take much more time than a iteration of local model training. As a result, when the server/client finally received/uploaded the model, this model is probably from several iterations ago. Simply considering weighted aggregation of these stale models like FedAvg will mostly result in a degradation of model performance in Overlap-FedAvg.

To simplify the analysis of the degradation, we treat clients' E iterations of local model training as one global model training, as Fig. 3 illustrated, where $w^k_{t,i}$ is defined as the kth client's model weights at the ith local iteration in the tth global iteration. Consequently, the staleness of the model is limited to 1 global iteration (i.e., E local iterations). Then, we represent the FedAvg's updating rule in Equality (1) for the latter comparison, since it does not produce stale models at all, and is what Overlap-FedAvg pursues for

$$w_{t+1} = \underbrace{\sum_{k=0}^{N} p_k w^k_{t+1} = w_t - \eta \overbrace{\sum_{k=0}^{N} p_k \sum_{e=0}^{E-1} \nabla F(w^k_{t,e})}^{w^k_{t+1} = w_t - \mu \sum_e \nabla F(w^k_{t,e})}}_{Global\ Model\ Updating}, \tag{1}$$

However, in Overlap-FedAvg, both w^k_{t+1} and $\nabla F(w^k_{t,e})$ are not accessible when the central server updating the

global model to w_{t+1} due to the communication-computation overlap. Instead, only w_t^k can be retrieved at that specific time. Thus, in Overlap-FedAvg, the updating rule is shown in Equality (2)

$$w_{t+1} = \phi(w_t, p_0, w_t^0, p_1, w_t^1, \ldots, p_N, w_t^N), \tag{2}$$

where $\phi(\cdot)$ is the core function to produce latest global model weights, and its goal is to generate outcomes that approach results from Equality (1) at the same timestamp as much as possible. Considering w_{t-1} can be readily obtained from history, and $w_t^k = w_{t-1} - \mu \sum_e \nabla F(w_{t-1,e}^k)$, Equality (2) can be further decomposited into Equality (3)

$$w_{t+1} = \phi(w_t, \ldots, p_k, w_t^k, w_{t-1}^k, \ldots, \sum_e \nabla F(w_{t-1,e}^k), \ldots). \tag{3}$$

From the decomposition, we can see that in vanilla FedAvg, w_{t+1} is updated by w_t and each client's gradients at w_t^k (i.e., $\nabla F(w_{t,e}^k)$), which is expected. However, in Overlap-FedAvg, w_{t+1} is improved by w_t and the gradients at w_{t-1}, leading to the gradients staleness problem in Overlap-FedAvg. Hence, it is clear that our goal finally turns into estimating $\nabla F(w_t^k)$ by $\nabla F(w_{t-1}^k)$.

Inspired by the delay compensation in asynchronous distributed deep training [37], taylor series expansion and fisher information matrix were utilized to alleviate the gradients staleness problem. Specifically, by taylor series expansion, $\nabla F(w_t^k)$ can be approximately represented by $\nabla F(w_{t-1}^k)$:

$$\nabla F(w_t^k) \approx \nabla F(w_{t-1}^k) + \mathbb{H}(F(w_{t-1})) \odot (w_t - w_{t-1}), \tag{4}$$

where $\mathbb{H}(\cdot)$ is the hessian matrix of $\cdot$. However, the hessian matrix is generally very difficult to compute and is against the lightweight characteristic of federated learning. Therefore, a cheap hessian matrix approximator was introduced with fisher information matrix [38]:

$$\epsilon_t \triangleq \mathbb{E}_{y|x,w^*} ||G(w_t) - \mathbb{H}(F(w_t))|| \to 0, t \to +\infty, \tag{5}$$

where $G(w_t)$ is the outer product matrix of the gradients at w_t.

Additionally, in order to control the variance of the approximator, λ was imported. Thus, $\nabla F(w_t^k)$ can be cheaply approximated by $\nabla F(w_{t-1}^k)$ as

$$\nabla F(w_t^k) \approx \nabla F(w_{t-1}^k) + \lambda \nabla F(w_{t-1}) \odot \nabla F(w_{t-1}) \odot (w_t - w_{t-1}). \tag{6}$$

Consequently, we had an asymptotically unbiased and cheap approximator to represent the $F(w_t^k)$ from $F(w_{t-1}^k)$.

3.4 Application of NAG

Compared to local model training with pre-collected data or vanilla distributed deep learning, federated learning obtained good privacy, but lost the relatively fast convergence speed. As a result, some work was recently investigating the possibility of utilizing momentum in federated learning [39], [40] to accelerate its convergence speed. However, this work was focusing on representing the momentum with model weights w_t, which is slightly against the instinct as momentum is essentially describing the trend of gradients. Hence, their algorithms are relatively difficult to understand compared to normal momentum-based acceleration methods (e.g., NAG, Adam [21]) as momentum is superficially unrelated to the model weights.

In Overlap-FedAvg, with the decomposition of the original model updating rule and the abstraction and estimation of the latest gradients from the stale model weights, many parameters optimizing acceleration techniques can be applied to the Overlap-FedAvg intuitively and easily. Particularly, the NAG, one of the most popular approaches to speed-up the training process of the DNN, was utilized to accelerate the Overlap-FedAvg in this proposal as a example. The details and analysis are provided.

In detail, the normal NAG updating rule in SGD is illustrated as follows:

$$\begin{cases} v_{t+1} &= \beta v_t + \nabla F(w_t) + \beta(\nabla F(w_t) - \nabla F(w_{t-1})) \\ w_{t+1} &= w_t - \eta v_{t+1}. \end{cases} \tag{7}$$

Since we already had the unbiased estimation of $\nabla F(w_t)$ with $\nabla F(w_{t-1})$, the NAG updating rule in Overlap-FedAvg could be written as

$$\begin{cases} v_{t+1} &= \beta v_t + \nabla F^{ah}(w_{t-1}) \\ & \quad +\beta(\lambda \nabla F(w_{t-1}) \odot \nabla F(w_{t-1}) \odot (w_t - w_{t-1})) \\ w_{t+1} &= w_t - \eta v_{t+1}, \end{cases} \tag{8}$$

where $\nabla F^{ah}(\cdot)$ is denoted as the compensated gradients with $(\cdot)$.

3.5 The Overall Model Updating Formula

Overall, when the central server updated the global model in the Overlap-FedAvg, the following procedures were executed:

(1) Restoring the gradients of received model weights by $\nabla F(w_{t-1}) = \frac{w_{t-1} - w_t}{\eta}$. This step treated several iterations (say E iterations) of clients' local training as one global model training with the same learning rate but E times larger gradient values. This behavior was expected because we want Overlap-FedAvg to have an identical learning speed regardless of how the adaptive E variants, and it matches the chain-like characteristic of SGD. Although the gradient values are scaled up (if E is not equal to 1), the scaling factor is a fixed constant (namely E) and can be extracted with ease, and we argue that it is not a big problem to the latter transformation.
(2) Compensating $\nabla F(w_{t-1})$ to $\nabla F^{ah}(w_{t-1})$ with the formula proposed in Section 3.3.
(3) With the unbiased estimation of the gradients on the current model weights, the NAG algorithm was employed to speed up the federated learning process.

Specifically, we had the updating function $\phi(\cdot)$ shown in Algorithm 3:

Algorithm 3. $\phi(w_t, p_0, w^0_{t-E}, p_1, w^1_{t-E}, \ldots, p_N, w^N_{t-E})$

```
# Restore the gradients of the last global iteration.
for k ∈ 1, ..., N do
    ∇F(w^k_{t-1}) = (w^k_{t-1} − w_t)/η
end for
# Calculate the weighted gradients for the central server.
∇F(w_{t-1}) = Σ_{k=0}^{N} p_k ∇F(w^k_{t-1})
# Compensate the stale gradients
∇F^{ah}(w_{t-1}) = ∇F(w_{t-1}) + λ∇F(w_{t-1}) ⊙ ∇F(w_{t-1}) ⊙ (w_t − w_{t-1})
# Update the momentum and update the globla model
v_{t+1} = βv_t + ∇F^{ah}(w_{t-1}) + (∇F^{ah}(w_{t-1}) − ∇F(w_{t-1}))
w_{t+1} = w_t − ηv_{t+1}
```

4 Theoretical Analysis

In this section, we provide the theoretical analysis of the proposed Overlap-FedAvg framework. Specifically, the goal of the Overlap-FedAvg is to solve the following generic optimization problem, which is formulated as Equation (9):

$$\min_{w}\left\{F(w) \triangleq \sum_{k=1}^{N} p_k F_k(w)\right\}, \tag{9}$$

where N is the number of devices, p_k is the aggregation weight of the kth client satifying $p_k \geq 0$ and $\sum_{k=1}^{N} p_k = 1$, $F_k(\cdot)$ refers to clients' local objective function defined by Equation (10):

$$F_k(w) \triangleq \frac{1}{n_k}\sum_{i=1}^{n_k} \mathcal{L}(w; x_{k,i}), \tag{10}$$

where $x_{k,i}$ refers to the kth client's ith local training data, n_k is the size of the kth client's training data and $\mathcal{L}(\cdot)$ is the loss function.

In Overlap-FedAvg, each client will perform E iterations of local training where E is self-adaptive to the environment instead of manually configured in vanilla FedAvg, and then perform a local improved model uploading procedure to the central server for it to update the global model. Therefore, from the perspective of the central server, considering E iterations of clients' local model training as one iteration of central server's global model training (which is explained in Section 3.3), the global model's update rule can be illustrated in Equation (11)

$$\begin{aligned}\nabla F^{ah}_k(w_{t-1}) = {} & \nabla F_k(w_{t-1}) \\ & + \lambda \nabla F_k(w_{t-1}) \odot \nabla F_k(w_{t-1}) \odot (w_t - w_{t-1})\end{aligned} \tag{11}$$

$$w_{t+1} = w_t - \eta_t \sum_{k=1}^{N} p_k \nabla F^{ah}_k(w_{t-1}). \tag{12}$$

In order to present our theorem, some mild assumptions need to be made.

Assumption 4.1. *For the kth client, where $k \in range(1, N)$, F_k is L_1-smooth. That is, for all x and y*

$$F_k(y) \leq F_k(x) + (y-x)^T \nabla F_k(x) + \frac{L_1}{2}||y-x||_2^2. \tag{13}$$

Assumption 4.2 ([41], [42]). *For the kth client, where $k \in range(1, N)$, ∇F_k is L_2-smooth. That is, for all x and y*

$$\nabla F_k(y) \leq \nabla F_k(x) + (y-x)^T \mathbb{H}(F_k(x)) + \frac{L_1}{2}||y-x||_2^2, \tag{14}$$

where $\mathbb{H}(\cdot)$ is the second order derivative.

Assumption 4.3. *For the kth client, where $k \in range(1, N)$, F_k is μ-strongly convex. That is, for all x and y*

$$F_k(y) \geq F_k(x) + (y-x)^T \nabla F_k(x) + \frac{\mu}{2}||y-x||_2^2. \tag{15}$$

A recent work [37] had demonstrated the convergence rate of delay compensated asynchronous SGD under distributed deep learning's setting, and Theorem 4.1 is derived from its main theorem.

Theorem 4.1. *Let Assumptions 4.1 and 4.2 hold, if the expectation of the $||\cdot||_2^2$ norm of the ∇F^{ah}_k is upper bounded by a constant G, and the diagonalization error of $\mathbb{H}(\cdot)$ is upper bounded by ϵ_D, then the difference between the real gradients and approximated gradients at w_t is*

$$\left|\left|\nabla F_k(w_t) - \nabla F^{ah}_k(w_{t-1})\right|\right| \leq G\eta_t\left(\frac{L_1\eta_t}{2} + C_\lambda + \epsilon_t\right), \tag{16}$$

where $C_\lambda = (1-\lambda)L_1^2 + \epsilon_D$, $\epsilon_t \triangleq \mathbb{E}_{(y|x,w^*)}||\nabla F_k(w_t) - \mathbb{H}(F_k(w_t))||$ [38] and η_t is the learning rate at iteration t.

Proof.

$$\begin{aligned}&\left|\left|\nabla F_k(w_t) - \nabla F^{ah}_k(w_{t-1})\right|\right| \\ &= \underbrace{\left|\left|\nabla F_k(w_t) - \nabla F^{h}_k(w_{t-1})\right|\right|}_{A} + \underbrace{\left|\left|\nabla F^{h}_k(w_{t-1}) - \nabla F^{ah}_k(w_{t-1})\right|\right|}_{B},\end{aligned} \tag{17}$$

where $\nabla F^h_k(\cdot)$ is denoted as the first order approximation of $\nabla F_k(\cdot)$.

For the term A, as ∇F_k is L_2-smooth, we have Inequality (18) as

$$\begin{aligned}A &= \left|\left|\nabla F_k(w_t) - \nabla F^{h}_k(w_{t-1})\right|\right| \\ &\leq \frac{L_2}{2}||w_t - w_{t-1}||^2 \leq \frac{L_1 G}{2}\eta_t^2.\end{aligned} \tag{18}$$

For the term B, we have Inequality (19) as

$$\begin{aligned} B &= \left|\left|\nabla_k^h(w_{t-1}) - F_k^{ah}(w_{t-1})\right|\right| \\ &\le ||(\lambda\nabla F_k(w_{t-1}) \odot \nabla F_k(w_{t-1}) - \mathbb{H}(F(w_{t-1})))(w_t - w_{t-1})|| \\ &\le (||\lambda\nabla F_k(w_{t-1}) \odot \nabla F_k(w_{t-1}) - \nabla F_k(w_{t-1}) \odot \nabla F_k(w_{t-1})|| \\ &\quad + ||\nabla F_k(w_{t-1}) \odot \nabla F_k(w_{t-1}) - Diag(\mathbb{H}(F(w_{t-1})))|| \\ &\quad + ||Diag(\mathbb{H}(F(w_{t-1}))) - \mathbb{H}(F(w_{t-1}))||)G\eta_t \\ &\le (C_\lambda + \epsilon_t)G\eta_t, \end{aligned} \tag{19}$$

where $C_\lambda = (1-\lambda)L_1^2 + \epsilon_D$. Taking Inequality (18) and Inequality (19) into Equation (17), we have Inequality (20)

$$\left|\left|\nabla F_k(w_t) - \nabla F_k^{ah}(w_{t-1})\right|\right| \le G\eta_t\left(\frac{L_1\eta_t}{2} + C_\lambda + \epsilon_t\right), \tag{20}$$

where $C_\lambda = (1-\lambda)L_1^2 + \epsilon_D$. □

Recently another work [31] shared a theoretical analysis of the vanilla FedAvg algorithm on the Non-IID and IID dataset and gave a necessary condition for the vanilla FedAvg on Non-IID dataset to converge: η must decay. As Theorem 4.1 shown, with $\eta \to 0$, $\left|\left|\nabla F_k(w_t) - \nabla F_k^{ah}(w_{t-1})\right|\right| \to 0$ as well. Hence, intuitively Overlap-FedAvg converges as long as FedAvg converge. We will prove this intuition formally.

Another theoretical analysis of federated learning [31] defined $v_{t+1}^k = w_t^k - \eta_t\nabla F_k(w_t^k, \xi_t^k)$, $g_t = \sum_{k=1}^N p_k\nabla F_k(w_t^k, \xi_t^k)$, $\overline{v}_t = \sum_{k=1}^N p_k v_t^k$, $\overline{w}_t = \sum_{k=1}^N p_k w_t^k$ and $\overline{g}_t = \sum_{k=1}^N p_k\nabla F_k(w_t^k)$ to help abstract the problem, where ξ_t^k is a sample uniformly chosen from the kth client's local training dataset. We will inherit these notations to our analysis. Furthermore, we denote the compensated g_t as g_t^{ah} and compensated $\overline{g}_t$ as $\overline{g}_t^{ah}$. Thus, $\left|\left|g_t^{ah} - g_t\right|\right| \le \epsilon_c$

The main theorem of the literature [31] is heavily based on its Key Lemma $1-3$. Among them the Key Lemma 1 is the most important, and it is also the only lemma that will be interfered by the introduction of ϵ_c. For the simplicity, we will denote $G\eta_t(\frac{L_1\eta_t}{2} + C_\lambda + \epsilon_t)$ as ϵ_c. Consequently, we will only give a revised version of its Key Lemma 1 under Overlap-FedAvg settings.

Theorem 4.2. *Let Assumptions 4.1 to 4.3 hold. If $\eta_t \le \frac{1}{4L_1}$, we have*

$$\begin{aligned} \mathbb{E}||\overline{v}_{t+1} - w^*||^2 &\le (1-\eta_t\mu)\mathbb{E}||\overline{w}_t - w^*||^2 \\ &\quad + \eta_t^2\mathbb{E}||g_t - \overline{g}_t||^2 + 6L_1\eta_t^2\Gamma + 2\mathbb{E}\sum_{k=1}^N p_k\left|\left|\overline{w}_t - w_t^k\right|\right|^2 \\ &\quad + 2\eta_t^2\epsilon_c G + \eta_t^2\epsilon_c^2 + 2\eta_t\epsilon_c||\overline{w}_t - w^*||, \end{aligned} \tag{21}$$

where $\Gamma = F^* - \sum_{k=1}^N p_k F_k^* \ge 0$ and $\epsilon_c = G\eta_t(\frac{L_1\eta_t}{2} + C_\lambda + \epsilon_t)$

Proof. Because $\overline{v}_{t+1} = \overline{w}_t - \eta_t g_t^{ah}$, we have Equation (22) as

$$\begin{aligned} ||\overline{v}_{t+1} - w^*|| &= \left|\left|\overline{w}_t - \eta_t g_t^{ah} - w^* - \eta_t\overline{g}_t^{ah} + \eta_t\overline{g}_t^{ah}\right|\right|^2 \\ &= \underbrace{\left|\left|\overline{w}_t - w^* - \eta_t\overline{g}_t^{ah}\right|\right|^2}_{C} + \underbrace{2\eta_t < \overline{w}_t - w^* - \eta_t\overline{g}_t^{ah}, \overline{g}_t^{ah} - g_t^{ah} >}_{D} \\ &\quad + \eta_t^2\left|\left|g_t^{ah} - \overline{g}_t^{ah}\right|\right|, \end{aligned} \tag{22}$$

where w^* is the model weights we are pursing. For the second term D, because of $\mathbb{E}g_t^{ah} = \overline{g}_t^{ah}$, the expectation of D is equal to 0. For the first term C, after splitting we will have Equation (23):

$$\begin{aligned} C &= \left|\left|\overline{w}_t - w^* - \eta_t\overline{g}_t^{ah}\right|\right|^2 \\ &= ||\overline{w}_t - w^*||^2\underbrace{-2\eta_t < \overline{w}_t - w^*, \overline{g}_t^{ah} >}_{E} + \underbrace{\eta_t^2\left|\left|\overline{g}_t^{ah}\right|\right|^2}_{F}. \end{aligned} \tag{23}$$

From Assumption 4.1, it can be derived that $\left|\left|\nabla F_k(w_t^k)\right|\right| \le 2L_1(F_k w_t^k - F_k^*)$. As a result, for the term F, we have Inequality (24):

$$\begin{aligned} F = \eta_t^2\left|\left|\overline{g}_t^{ah}\right|\right|^2 &\le \eta_t^2\sum_{k=1}^N p_k\left|\left|\nabla F_k^{ah}(w_t^k)\right|\right|^2 \\ &\le \eta_t^2\sum_{k=1}^N p_k\left|\left|\nabla F_k(w_t^k) + \epsilon_c\right|\right|^2 \\ &\le \eta_t^2\sum_{k=1}^N p_k(\left|\left|\nabla F_k(w_t^k)\right|\right|^2 + 2 < \nabla F_k(w_t^k), \epsilon_c > + ||\epsilon_c||^2) \\ &\le 2L_1\eta_t^2\sum_{k=1}^N p_k(F_k(w_t^k) - F_k^*) + 2\eta_t^2\epsilon_c\sum_{k=1}^N p_k\nabla F_k(w_t^k) + \eta_t^2\epsilon_c^2 \\ &\le 2L_1\eta_t^2\sum_{k=1}^N p_k(F_k(w_t^k) - F_k^*) + 2\eta_t^2\epsilon_c G + \eta_t^2\epsilon_c^2. \end{aligned} \tag{24}$$

And thus F is successfully bounded. Next we target at bounding the second term E, which is also not too difficult as we only need to bring the disturbance ϵ_c to the equality. Formally, for the second term E, we have Inequality (25):

$$\begin{aligned} E = -2\eta_t\langle\overline{w}_t - w^*, \overline{g}_t^{ah}\rangle &= -2\eta_t\sum_{k=1}^N p_k\langle\overline{w}_t - w^*, \nabla F_k(w_t^k)\rangle \\ &\le -2\eta_t\sum_{k=1}^N p_k\langle\overline{w}_t - w_t^k, \nabla F_k(w_t^k)\rangle + 2\eta_t\sum_{k=1}^N p_k\langle w_t^k - w^*, \epsilon_c\rangle \\ &\quad + 2\eta_t\sum_{k=1}^N p_k\langle w_t^k - w^*, \nabla F_k(w_t^k)\rangle \\ &\le -2\eta_t\sum_{k=1}^N p_k\langle\overline{w}_t - w_t^k, \nabla F_k(w_t^k)\rangle + 2\eta_t\epsilon_c||\overline{w}_t - w^*|| \\ &\quad - 2\eta_t\sum_{k=1}^N p_k\langle w_t^k - w^*, \nabla F_k(w_t^k)\rangle. \end{aligned} \tag{25}$$

Here we want to eliminate all $\nabla F_k(\cdot)$. By Cauchy-Schwarz inequality, AM-GM inequality and Assumption 4.3, the inequality can be formalized as Inequality (26):

$$\begin{aligned} E &= -2\eta_t < \overline{w}_t - w^*, \overline{g}_t^{ah} > \\ &\leq \eta_t \sum_{k=1}^{N} p_k \left(\frac{1}{\eta_t} \left|\left| \overline{w}_t - w_t^k \right|\right|^2 + \eta_t \left|\left| \nabla F_k(w_t^k) \right|\right|^2 \right) \\ &\quad - 2\eta_t \sum_{k=1}^{N} p_k \left(F_k(w_t^k) - F_k(w^*) + \frac{\mu}{2} \left|\left| w_t^k - w^* \right|\right|^2 \right) \\ &\quad + 2\eta_t \epsilon_c \left|\left| \overline{w}_t - w^* \right|\right|. \end{aligned} \tag{26}$$

Putting these terms back to C and following the induction in [31], we can derive the revised version of the Key Lemma 1, which is Theorem 4.2 in this paper. □

Comparing Theorem 4.2 with the Key Lemma 1 originated in literature [31], several terms that Overlap-FedAvg introduced are strongly relevant to the learning rate η_t. Therefore, with the decaying of the learning rate during the federated training process on the IID or Non-IID dataset, these introduced terms will gradually approach nearly zero and eventually become negligible for the convergence of the learning.

5 Experiments

5.1 Experimental Setup

We evaluated the effectiveness of the proposed Overlap-FedAvg by applying it to extensive models and datasets. For the base settings, the total number of the illustrative clients in our experiments was set to 10, the max number of global epoch (a epoch usually contains thousands of iterations depending on the size of the dataset and the batch size) to train was set to 500 and all experiments were done under a Non-IID environment in order to simulate the real-world scenario. Furthermore, the code of the experiments was developed with Pytorch and Pytorch Distributed RPC[2] Framework with GLOO backend[3] from scratch. NCCL backend was not used in our experiments due to its limited support for many edge devices without Nvidia GPU. To the best of our knowledge, Overlap-FedAvg is the first attempt to parallel the training phase and communication phase in the federated learning process, and it is totally capable of coping with other compression methods. As a result, in this work, we just compared the efficiency of Overlap-FedAvg with the vanilla FedAvg [7] (i.e., the metrics of FedAvg were treated as the baseline). Particularly, the following experiments shared the same experimental configurations.

5.2 Models and Partition of Non-IID Datasets

Several datasets were chosen to investigate the efficiency of the Overlap-FedAvg, which covered two typical application fields of deep learning. Specifically, Mnist [43], Fashion-Mnist [44], EMnist [45], Cifar10 [46], and Cifar100 [46] were benchmark datasets in Image classification and Wikitext-2 [47] was the benchmark dataset in NLP. In order to meet the non-IID characteristic in real-world federated learning, we manually divided each of these datasets into ten Non-IID parts. The goal of this Non-IID partition was to make each part be different in both quantity and classes from other parts. The visualization of the partitioning is shown in Fig. 4. Some clients had plenty of training samples while some other clients only had a few, and in most clients, there are some classes that were insufficient or even absent.

Moreover, there were six benchmark models unitized to train on these datasets: Multi-Layer Perceptron (MLP), MnistNet, CNNCifar, VGG^R, ResNet^R and Transformer [48]. Among them, MLP and CNNCifar were also presented in the reference [7] to testify the effectiveness of the vanilla

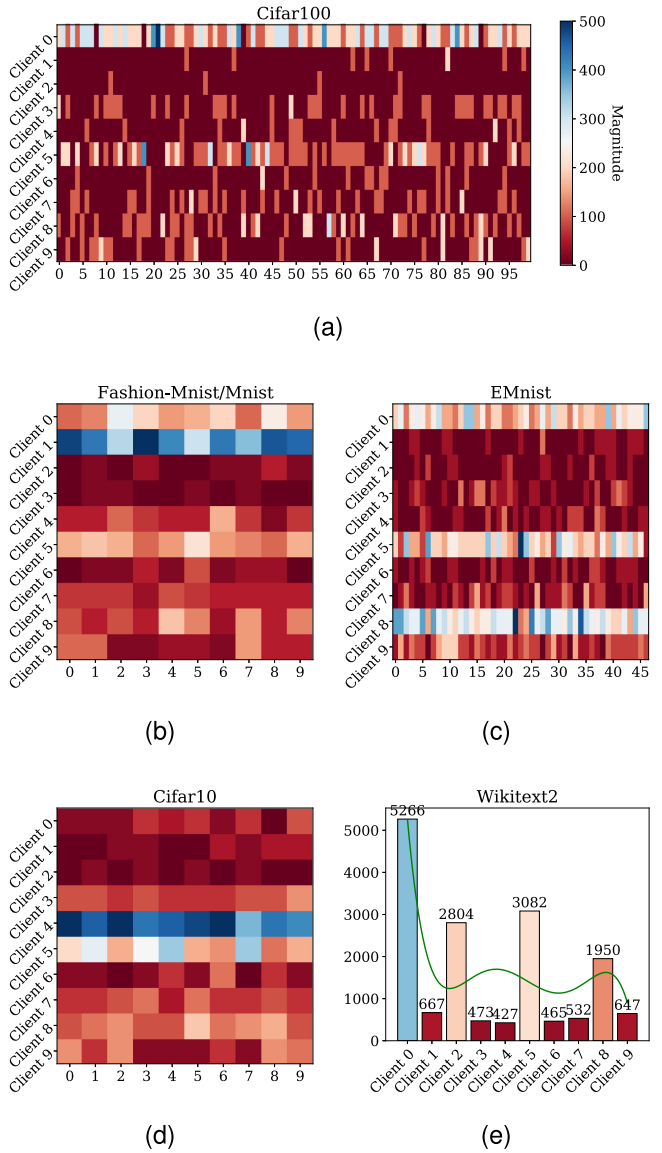

Fig. 4. Each sub-figure has two axes: a client axis and a label (class) axis, and the color of each cell or bar represents the magnitude of the assigned data. (a) The Non-IID partition of the Cifar100 dataset, which has in total 100 classes and 50000 training samples. (b) The Non-IID partition of the Fashion Mnist dataset and the Mnist dataset, which have in total 10 classes and 60000 training samples. (c) The Non-IID partition of the EMnist dataset, which has in total 47 classes and 112800 training samples. (d) The Non-IID partition of the Cifar10 dataset, which has in total 10 classes and 50000 training samples. (e) The Non-IID partition of the Wikitext2 dataset, which has over 16000 training samples.

2. https://pytorch.org/docs/stable/rpc.html
3. https://github.com/facebookincubator/gloo

TABLE 1
The Basic Configuration of the Transformer Model

Params	# of word Embeddings	# of Head	# of Hidden Units	# of Layers	Dropout Rate
Value	200	2	200	2	0.2

FedAvg, and here we utilized these two models to make a comparison between the proposed Overlap-FedAvg and the vanilla FedAvg. MnistNet, as a simplification from CNNCifar, was a standard model with only 2 convolutional layers and 2 fully connected layers without any pooling. Apart from that, the input of the MnistNet was also altered to process $1 \times 28 \times 28$ matrices instead of $3 \times 32 \times 32$ matrices. Similarly, VGG^R here was also a simplified version of the original VGG11 [5], where all dropout layer and batch normalization layer were removed and the fully connected layers' size as well as the number of convolutional filters was reduced by half. This kind of simplification was also unitized in another state-of-art work [30]. Furthermore, the ResNet^R model used in our experiments had its batch normalization layers removed. Besides, all other layers remained the same as the original ResNet18 [49]. Finally, the basic configuration of the transformer model adopted from pytorch[4] is summarized in Table 1.

5.3 Comparison of Accuracy

First, we validated the efficacy of the hierarchical computing strategy and the data compensation mechanism in Overlap-FedAvg. Detailedly, we trained MLP on Mnist, MnistNet on Fasion-Mnist, MnistNet on EMnist and CNNCifar on Cifar10 with a learning rate $\eta = 0.001$. The number of the client iteration E was set to a fixed constant 5 for vanilla FedAvg, while the upper limit number of the client iteration was set to 5 for Overlap-FedAvg. That is to say, in Overlap-FedAvg, although the communication interval is adaptive to the environment, it can not exceed 5, which limits the worst performance of Overlap-FedAvg.

The accuracy curve of these experiments is shown in Fig. 5. As we can see, when setting $\lambda = 0$ (i.e., do not compensate the stale data at all), the final accuracy of the Overlap-FedAvg was already surpassed vanilla FedAvg in some lightweight experiments (i.e., experiments with lightweight models or datasets). This phenomenon was caused by the lower synchronization interval in Overlap-FedAvg. Namely, these models were all relatively small, and when transmitting such a small model, the communication interval of 5 is large, and therefore Overlap-FedAvg adaptively lowered the communication interval to a suitable value that fitted the current network and hardware condition, which proved the adaptability of the hierarchical computing strategy. Moreover, although the communication interval was automatically reconfigured to a more appropriate value, the model parameters staleness problem still existed. Therefore, we utilized the compensation method to alleviate this problem, and the introduced hyper-parameter λ was used to adjust the compensation. As shown in Fig. 5, it is clear that a model with higher final accuracy can be obtained by fine-tuning λ.

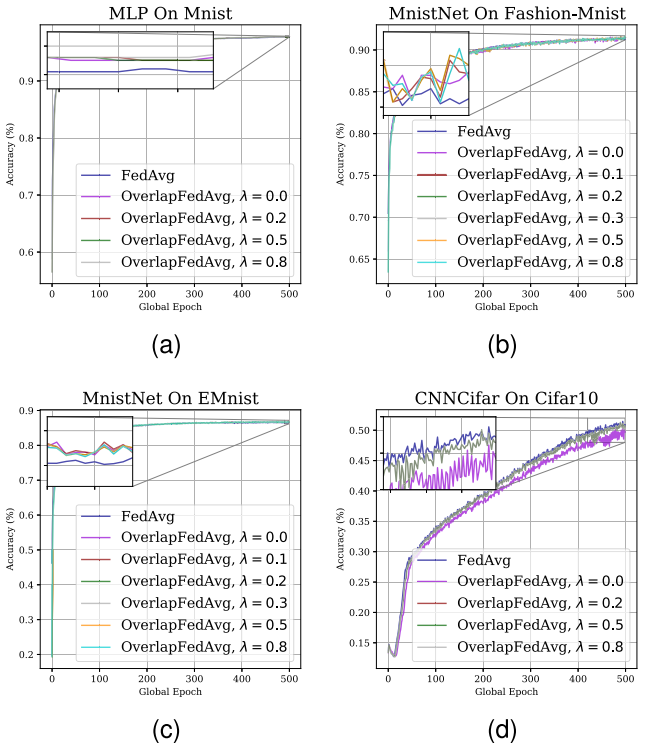

Fig. 5. (a) The accuracy curve of MLP trained on Mnist. (b) The accuracy curve of MnistNet trained on Fashion-Mnist. (c) The accuracy curve of MnistNet trained on EMnist. (d) The accuracy curve of CNNCifar trained on Cifar10.

Moreover, we expanded our experiments to some large models with complex datasets. Specifically, the VGG^R was trained on Cifar10, and ResNet^R was trained on Cifar10, Cifar100 with $\eta = 0.0001$, respectively. The results of all the experiments were described in Table 2. With the larger model size, the communication interval can not be decreased by Overlap-FedAvg for higher communication efficiency as well as better final accuracy. However, the time originally used for model uploading and model downloading can at least be saved by Overlap-FedAvg, which could be still a large amount in some cases. From the Table, we can see that when the λ was configured to 0, unlike previous lightweight experiments, Overlap-FedAvg had a lower final accuracy compared to vanilla FedAvg due to the stale model parameters problem. However, by adjusting the λ to properly compensate the data, the gap between Overlap-FedAvg and vanilla FedAvg can be greatly reduced.

Furthermore, we validated the Overlap-FedAvg's feasibility by applying it to gradients sensitive Natural Language Processing (NLP) tasks. Specifically, we utilized transformer to be trained on the wikitext-2 dataset, and used perplexity to denote the model's performance (the lower the better), as drawn in Fig. 6. With the additional assistance of the hierarchical computing strategy and the data compensation mechanism, Overlap-FedAvg with the stale problem was also capable of achieving nearly the same convergence speed with respect to the global iteration, and reaching a very similar final accuracy compared to the vanilla FedAvg.

Then, to evaluate the effectiveness of the NAG algorithm implemented in Overlap-FedAvg, we fixed $\lambda = 0.2$ and

4. https://github.com/pytorch/examples/blob/master/word_language_model/model.py

TABLE 2
Accuracy Comparison Between Vanilla FedAvg and Overlap-FedAvg With Different λ and Fixed $\beta = 0.2$

Model	dataset	η	FedAvg	Overlap-FedAvg			
			Accuracy / PPL	$\lambda = 0.0$ Accuracy / PPL	$\lambda = 0.2$ Accuracy / PPL	$\lambda = 0.5$ Accuracy / PPL	$\lambda = 0.8$ Accuracy / PPL
MLP	Mnist	0.001	0.9711	0.9776	0.9775	0.9775	0.9777
MnistNet	Fmnist	0.001	0.9130	0.9146	0.9145	0.9153	0.9143
MnistNet	EMNist	0.001	0.8661	0.8672	0.8676	0.8673	0.8676
CNNCifar	Cifar10	0.001	0.5088	0.4970	0.5060	0.5058	0.5058
VGG^R	Cifar10	0.0001	0.4321	0.4272	0.4247	0.4248	0.4247
ResNet^R	Cifar10	0.0001	0.4356	0.4341	0.4348	0.4345	0.4345
ResNet^R	Cifar100	0.0001	0.0895	0.0865	0.0866	0.0866	0.0866
Transformer	Wikitext-2	0.0001	547.067	546.966	546.921	546.920	546.920
				$\beta = 0.0$			

trained several DNN models with different β, which is illustrated in Fig. 7 and Table 3. From the Fig. 7, we can see that when the NAG was enabled, Overlap-FedAvg significantly outperformed the convergence speed of vanilla FedAvg, which strongly verified the usability of this acceleration. Consequently, considering the implementation of this acceleration was so intuitive and easy to implement, many other acceleration methods originated in SGD, be it Adam [21] or RMSProp [22], should be applied to Overlap-FedAvg with minimal effects.

5.4 Comparison of Training Speed

In this section, we compared the training speed of the Overlap-FedAvg and the vanilla FedAvg to demonstrate the communication efficiency of the Overlap-FedAvg framework. Specifically, the average wall-clock time for one iteration of federated learning in all above-mentioned experiments is presented in Fig. 8 and Table 4.

From the Fig. 8, it can be seen that in all experiments, Overlap-FedAvg successfully lowered the wall-clock time for each iteration training. Specifically, when training lightweight models (e.g., MLP, MnistNet or CNNCifar), compared to vanilla FedAvg, Overlap-FedAvg approximately saveed 10 percent of the training time. When it came to train some slightly heavier models (e.g., VGG^R, ResNet^R, Transformer), Overlap-FedAvg significantly boosted the training process by at most 40 percent. Consequently, since the size of the parameters from MLP to Transformer is increasing, which is documented in Table 4, we concluded that Overlap-FedAvg saveed more training time with heavier models.

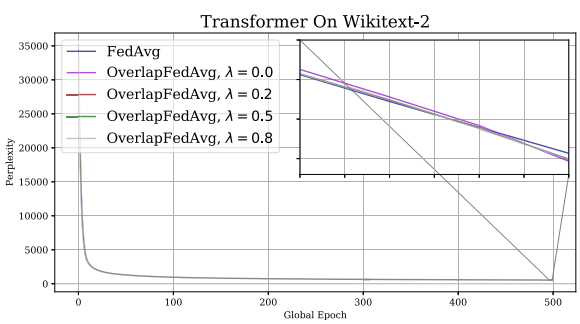

Fig. 6. The perplexity curve of Transformer trained on Wikitext-2 dataset.

We also compared the efficiency of the proposed Overlap-FedAvg from the hardware's perspective. Specifically, we recorded the utilization rate of the computation during the federated learning process, and made a visualization of it after the learning, which is shown in Fig. 9. It can be seen that Overlap-FedAvg achieveed a much higher hardware utilization rate compared to FedAvg due to the continuous local training strategy. In fact, Overlap-FedAvg almost made the best use of the computational resource as the hardware utilization rate of the Overlap-FedAvg was nearly fixed at 100 percent, which further validates the effectiveness of Overlap-FedAvg.

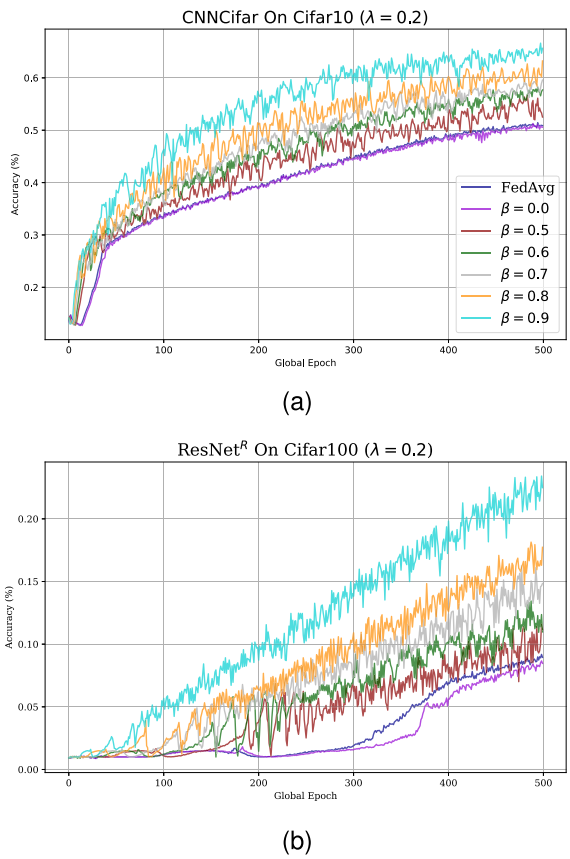

Fig. 7. (a) CNNCifar trained on Cifar10 and (b) ResNet^R trained on Cifar100. For Overlap-FedAvg, λ is fixed to 0.2. It can be seen that the NAG implemented with compensated gradients drastically boosted the convergence speed of the federated learning.

TABLE 3
Accuracy Comparison of Overlap-Fedavg With Different β and Fixed $\lambda = 0.2$

Model	dataset	η	β	Accuracy/PPL	
				50 iterations	500 iterations
MLP	Mnist	0.001	0.0	0.9317	0.9775
MLP	Mnist	0.001	0.1	0.9397	0.9793
MLP	Mnist	0.001	0.5	0.9536	0.9805
MLP	Mnist	0.001	0.8	0.9607	0.9781
MnistNet	FMNist	0.001	0.0	0.8596	0.8676
MnistNet	FMNist	0.001	0.5	0.8298	0.9145
MnistNet	Emnist	0.001	0.0	0.8134	0.8676
MnistNet	Emnist	0.001	0.1	0.7921	0.8651
MnistNet	Emnist	0.001	0.5	0.8119	0.8647
CNNCifar	Cifar10	0.001	0.0	0.2890	0.5060
CNNCifar	Cifar10	0.001	0.5	0.2984	0.5252
CNNCifar	Cifar10	0.001	0.6	0.3088	0.5769
CNNCifar	Cifar10	0.001	0.7	0.3138	0.5958
CNNCifar	Cifar10	0.001	0.8	0.3234	0.6323
CNNCifar	Cifar10	0.001	0.9	0.3853	0.6564
VGG^R	Cifar10	0.0001	0.0	0.1003	0.4247
VGG^R	Cifar10	0.0001	0.5	0.1364	0.4615
ResNet^R	Cifar10	0.0001	0.0	0.1382	0.4345
ResNet^R	Cifar10	0.0001	0.5	0.1483	0.4388
ResNet^R	Cifar10	0.0001	0.9	0.2252	0.5164
ResNet^R	Cifar100	0.0001	0.0	0.0100	0.0866
ResNet^R	Cifar100	0.0001	0.5	0.0121	0.1116
ResNet^R	Cifar100	0.0001	0.6	0.0141	0.1354
ResNet^R	Cifar100	0.0001	0.7	0.0150	0.1611
ResNet^R	Cifar100	0.0001	0.8	0.0161	0.1814
ResNet^R	Cifar100	0.0001	0.9	0.0262	0.2341
Transformer	Wikitext-2	0.001	0.0	1292.328	546.921
Transformer	Wikitext-2	0.001	0.1	1234.023	529.433
Transformer	Wikitext-2	0.001	0.5	1110.573	484.967
Transformer	Wikitext-2	0.001	0.9	1177.568	385.708

$\lambda = 0.2$

TABLE 4
The Average Wall-Clock Time for One Iteration of Federated Learning

Model	dataset	Parameters	Time / Iteration (Second)	
			FedAvg	Overlap-FedAvg
MLP	Mnist	199,210	31.2	**28.85** (↓7.53%)
MnistNet	FMnist	1,199,882	32.96	**28.31** (↓14.11%)
MnistNet	Emnist	1,199,882	47.19	**42.15** (↓10.68%)
CNNCifar	Cifar10	878,538	48.07	**45.33** (↓5.70%)
VGG^R	Cifar10	2,440,394	64.4	**49.33** (↓23.40%)
ResNet^R	Cifar10	11,169,162	156.88	**115.31** (↓26.50%)
ResNet^R	Cifar100	11,169,162	156.02	**115.3** (↓26.10%)
Transformer	Wikitext-2	13,828,478	133.19	**87.9** (↓34.0%)

6 Discussion

6.1 Privacy Security of OverLep-FedAvg

The main target for federated learning is to train the DNN model without revealing any local data to others, including the central server who coordinates the entire training process. However, some researchers played the role of attackers and tried to find ways to gain private information from the intermediate data (e.g., model weights, gradients.) generated during the training process, which was regarded as desensitized. Recently, some work [50], [51] pointed out that gradients can be utilized to inversely generate the training samples, which to some extent proved false of the previous idea that gradients are not sensitive data in deep learning. In other words, the transmission of the exposed weight in federated learning may result in the privacy leakage of the decentralized clients [52].

In vanilla FedAvg, the learning rate of each client was configured by the central server, and the local DNN weights trained by various clients were accumulated to compute the global weights on the central server. Therefore, if the central server is "curious", it is totally feasible for it to compute the clients' gradients from their weights with the known learning rate, and then extracts the hidden training data with the method proposed in references [50], [51], which is no doubt detrimental to privacy security of the federated learning. Fortunately, since the learning rate is not really involved when updating the global model in the vanilla FedAvg, this issue can be resolved by letting clients generate their own learning rate at the beginning of the training, since currently the data restoring techniques in [50], [51] was basically faking a input and label, passing them into the model, getting a fake gradients, utilizing back-propagation to calculate the loss between the real gradients from the clients and the fake gradients, and then using SGD to optimize the input and label (i.e., $\mathcal{L}(\cdot) = ||\nabla F_{fake} - \nabla F_{real}||$). Thus, by adding a unknown learning rate, the SGD has to optimize two multiplied parameters (i.e., $\mathcal{L}(\cdot) = ||\eta \nabla F_{fake} - w_{real}||$), which is theoretically impossible to optimize. In fact, we think it is

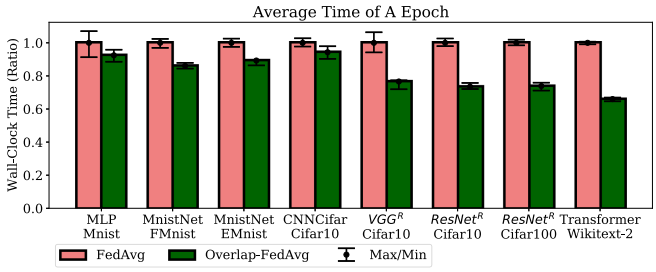

Fig. 8. The average wall-clock time for one iteration of federated learning in MLP on Mnist, MnistNet on FMnist, MnistNet on EMnist, CNNCifar on Cifar10, VGG^R on Cifar10, ResNet^R on Cifar10, ResNEt^R on Cifar100 and Transformer on Wikitext-2. We can see that the Overlap-FedAvg saved more time with the increase of the model size.

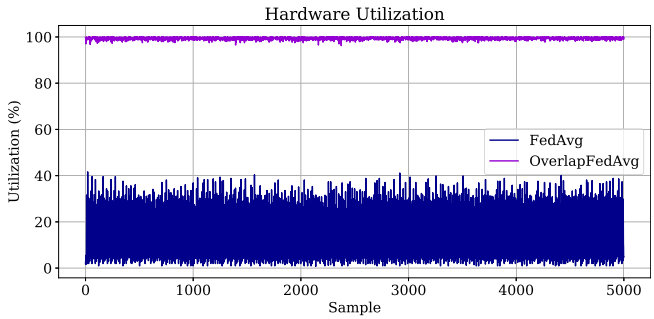

Fig. 9. The hardware utilization rate during the federated learning process. Overlap-FedAvg had a much higher utilization rate compared to vanilla FedAvg.

very crucial to make sure the clients' learning rate is not known to the central server. Therefore, as the Overlap-FedAvg needs gradients to calculate the compensation, a question emerges: will the Overlap-FedAvg work safely without the central server knowing the clients' learning rate?

Yes, it will. Overlap-FedAvg indeed needs gradients, but it does not need the *real* gradients since the learning rate is a constant that can be extracted. In other words, no matter what the real learning rate is in clients, the central server can just assume it is 1.0, computing scaled gradients and utilizing the scaled gradients to calculate the compensation. Formally, presuming the clients' learning rate is η_c, and therefore the scaled gradients $\nabla F(w_{t-1})_{scaled} = \eta_c \nabla F(w_{t-1})_{real}$. When compensating, following the algorithm we can derive the compensated gradients as Equation (27):

$$\begin{aligned}
&\nabla F(w_t)^{ah}_{scaled} = \nabla F(w_{t-1})^{ah}_{scaled} \\
&\quad + \lambda \nabla F(w_{t-1})^{ah}_{scaled} \odot \nabla F(w_{t-1})^{ah}_{scaled} \odot (w_t - w_{t-1}) \\
&= \eta_c \nabla F(w_{t-1})^{ah}_{real} \\
&\quad + \lambda \eta_c^2 \nabla F(w_{t-1})^{ah}_{real} \odot \nabla F(w_{t-1})^{ah}_{real} \odot (w_t - w_{t-1}) \\
&= \eta_c \nabla F(w_t)^{ah}_{real},
\end{aligned} \tag{27}$$

which suggests that the compensated scaled gradients is essentially equivalent to the compensated real gradients multiplied by the unknown clients' learning rate η_c, and can be directly brought into the SGD updating rule to update the global model. In summary, the privacy security of Overlap-FedAvg is guaranteed without leaking the learning rate to the central server.

7 Conclusions and Future Work

Learning from massive data stored in different locations is essential in many scenarios. To efficiently accomplish the learning, federated learning was introduced to provide a practical and privacy-preserving approach for DNN training. However, federated learning introduces massive communication overhead resulted from the heavy communication of DNN weights and low bandwidth.

In this paper, we proposed a novel federated learning framework, named Overlap-FedAvg, to achieve communication-efficient federated learning from the structural perspective. Comprehensively, Overlap-FedAvg first parallels the training phase with communication phase to allow continuous local model training without any blocking caused by the frequent communicating. Then, Overlap-FedAvg introduces a data compensation mechanism to resolve the stale data problem brought by the parallelism, and ensures the same convergence rate in comparison with the vanilla FedAvg under the same circumstance. Finally, an intuitive and easy-to-implement NAG algorithm is described, which could be coped with many other federated learning algorithms. A theoretical analysis was provided to guarantee the convergence of the Overlap-FedAvg. Extensive experiments also demonstrated that Overlap-FedAvg with only parallelism and data compensation is already capable to considerably speed up the federated learning process while maintaining nearly the same model performance (i.e., final accuracy) compared to FedAvg. Furthermore, with the NAG enabled, Overlap-FedAvg massively surpassed the vanilla FedAvg in accuracy metric concerning both the number of training iteration and the wall-clock time, which strongly validate the feasibility and effectiveness of our proposed method, especially when the model is relatively big and the network bandwidth of clients is slow or unstable.

Despite the good efficacy of Overlap-FedAvg, it inevitably introduced a new hyper-parameter λ in the data compensation mechanism to properly estimate the Hessian matrix of model weights by controlling the variance between estimated values and real values. Consequently, a comprehensive sampling of the environment is required to obtain an optimal λ, which is costly for both time and computational resources. In future work, we would like to investigate methods capable of adaptively selecting λ by evolutionary algorithms or reinforcement learning to further improve the practicability of the Overlap-FedAvg.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1002201, in part by the National Natural Science Fund for Distinguished Young Scholar under Grant 61625204, and in part by the State Key Program of the National Science Foundation of China under Grant 61836006. Yuhao Zhou and Qing Ye are co-first authors and contributed equally to this article.

References

[1] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] T. B. Brown *et al.*, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv: 1810.04805*.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[6] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.
[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[8] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 1–43, 2018.
[9] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," 2019, *arXiv: 1912.04977*.
[10] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[11] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1492–1500.
[12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[13] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv: 1610.05492*.
[14] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNS," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.
[15] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SIGNSGD: Compressed optimisation for non-convex problems," 2018, *arXiv: 1802.04434*.
[16] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.
[17] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," 2017, *arXiv: 1712.01887*.
[18] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. 32th Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1299–1309.
[19] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence," *Doklady AN SSSR*, vol. 269, pp. 543–547, 1983.
[20] Z. Yang, W. Bao, D. Yuan, N. H. Tran, and A. Y. Zomaya, "Federated learning with NESTEROV accelerated gradient momentum method," 2020, *arXiv: 2009.08716*.
[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[22] T. Tieleman and G. Hinton, (2012), *Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude* [Online]. Available: https://www.cs.toronto.edu/ hinton/coursera/lecture6/lec6.pdf
[23] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.
[24] Q. Ye, Y. Sun, J. Zhang, and J. C. Lv, "A distributed framework for EA-based NAS," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1753–1764, Jul. 2021.
[25] Q. Ye, Y. Han, Y. Sun, and J. Lv, "PSO-PS: Parameter synchronization with particle swarm optimization for distributed training of deep neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.
[26] K. Bonawitz *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
[27] S. Hardy *et al.*, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv: 1711.10677*.
[28] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[29] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv: 1806.00582*.
[30] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.
[31] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-IID data," 2019, *arXiv: 1907.02189*.
[32] Z. Qu, K. Lin, J. Kalagnanam, Z. Li, J. Zhou, and Z. Zhou, "Federated learning's blessing: FEDAVG has linear speedup," 2020, *arXiv: 2007.05690*.
[33] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, "Error feedback fixes SIGNSGD and other gradient compression schemes," 2019, *arXiv: 1901.09847*.
[34] A. Danalis, K.-Y. Kim, L. Pollock, and M. Swany, "Transformations to parallel codes for communication-computation overlap," in *Proc. ACM/IEEE Conf. Supercomput.*, 2005, pp. 58–58.
[35] M. J. Quinn and P. J. Hatcher, "On the utility of communication–computation overlap in data-parallel programs," *J. Parallel Distrib. Comput.*, vol. 33, no. 2, pp. 197–204, 1996.
[36] V. Marjanović, J. Labarta, E. Ayguadé, and M. Valero, "Overlapping communication and computation by using a hybrid MPI/SMPSS approach," in *Proc. 24th ACM Int. Conf. Supercomput.*, 2010, pp. 5–16.
[37] S. Zheng *et al.*, "Asynchronous stochastic gradient descent with delay compensation," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 4120–4129.
[38] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2001.
[39] 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.
[40] Z. Huo *et al.*, "Faster on-device training using new federated momentum algorithm," 2020, *arXiv: 2002.02090*.
[41] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2737–2745.
[42] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent converges to minimizers," 2016, *arXiv: 1602.04915*.
[43] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," ATT Labs, Atlanta, GA, USA, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist.
[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.
[45] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2921–2926.
[46] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009.
[47] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv: 1609.07843*.
[48] A. Vaswani *et al.*, "Attention is all you need," in *Proc. 31st Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[50] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14774–14784.
[51] B. Zhao, K. R. Mopuri, and H. Bilen, "IDLG: Improved deep leakage from gradients," 2020, *arXiv: 2001.02610*.
[52] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," 2020, *arXiv: 2003.02133*.

Yuhao Zhou is currently working toward the degree in computer science with the College of Computer Science, Sichuan University, China. Since 2019, he has been research training under Jiancheng Lv instructions. His main research interests include distributed machine learning, federated learning, and optimization.

Qing Ye received the BS and MS degree from the College of Computer Science, Sichuan University, China. He is currently working toward the PhD degree with the College of Computer Science, Sichuan University. His main research interests include distributed machine learning, deep learning, and neural architecture search.

Jiancheng Lv (Member, IEEE) received the PhD degree in computer science and engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2006. He was a research fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He is currently a professor with Data Intelligence and Computing Art Laboratory, College of Computer Science, Sichuan University, Chengdu, China. His research interests include neural networks, machine learning, and big data.