Towards Decentralized Deep Learning with Differential Privacy

Hsin-Pai Cheng[1⋆], Patrick Yu[2⋆], Haojing Hu[3⋆], Syed Zawad[4⋆], Feng Yan[4], Shiyu Li[5], Hai Li[1], and Yiran Chen[1]

[1] ECE Department, Duke University, Durham NC 27708, USA
{hc218,hai.li,yiran.chen}@duke.edu
[2] Monta Vista High School, Cupertino CA 95014, USA pyu592@student.fuhsd.org
[3] Beihang University of Aeronautics and Astronautics, Beijing, China
haojinghu@buaa.edu.cn
[4] CSE Department, University of Nevada, Reno NV 89557, USA {szawad,fyan}@unr.edu
[5] Tsinhua University, Beijing, China shiyu.li@duke.edu

Abstract. In distributed machine learning, while a great deal of attention has been paid on centralized systems that include a central parameter server, decentralized systems have not been fully explored. Decentralized systems have great potentials in the future practical use as they have multiple useful attributes such as less vulnerable to privacy and security issues, better scalability, and less prone to single point of bottleneck and failure. In this paper, we focus on decentralized learning systems and aim to achieve differential privacy with good convergence rate and low communication cost. To achieve this goal, we propose a new algorithm, Leader-Follower Elastic Averaging Stochastic Gradient Descent (LEASGD), driven by a novel Leader-Follower topology and differential privacy model.
We also provide a theoretical analysis of the convergence rate of LEASGD and the trade-off between the performance and privacy in the private setting. We evaluate LEASGD in real distributed testbed with poplar deep neural network models MNIST-CNN, MNIST-RNN, and CIFAR-10. Extensive experimental results show that LEASGD outperforms state-of-the-art decentralized learning algorithm DPSGD by achieving nearly 40% lower loss function within same iterations and by 30% reduction of communication cost. Moreover, it spends less differential privacy budget and has final higher accuracy result than DPSGD under private setting.

1 Introduction

With data explosion and ever-deeper neural network structures such as *VGGnet* [1] and *Resnet* [2], distributed learning systems play an increasingly important role in training large-scale models with big training data sources [3][4][5]. Training time can be greatly reduced by dividing the data set into subsets and distributing them over different workers to train the model concurrently known as *data parallelism* [6]. Many modern machine learning systems extend the data parallelism concept from data center clusters to the server-client scenario, where clients help train a global model by iterating the model over their own private data sets.

⋆ Equal Contribution

Most distributed learning systems have centralized parameter server(s) to maintain a global copy of the model and coordinate information among workers/clients. However, such system topology is vulnerable in privacy because once the central server(s) is compromised, information of the entire system can be exposed [7]. Decentralized distributed learning systems are less vulnerable to privacy as the critical information such as training data, model weights, and the states of all workers can no longer be observed or controlled through a single point of the system [8], which greatly reduces the risk of privacy leakages. Moreover, decentralized systems are more robust to problems like communication bottleneck and single point of failure compared to the centralized design. Despite all these advantages over centralized topology, decentralized systems usually perform worse in convergence rate and are known to have higher communication cost due to the multi-way communication behaviors, especially the connection across the network is relatively intricate. In addition, most of the decentralized systems still can not guarantee deferentially privacy [9][10]. There are several recent works try to solve some of the above problems of decentralized learning systems. For example, DPSGD [9] focuses on improving communication efficiency and convergence rate of decentralized learning systems. However, it is not deferentially private. [8] is the recent work that considers both decentralized design and differential privacy. However, it is based on a simple linear classification task, not a good representation of the modern neural networks, which have much more complex and deeper structures. To this end, we propose a new algorithm called *Leader-Follower Elastic Averaging Stochastic Gradient Descent* (LEASGD), which provides differential privacy with improved communication efficiency and convergence rate. To improve both the communication and training efficiency while also facilitate differential privacy preserving, we propose a novel communication protocol that is driven by a dynamic leader-follower design. Workers with temporal better learning performance and can speedup the learning of followers to improve learning efficiency. The parameters are only transferred between the worker-follower pair, which significantly reduces the communication amount. To satisfy differential privacy, we follow the approach proposed in [11] to add stochastic noise on the information transmitted. We calibrate the noise scale by analyzing the sensitivity of the updating functions in our algorithm and further demonstrate the trade-off between accuracy and privacy theoretically. LEASGD adopts the insight of the *Elastic Averaging Stochastic Gradient Descent* (EASGD) algorithm [12] by exerting the elastic force between the leader and follower at each update. Inspired by [13], we use *momentum account* to quantify the privacy budget which provides a tighter bound of privacy budget ϵ than the classical *Strong Composition Theorem* [14]. Additionally, we mathematically prove the convergence rate of LEASGD.

To conclude the comparison, we comprehensively evaluate our algorithm and compare it with the state-of-the-art approach DPSGD [9] on three main aspects: the convergence rate, the communication cost, and the privacy level. The theoretical analysis shows LEASGD converges faster than DPSGD after enough iterations. The experimental results on MNIST-CNN, MNIST-RNN, and CIFAR-10 show LEASGD achieves higher accuracy within the same iterations and within the same communication than DPSGD in the non-private setting. Also, it outperforms DPSGD by spending less privacy budget and reaching higher accuracy in the private setting.

2 Related Work

Decentralized Distributed Learning Algorithm. Different from centralized distributed learning algorithms, decentralized algorithms do not need any central entity. Every workers maintains its own copy of the training model and directly communicates with other workers. Many decentralized algorithms aim at solving distributed consensus problems [15][16][17], which is also the goal of our paper. Lian *et al.* [9] proposed DPSGD, which outperforms centralized algorithms by achieving the same convergence rate with lower communication complexity. All the decentralized algorithms mentioned above depend on a double stochastic matrix to organize the communication, which is hard to obtain and tune in real-life applications and they also do not implement their algorithms in a privacy-preseving setting. Moreover, adversarial attackers who reveal this matrix can easily obtain the model information by doing a simple linear combination, and even worse, to reveal the private data. Yan *et al.* [10] developed a scheme to prevent such attack, but the scheme depends on a specific communication topology. Our privacy-preserving scheme is not limited by any specific communication topology and we coordinate the communication in a random manner to reduce vulnerability.

EASGD. EASGD is first proposed by Zhang *et al.* [12] to solve the distributed consensus optimization problem. The basic idea is to let multiple workers pull a single master to the global optimum and the movement of model parameters is proportional to the distance between workers and master, which is so-called elastic force. They show EASGD can achieve final better accuracy than the DOWNPOUR [18] and other parallel algorithms and converge even faster in its momentum version EASGD. However, the communication topology in EASGD highly resembles a centralized one. The master in its algorithm plays a similar role as the central node. Moreover, they did not consider any privacy constraints in their algorithm.

Differential Privacy in Machine Learning *Differential Privacy* (DP), first proposed by Dwork, provides a specific quantitative method to measure and protect privacy [11]. It was then studied and applied to fields in *Machine Learning*. For instance, Kairouz *et al.* [19] studied utility and privacy trade-off of different mechanisms under local DP setting. Abadi *et al.* [13] proposed differential privacy SGD and suggested the momentum account method to compute a much tighter bound of ϵ, δ privacy budget (which we also adopt in this paper) than classical strong composition theorem [14]. For distributed machine learning, Bellet *et al.* [8] proposed a completely decentralized and asynchronous algorithm to solve personalized optimization problem and also use DP in their privacy-preserving method. However, they did not compare with other parallel algorithms in the private setting and their experiment only focused on simple linear classification task and light-weight dataset, MovieLens-100K, rather than a more complex data set such as CIFAR-10 on which we conduct the experiment for our method. Furthermore, the model in their experiment is a simple p-dimensional vector. It is not clear whether their method could be applied for deeper neural networks, which we demonstrate in our experiments.

In [20], the basic idea of LEASGD is outlined and some preliminary analysis and evaluation results are present. The current extended version of this paper provides a more detailed description of the LEASGD approach, as well as a more comprehensive theoretical analysis and experimental evaluation.

3 Non-private Leader-Follower Elastic Averaging Stochastic Gradient Descent Algorithm

In this section, we introduce LEASGD in a non-private setting. The proposed algorithm includes both the model update at each worker and the communication protocol among workers. We start with the synchronous version (Algorithm 1) and then extend it to the asynchronous version (Algorithm 2), which is more commonly used in practice.

3.1 Problem Setting

Without loss of generality, we follow the problem setting of distributed decentralized autonomous learning in [10]. We assume there are m workers each with a set of local data S_i and $i \in \{1, 2.....m\}$, which can only be accessed locally by worker i. To solve consensus problem, we assume all data sets S_i are homogeneous but can have different data distributions. Along the training process, each worker i computes a parameter vector w_t^i at each iteration t to represent the learning outcomes and then computes the corresponding loss function $f_t^i(w^i) = l(w_t^i, x_t^i, y_t^i)$ with the input x_t^i and given labels y_t^i. It is worth noting that fed data $\{x_t^i, y_t^i\} \subseteq S_i$. After learning from the data, each worker has two ways to contribute to the global learning progress: 1) Update its model parameters by local gradient descent; 2) Communicate with other workers to update each other's model parameters. To reduce the communication cost, communication is not always required at the end of each iteration. We define a communication interval τ to represent how many iterations between each update in our learning algorithm. When the training process is done, each worker has its own variation of the same model (i.e., performing the same task but with different trained model parameters w^i). This is quite different compared to the personalized distributed learning [8], where different workers have completely different models to solve personalized problems. It is also different from [9], where all workers have the same version of the model.

Given each worker has its own local version of the model, it is necessary to assemble all local models by averaging the loss function. We formulate it as an optimization problem as follows:

$$w^* = \{w^1, ..., w^m\}, \tag{1}$$

$$\underset{w^i \in \Omega}{argmin}\, \overline{F}(w, T) = \frac{1}{m} \sum_{i=1}^{m} f_T^i(w^i), s.t. \Omega \subseteq \mathbb{R}^n \, and \, T \in \mathbb{R}, \tag{2}$$

where T presents the predefined number of iterations in τ.

3.2 Decentralized Leader-Follower Topology

To support the decentralized design, we categorize all workers into two worker pools - leader pool with workers of higher loss function values and follower pool with workers of lower loss function values, see Figure 1(a). The core idea is to let followers to pull so that better performing can guide the followers in the right direction to improve the learning. Specifically, we use an elastic updating rule to regulate the learning updates in each leader-follower pair as follows:

Algorithm 1: Synchronous Follower Elastic Averaging Stochastic Gradient Descent Gradient Descent

Require: number of workers m, number of followers L, categorization interval k, communication interval τ, elastic factor ρ, learning rate η

for $t = 1, 2, ..., T$ **do**
  if $t \; mod \; k\tau \; is \; 0$ **then**
    if $t \; = \; 0$ **then**
      randomly select L followers $l_1, ..., l_L$
    else
      sort m based on loss f and select top L
      $l_1, ..., l_L = \underset{i=1,...,m}{argmax} L \; f_t^i(\omega^i)$
    end if
  end if
  if $t \; mod \; \tau \; is \; 0$ **then**
    for workers: $i \in \{l_1, ..., l_L\}$ **do**
      local SGD updating
    end for
    for workers: $i \notin \{l_1, ..., l_L\}$ **do**
      randomly select a follower f from follower pool
      transmit parameter vector with f and do elastic updating
    end for
  else
    for all workers: $i \in \{1, ..., m\}$ **do**
      local SGD updating
    end for
  end if
end for

$$w_{t+1}^i = w_t^i - \eta g_t^i + \eta\rho(w_t^f - w_t^i) \quad and \quad w_{t+1}^f = w_t^f - \eta g_t^f + \eta\rho(w_t^i - w_t^f). \quad (3)$$

We use i to denote a leader; f is a follower; k is the categorization interval; ρ is elastic factor; g is gradient; and η is learning rate. Given learning is a dynamic process, the two worker pools are dynamically updated based on the learning progress. The pools are recategorized each $k\tau$ time interval. This protocol enables the convergence rate of our algorithm to have a limited upper bound, which will be discussed in detail in Section 5. To avoid over-fitting to one worker's model during the training process, we add the L2-normalization on the training loss function:

$$f_t^i(w^i) = l(w_t^i, x_t^i, y_t^i) + \lambda \parallel w^i \parallel_2 . \quad (4)$$

We also randomly pair the leaders and followers after each learning updates to avoid one follower's model having excessive influence on others. This randomization mechanism also benefits the privacy-preserving as randomized communication can confuse the attacker and make it more difficult to trace the information source.

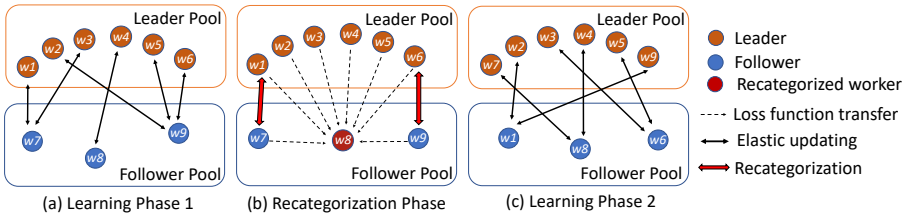


Fig. 1: The dynamic Leader-Follower topology. (a) shows the structure of leader pool and follower pool. (b) shows a recategorization phase where one of the workers is elected as recategorized worker and gathers the latest loss function values from all other workers. (c) shows the new structure of leader pool and follower pool after recategorization (note the randomization used for avoiding over-fitting).

3.3 Algorithm Hyperparameters

Next, we discuss in details of the hyperparameters in our algorithm.
Elastic factor ρ : This hyperparameter adjusts the exploration and exploitation trade-off in our learning algorithm. A large ρ represents more exploitation, which leads to a faster convergence rate at the beginning of training process especially in a convex case. However, workers can fall into a local optimum easily because the tight relations of the follower prevent them from further exploring the parameter space and this may compromise the final accuracy. On the the other hand, a rather small ρ leans towards exploration which could avoid this to some extent, but it also causes a much slower convergence rate and thus compromises the accuracy within the predetermined number of iterations. Selecting a well-balanced ρ is therefore important for the final accuracy.

Number of Followers L : Similar to ρ, this hyperparameter reflects learning aggressiveness. Small L enables more to guide followers to improve the convergence rate, the formal analysis is provided in Section 5.1. However, if L is too small, the communication between and followers would increase. More importantly, workers with bad learning progress can be incorrectly categorized as leaders and thus could not receive help from workers with good learning progress, which can eventually result in worse final accuracy.
Categorization Interval k: It represents how frequently we recategorize and followers. A rather frequent categorization can ensure that and followers are timely identified. However, each categorization comes with a communication cost as the loss function values need to be collected and so a smaller k means higher communication cost. Our key insight here is that k can be set smaller at the beginning of the training process as workers' local models usually change more dramatically during that time and have more variance between them. As the training progresses, the loss function values tend to be more stable and we can decrease the recategorization frequency to reduce the communication overhead.

3.4 Asynchronous LEASGD Algorithm

Algorithm 2: Asynchronous Follower Elastic Averaging Stochastic Gradient Descent Algorithm

1: **Require:** all workers have the same follower list F, this worker index w, current iteration t_i, communication interval τ, categorization interval k
2:
3: **if** w is a follower **then**
4: compute local SGD
5: **else**
6: **if** $t_i \; mod \; \tau \; is \; 0$ **then**
7: randomly select follower $f \in F$
8: transmit parameter vector to f and do elastic updating
9: **else**
10: compute local SGD
11: **end if**
12: **if** $t_i \; mod \; k\tau \; is \; 0$ **then**
13: **for** all $f \in F$ **do**
14: get f's loss function value l_f
15: get w's loss function value l_w
16: **if** find $l_f < l_w$ **then**
17: recategorize all workers' follower pool by replacing f with w
18: **break**
19: **end if**
20: **end for**
21: **end if**
22: **end if**

Next, we introduce LEASGD in the asynchronous manner as shown in Algorithm 2. To make our learning system fully asynchronous and decentralized, there is no global clock to coordinate all workers nor a global supervisor to master all workers. We set the number of wake up iterations as t_i for different workers according to a Poisson Stochastic Process with different arrival rate λ_i based on local clock time t, that is $P[t_i(t + \triangle t) - t_i(t) = k] = \frac{e^{-\lambda_i \triangle t}(\lambda_i \triangle t)^k}{k!}$. The larger the λ_i, the more frequently the worker updates its model. Moreover, the worker pools also updated asynchronously as shown in Line 17-24 in Algorithm 2. Such mechanism may not guarantee all the leaders and followers are identified timely, but it reduces the communication cost of recategorization.

Note that the stochastic gradient descent in the algorithm can be replaced by other gradient descent optimization methods such as the mini-batch gradient descent without affecting the theoretical results.

4 Private-preserving Scheme

In this section, we consider the privacy-preserving setting of the proposed algorithm. We first explain the notion of differential privacy, which serves as the theoretical foundation for our scheme. We then introduce our privacy-preserving scheme and the privacy budget ϵ spent along the iterations.

4.1 Differential Privacy Model

Despite there is no direct exchange of raw data in the communication, the risk of leaking the data still exists when we transmit the parameter vectors. If the two consecutive transmitted vectors w_t^i and w_{t+1}^i are from same worker i, the attacker can easily derive the gradient by subtracting one vector from the other and due to the gradient g_t^i is proportional to the raw data x_t^i. More details about this type of attack can be referred to [10].

There are several methods proposed to preserve the privacy of distributed learning systems and prevent eavesdropping in the communication network. For example, Yan *et.al.* proposed a series of communication topologies to prevent the sensitive message from leaking to malicious workers [10]. In this work, we rely on a more general and flexible theory that does not depend on specific topology specified in the differential privacy theory in [11]. The basic property of differential privacy mechanism is that under a little perturbation of the input of the algorithm, the change of its output's probability distribution is within a limited bound. The specific definition is as below:

Definition 1 *A randomized mechanism $\mathcal{M}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satises (ϵ, δ)-differential privacy if for any two adjacent inputs D, $D' \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ it holds that*

$$Pr(\mathcal{M}(D) \in \mathcal{S}) \leq e^{\epsilon} Pr(\mathcal{M}(D') + \delta. \tag{5}$$

When differential privacy applied in the machine learning, adjacent inputs D, D' refer two datasets that are only different at one training sample and the randomized mechanism $\mathcal{M}$ is the training update algorithm. In our setting we use elastic updating rule in Eq. 3. The factor ϵ in Eq. 5 denotes the privacy upper bound to measure an algorithm and δ denotes the probability of breaking this bound. The δ is generally set smaller than the reciprocal of the number of samples in the local data set S_i, which ensures that none of the samples can be revealed by the differential private attack.

4.2 Privacy-preserving Scheme

The general idea to preserve differential privacy is to add noise on the output of the algorithm and the noise scale is based on the sensitivity of the output function as defined in [11].

Definition 2 *- For $f : \mathcal{D} \rightarrow \mathcal{R}^d$, the $L2 - sensitivity$ of f:*

$$\triangle f = \max_{\mathcal{D}, \mathcal{D}_2} \parallel f(\mathcal{D}_1) - f(\mathcal{D}_2) \parallel_2 \tag{6}$$

for all $\mathcal{D}_1$, $\mathcal{D}_2$ differing in at most one element.

For different input data, Eq. 3 only differs in the gradient g_t^i part. In other words, the sensitivity of the updating rule of LEAGSD is same as the gradient g_t^i. Thus we use the similar scheme as the differentially private SGD algorithm in [13]. To limit the sensitivity of gradient, we clip the gradient into a constant C. The clipped gradient $\overline{g}_t^i = g_t^i / max(1, \frac{\|g_t^i\|_2}{C})$. Then, we add Gaussian noise on the clipped gradient

$$\tilde{g}_t^i = \overline{g}_t^i + \mathcal{N}(0, \sigma_2^2 C^2). \tag{7}$$

By using $\tilde{g}_t^i$ to replace g_t^i in equation 3, we obtain the differential-privacy preserving scheme of LEASGD as:

$$\tilde{w}_{t+1}^i = \tilde{w}_t^i - \eta\tilde{g}_t^i + \eta\rho(\tilde{w}_t^j - \tilde{w}_t^j) \quad and \quad \tilde{w}_{t+1}^j = \tilde{w}_t^i - \eta\tilde{g}_t^i + \eta\rho(\tilde{w}_t^j - \tilde{w}_t^i). \quad (8)$$

When we choose the variance of Gaussian noise $\sigma_2 = \frac{\sqrt{2ln(1.25/\delta)}}{\epsilon}$, we ensure that each communication step of LEAGSD is (ϵ, δ)-DP. Using the property of DP-mechanism in [14], the composition of a series of DP-mechanisms remains DP, which guarantees that for each worker i, its training algorithm $\mathcal{M}_i$ at each iteration is DP. To compute the total (ϵ, δ), we don't use the strong composition theory [14]. Instead, we use the momentum account method [13], which provides a much tighter bound of ϵ to evaluate the privacy-preserving performance of our algorithm.

5 Analysis

5.1 Convergence Rate Analysis

In this section, we conduct a convergence rate analysis for Synchronous LEASGD in a strongly-convex case and also compare it with the DPSGD theoretically. Before we show the result of the convergence rate, we first introduce some assumptions held in the analysis.

Assumption 1 *These assumptions are held throughout the analysis:*
1. ***i.i.d. Assumption****: We divide our system into several sub-systems with only 1 follower and p leaders. And all the variables in these sub-systems are i.i.d.*
2. ***Correct Categorization****: Assume that since the categorization step in Algorithm 1 is implemented, the identity of all workers will not change until the next categorization*
3. ***Bounded Stochastic Gradient****: Assume that the variance of all the local gradients is bounded for any w for any workers from $1, ..., m$ and input x_t^i. There exist constant σ_1 such that*

$$E[g_t^i - \bigtriangledown f(w_t^i)] = 0 \quad and \quad E[\parallel g_t^i - \bigtriangledown f(w_t^i) \parallel^2] \leq \sigma_1^2. \quad (9)$$

4. ***Strongly Convex Condition****: We focus on the strongly-convex case in this analysis. Correspondingly, there exists $0 < \mu \leq L$ for all the loss functions described in Eq. 4, we have:*

$$\mu \parallel w_i - w_j \parallel^2 \leq \langle \bigtriangledown f(w_i), \bigtriangledown f(w_j) \rangle \leq L \parallel w_i - w_j \parallel^2 . \quad (10)$$

We define that

$$d_t = \frac{E\sum_{i=1}^{p} \parallel w_t^i - w^* \parallel^2 + E \parallel w_t^f - w^* \parallel^2}{p+1}. \quad (11)$$

Proposition 1 *(Convergence rate of Algorithm 1) If $0 \leq \eta \leq \frac{2(1-\beta)}{\mu+L}$ and $0 \leq \alpha = \eta\rho < 1$,$0 \leq \beta = p\alpha < 1$, then we obtain the convergence of d_t as follows:*

$$d_t \leq h^t d_0 + (c_0 - \frac{\eta^2\sigma_1^2}{\gamma})(1-\gamma)^t(1-(\frac{p}{p+1})^t) + \eta^2\sigma_1^2\frac{1-h^t}{\gamma},$$
$$where\ 0 < h = \frac{p(1-\gamma)}{p+1} < 1, k = \frac{1-\gamma}{p+1}, \gamma = 2\eta\frac{\mu L}{\mu+L}, \tag{12}$$
$$and c_0 = \max_{i=1,\ldots,p,f} \parallel w_0^i - w^* \parallel^2 .$$

Under the Assumption 1.1, we simplify our system by dividing it into several subsystems and each includes only 1 follower and p leaders. We assume that the convergence rate of each subsystem is same as that of the whole system. To obtain this convergence rate, we rely on the following theorem.

Theorem 1 *Let* $y_t = \frac{1}{p}\sum_{i=1}^p w_t^i$, $a_t = E \parallel y_t - w^* \parallel^2$, $b_t = \frac{1}{p}\sum_{i=1}^p E \parallel w_t^i - w^* \parallel^2$, $c_t = E \parallel w_t^f - w^* \parallel^2$, $\alpha = \eta\rho$, $\beta = p\alpha$,$\gamma = 2\eta\frac{\mu L}{\mu+L}$. *If* $0 \leq \eta \leq \frac{2(1-\beta)}{\mu+L}$,$0 \leq \alpha < 1$,$0 \leq \beta < 1$, *then*

$$b_{t+1} \leq (1-\gamma-\alpha)b_t + \alpha c_t + \eta^2\sigma_1^2, \tag{13}$$
$$c_{t+1} \leq (1-\gamma-\beta)c_t + \beta a_t + \eta^2\sigma_1^2. \tag{14}$$

Proof of Proposition 1: from the sorting rule and the Assumption 1.1, we can easily obtain the inequality relation of a_t,b_t,c_t and d_t

$$a_t \leq d_t \leq c_t \quad and \quad b_t \leq d_t \leq c_t \tag{15}$$

Applying (15) in (14) and iterating through t times, we have

$$c_t \leq (1-\gamma)^t c_0 + \eta^2\sigma_1^2\frac{1-(1-\gamma)^t}{\gamma}. \tag{16}$$

Now replacing $d_t = \frac{pb_t+c_t}{p+1}$ in (13), (14), we have

$$d_t \leq hd_{t-1} + kc_{t-1} + \eta^2\sigma_1^2. \tag{17}$$

Applying (15) in (16), we have:

$$d_t \leq hd_{t-1} + k(c_0 - \frac{\eta^2\sigma_1^2}{\gamma})(1-\gamma)^{t-1} + (1+\frac{k}{\gamma})\eta^2\sigma_1^2. \tag{18}$$

Iterating t times though this inequality, we have

$$d_t \leq h^t d_0 + k(c_0 - \frac{\eta^2\sigma_1^2}{\gamma})\frac{(1-\gamma)^t - h^t}{1-\gamma-h} + (1+\frac{k}{\gamma})\eta^2\sigma_1^2\frac{1-h^t}{1-h}. \tag{19}$$

To simplify (19), we note that

$$k + h = 1 - \gamma, \tag{20}$$

$$\frac{1+\frac{k}{\gamma}}{1-h} = \frac{1+\frac{\frac{1-\gamma}{p+1}}{\gamma}}{1-\frac{p(1-\gamma)}{p+1}} = \frac{p+1+\frac{1-\gamma}{\gamma}}{p+1-p(1-\gamma)} = \frac{p+\frac{1}{\gamma}}{1+p\gamma} = \frac{1}{\gamma}. \tag{21}$$

So (19) can be rewritten as

$$d_t \leq h^t d_0 + (c_0 - \frac{\eta^2\sigma_1^2}{\gamma})(1-\gamma)^t(1-(\frac{p}{p+1})^t) + \eta^2\sigma_1^2\frac{1-h^t}{\gamma}. \tag{22}$$

This concludes the proof.

Under this proposition, it implies that the average gap between all workers and optimum in a subsystem includes three parts, which could also be applied to the whole system based on Assumption 1.1. The first part is a shrinkage part of itself by a constant factor $0 < h < 1$, which shows a exponential decreasing relationship between the gap and iteration t. The second part is tend to be 0 with the increase of t. The third part is a inherent noisy part with the variance of $\eta^2\sigma_1^2$. If we ignore the influence of the inherent noise on the gradient and extend $t \to \infty$, we can obtain an purely exponential decline of the gap, that is $E[d_{t+1}] \leq hE[d_t]$. Note that the shrink factor h is negatively correlated to p, which implies that, when our system is operating in a strongly convex setting, the larger worker scale can correspondingly result in a faster convergence rate of the system, which is in line with the hyperparameter analysis above.

According to the convergence rate analysis in DPSGD [9], its convergence rate is $O(1/[(p+1)t])$ with our denotation in the strongly-convex setting. Compared with our $O(h^t)$ rate, the convergence rate of DPSGD is relatively slower when we extend the $t \to \infty$.

5.2 Privacy Trade-off Analysis

In this section, we provide the trade-off analysis between the accuracy and privacy. Following the convergence rate analysis above, we can obtain the modified convergence rate of d_t after adding the extra noise as the only part that needs to be changed is the third part of equation 12. In the private setting, the noise is composed of two different parts. First, the inherent noise, which is the same as defined in Assumption 1.3. Second, the differential-privacy preserving noise, which is defined in the equation 7.

We assume that the two noise is independent of each other. Thus, the variance of the composed noise is the sum of the two independent noise variances and it satisfies $\sigma^2 < \sigma_1^2 + C^2\sigma_2^2$. Finally, the convergence rate in the private setting is as below:

Proposition 2 *(Trade-off for privacy) With the limits held in the Proposition 1, we can obtain the convergence of d_t after adding the Gaussian Noise:*

$$\begin{aligned} d_t &\leq h^t d_0 + (c_0 - \frac{\eta^2\sigma^2}{\gamma})(1-\gamma)^t[1-(\frac{p}{p+1})^t] + \eta^2\sigma^2\frac{1-h^t}{\gamma} \\ &< h^t d_0 + (c_0 - \frac{\eta^2\sigma^2}{\gamma})(1-\gamma)^t[1-(\frac{p}{p+1})^t] + \eta^2\sigma_1^2\frac{1-h^t}{\gamma} + \eta^2C^2\sigma_2^2\frac{1-h^t}{\gamma}. \end{aligned} \tag{23}$$

According to this proposition, the extra trade-off for privacy is $\eta^2C^2\sigma_2^2\frac{1-h^t}{\gamma}$ and when $t \to \infty$, this trade-off can be formulated as $\frac{\eta^2C^2\sigma_2^2}{\gamma}$. Note that this trade-off remains the same when p grows, which implies that our algorithm has stable scalability when applied in the private setting. In other words, despite the group of workers become larger, the noise in the communication is not accumulated to further compromise the

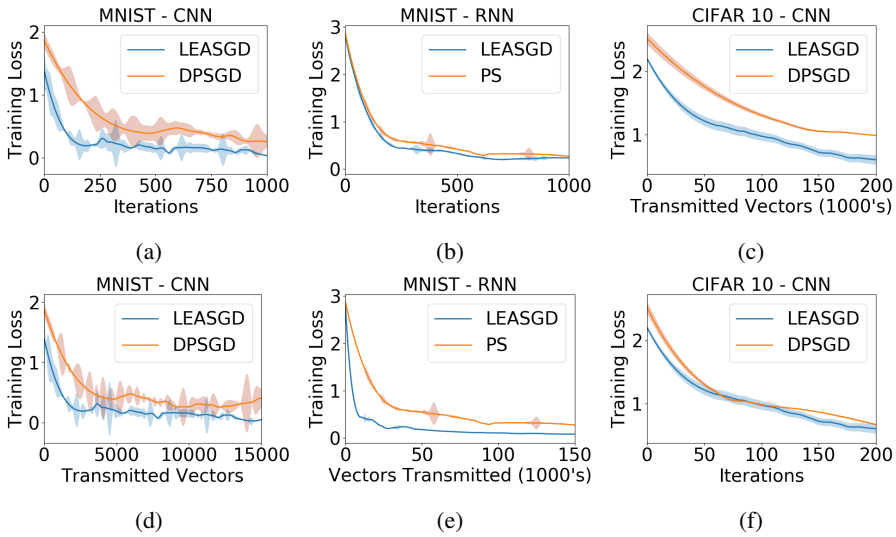


Fig. 2: Comparison between LEASGD, DPSGD, and PS. Training Loss VS Iterations for (a) MNIST-CNN, (b) MNIST-RNN, and (c) CIFAR 10. Training Loss VS Number of Transmitted vectors for (d) MNIST-CNN, (e) MNIST-RNN, and (f) CIFAR 10.

performance. Additionally, this analysis is under the assumption of strongly-convex setting. In fact, when applied in a non-convex setting, the adding noise in an appropriate scale could improve the performance as the noise motivates workers to explore more space and it becomes easier for those workers in the local optimum to get out of it.

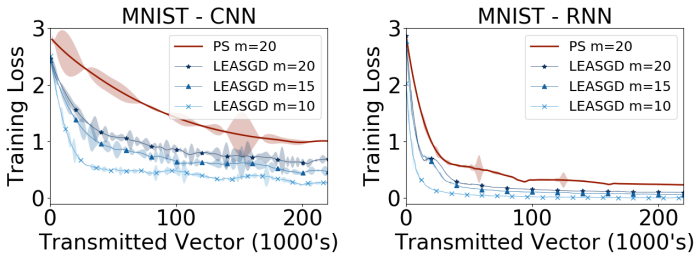
6 Experimental Evaluation

6.1 Experiment Setup

Non-private setting setup. In this setting, we perform experiments using MNIST-CNN, MNIST-RNN, and CIFAR-10. The structure of MNIST-CNN is a 3-layer Multi-layer Perceptron (MLP), and the MNIST-RNN has 32 hidden layers. For CIFAR-10, we use the ConvnetJS [21] as the model.
Private setting setup. In this setting, the neural network models and hyper-parameters are the same as in non-private setting. We test two algorithms of different worker sizes of $m = 5, 15$, respectively shown in Table 1 . Additionally, we tune the elastic factor $\rho = 10$ for MNIST-CNN and $\rho = 100$ for CIFAR-10. For the private factors, we set Gaussian noise variance $\sigma = 4$, clipping constant $C = 4$, and fixed δ budget $\delta = 10^{-5}$.

6.2 Non-private Setting Comparison

Accuracy Comparison. We first conduct experiments on a cluster of 15 nodes to compare our LEASGD against DPSGD in the non-private setting for MNIST-CNN and



(a) (b)

Fig. 3: Scalability of LEASGD (m is the total number of nodes).

CIFAR 10 models. For the MNIST-RNN model, we compare against Parameter Server (PS) as DPSGD does not support RNN models.

DPSGD has various communication topologies when communication matrix is set differently. We only test the ring topology for simplicity, which allows each worker to accept the information from two other neighbor workers. The training loss as a function of iterations is shown in Figure 2 (a)(b)(c), which demonstrate that LEASGD converges faster than DPSGD and PS at the beginning of the training process and also achieves a lower loss function at the end across all models.

Communication Cost. To quantitatively evaluate the communication cost, we track the average training loss of all workers with regard to the number of transmitted vectors as shown in Figure 2 (d)(e)(f). The results verify that LEASGD also outperforms DPSGD and PS in the efficiency of communication usage as within the same number of transmitted vectors, the loss function of LEASGD almost always under that of DPSGD and PS across all models.

Theoretically, in each iteration, LEASGD has less transmitted vectors compared to DPSGD and PS even though communication in LEASGD is two-way instead of one-way in DPSGD. In LEASGD, the number of transmitted vectors per iteration is $(m - L) * 2$ (14 in this experiment). In DPSGD, the number of transmitted vectors per iteration is the sum of numbers of neighbors of all workers, which is typically $2m$ (20 in this experiment) in a ring network.

Scalability. We also evaluate the scalability of LEASGD using MNIST-CNN and MNIST-RNN by varying the number of total nodes from 10 to 20 while keeping the percentage of leaders at 75%. The results are organized in Figure 3, where we can see LEASGD outperforms PS across all cases. We also observe that the training loss becomes slightly worsen when the number of nodes increases, but the degrading speed becomes slower, i.e., the gap between 20 nodes and 15 nodes are much smaller than the gap between 15 nodes and 10 nodes. This indicates that LEASGD scales well.

6.3 Differential Private Comparison

In the private setting, we use the *momentum account* to compute the totally spent ϵ and the adding noise scales are the same for two algorithms. As shown in Table 1, LEASGD achieves better accuracy with less ϵ than DPSGD. More importantly, the final accuracy of our algorithm does not vary greatly when the worker scale m increases.

MNIST - CNN		
Algorithm	Final accuracy	Total ϵ
LEASGD (m=5)	**0.97**	**4.183**
DPSGD (m=5)	0.97	4.505
LEASGD (m=15)	**0.97**	**4.651**
DPSGD (m=15)	0.95	4.843

CIFAR-10 - CNN		
Algorithm	Final accuracy	Total ϵ
LEASGD (m=5)	**0.74**	**4.655**
DPSGD (m=5)	0.71	4.925
LEASGD (m=15)	**0.72**	**4.116**
DPSGD (m=15)	0.68	4.56

Table 1: Private setting result of final accuracy and ϵ.

We believe this result benefits from two attributes of LEASGD: 1) the DP noise helps improve the accuracy by encouraging space exploration and helping workers trapped in local optimum to get out [22]; 2) the great scalability that prevents DP noise from accumulating when the worker scale expands.

7 Conclusion

In this paper, we propose a new decentralized algorithm LEASGD for training deep neural network with differential privacy. LEASGD ensures differential privacy with improved convergence rate and communication efficiency, thanks to the novel leader-follower protocol and privacy-preserving schemes. We theoretically prove its exponential decreasing convergence rate as a function of iterations and good scalability. We also provide a thorough analysis of the performance and privacy trade-off. The real distributed testbed evaluation results show LEASGD outperforms the state-of-the-art decentralized learning algorithm DPSGD in both convergence rate and privacy budget.

8 Acknowledgement

This work is supported in part by the following grants: National Science Foundation CCF-1756013, IIS-1838024 (using resources provided by Amazon Web Services as part of the NSF BIGDATA program), 1717657 and Air Force Research Laboratory FA8750-18-2-0057.

References

1. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.
2. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
3. Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Davis, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.
4. Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, and Paul Tucker. Large scale distributed deep networks. In *International Conference on Neural Information Processing Systems*, pages 1223–1231, 2013.

5. Eric P. Xing, Qirong Ho, Wei Dai, Kyu Kim Jin, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data*, 1(2):1335–1344, 2015.
6. Barnabas Poczos, Ruslan Salakhutdinov, Alexander Smola, and co chair. Scaling distributed machine learning with system and algorithm co-design.
7. Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H Brendan Mcmahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *arXiv preprint arXiv:1805.10559*, 2018.
8. Aurelien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. *AISTATS*, 2018.
9. Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.
10. Feng Yan, Shreyas Sundaram, SVN Vishwanathan, and Yuan Qi. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2483–2493, 2013.
11. Cynthia Dwork. Differential privacy. *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, 2006.
12. Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *Advances in Neural Information Processing Systems*, pages 685–693, 2015.
13. Martin Abadi, Andy Chu, Ian J Goodfellow, H Brendan Mcmahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. *computer and communications security*, pages 308–318, 2016.
14. Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. *Foundations of Computer Science Annual Symposium on*, 26(2):51–60, 2010.
15. S Sundhar Ram, Angelia Nedic, and Venugopal V Veeravalli. Distributed subgradient projection algorithm for convex optimization. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 3653–3656. IEEE, 2009.
16. Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.
17. Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip dual averaging for decentralized optimization of pairwise functions. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1388–1396, New York, New York, USA, 20–22 Jun 2016. PMLR.
18. Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.
19. Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In *Advances in neural information processing systems*, pages 2879–2887, 2014.
20. Hsin-Pai Cheng, Patrick Yu, Haojing Hu, Feng Yan, Shiyu Li, Hai Li, and Yiran Chen. LEASGD: an efficient and privacy-preserving decentralized algorithm for distributed learning. *CoRR*, abs/1811.11124, 2018 (appeared in PPML'18 Workshop - co-located with NeurIPS'18).
21. Convnetjs. `https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html`.
22. Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.