Serverless Federated AUPRC Optimization for Multi-Party Collaborative Imbalanced Data Mining

Xidong Wu
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, Pennsylvania, USA
xidong_wu@outlook.com

Zhengmian Hu
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, Pennsylvania, USA
huzhengmian@gmail.com

Jian Pei
Department of Computer Science
Duke University
Durham, North Carolina, USA
j.pei@duke.edu

Heng Huang*
Department of Computer Science
University of Maryland
College Park, Maryland, USA
henghuanghh@gmail.com

ABSTRACT

To address the big data challenges, serverless multi-party collaborative training has recently attracted attention in the data mining community, since they can cut down the communications cost by avoiding the server node bottleneck. However, traditional serverless multi-party collaborative training algorithms were mainly designed for balanced data mining tasks and are intended to optimize accuracy (*e.g.*, cross-entropy). The data distribution in many real-world applications is skewed and classifiers, which are trained to improve accuracy, perform poorly when applied to imbalanced data tasks since models could be significantly biased toward the primary class. Therefore, the Area Under Precision-Recall Curve (AUPRC) was introduced as an effective metric. Although multiple single-machine methods have been designed to train models for AUPRC maximization, the algorithm for multi-party collaborative training has never been studied. The change from the single-machine to the multi-party setting poses critical challenges. For example, existing single-machine-based AUPRC maximization algorithms maintain an inner state for local each data point, thus these methods are not applicable to large-scale multi-party collaborative training due to the dependence on each local data point.

To address the above challenge, in this paper, we reformulate the serverless multi-party collaborative AUPRC maximization problem as a conditional stochastic optimization problem in a serverless multi-party collaborative learning setting and propose a new ServerLess biAsed sTochastic gradiEnt (SLATE) algorithm to directly optimize the AUPRC. After that, we use the variance reduction technique and propose ServerLess biAsed sTochastic gradiEnt with Momentum-based variance reduction (SLATE-M) algorithm to improve the convergence rate, which matches the best theoretical convergence result reached by the single-machine online method. To the best of our knowledge, this is the first work to solve the multi-party collaborative AUPRC maximization problem. Finally, extensive experiments show the advantages of directly optimizing the AUPRC with distributed learning methods and also verify the efficiency of our new algorithms (*i.e.*, SLATE and SLATE-M).

*This work was partially supported by NSF IIS 1838627, 1837956, 1956002, 2211492, CNS 2213701, CCF 2217003, DBI 2225775.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599499

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; **Distributed algorithms**; *Computer vision tasks.*

KEYWORDS

AUPRC, federated learning, imbalanced data, stochastic optimization, serverless federated learning

ACM Reference Format:
Xidong Wu, Zhengmian Hu, Jian Pei, and Heng Huang. 2023. Serverless Federated AUPRC Optimization for Multi-Party Collaborative Imbalanced Data Mining . In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599499

1 INTRODUCTION

Multi-party collaborative learning, such as distributed learning [2, 9, 19] (typically focus on IID data and train learning model using the gradients from different parties) and federated learning [24] (focus on non-IID data and train model via periodically averaging model parameters from different parties coordinated by the server), have been actively studied at past decades to train large-scale deep learning models in a variety of real-world applications, such as computer vision [11, 37], natural language processing [10], generative modeling [4], *etc.* In literature, multi-party collaborative learning is also often called decentralized learning (compared to centralized learning in the single-machine setting). With different network topology, serverless algorithms could be converted into different multi-party collaborative algorithms (seen in 3.1). On the other hand, although there are many ground-breaking studies with DNN in data classification [11, 27, 32, 36], most works focus on balanced data sets, optimize the cross entropy, and use accuracy to

measure model performance. From the viewpoint of optimization, the cross entropy between the estimated probability distribution based on the output of deep learning models and encoding ground-truth labels is a surrogate loss function of the misclassification rate/accuracy. However, in many real-world applications, such as healthcare and biomedicine [8, 18, 47], where patients make up a far smaller percentage of the population than healthy individuals, the data distribution is frequently skewed due to the scarce occurrence of positive samples. The data from the majority class essentially define the result, and the accuracy fails to be an appropriate metric to assess classifiers' performance. As a result, areas under the curves (AUC), including area under the receiver operating curve (AUROC) and area under precision-recall curves (AUPRC) are given much attention since it excels at discovering models with strong predictive power in imbalanced binary classification [6, 16].

The prediction performance of models, which are trained with cross entropy as the surrogate loss for imbalanced binary classification, may be subpar because cross-entropy is not the surrogate function of AUC, which call for the study of AUC maximization. Recent works have achieved remarkable progress in directly optimizing AUROC with single-machine and multi-party training algorithms [21, 48]. Liu et al. [21] constructed deep AUC as a minimax problem and resolved the stochastic AUC maximization problem with a deep neural network as the classifier. Recently, Yuan et al. [43] and Guo et al. [13] extended the single-machine training to federated learning and proposed a PL-strongly-concave minimax optimization method to maximize AUROC.

However, AUROC is not suitable for data with a much larger number of negative examples than positive examples, and AUPRC can address this issue because it doesn't rely on true negatives. Given that an algorithm that maximizes AUROC does not necessarily maximize AUPRC [8], the design of AUPRC maximization algorithms has attracted attention [17, 26, 30, 31, 35]. Nonetheless, the multi-party algorithm for AUPRC maximization problems has not been studied. Existing AUPRC optimization methods cannot be directly applied to multi-party collaborative training, since they mainly focus on the finite-sum problem and maintain an inner state for each positive data point, which is not permitted in a multi-party online environment. In addition, to improve communication efficiency, serverless multi-party collaborative learning algorithms are needed to avoid the server node bottleneck in deep learning training. Therefore, it is desired to develop efficient stochastic optimization algorithms for serverless multi-party AUPRC maximization for deep learning to meet the challenge of data mining on large-scale imbalanced data sets.

The challenges to design serverless multi-party collaborative AUPRC maximization algorithm are three-fold. The first difficulty lies in the complicated integral definition. To overcome the problem of the continuous integral, we can use some point estimators. Several estimators of AUPRC have been presented in previous works [3, 5]. The average precision (AP) estimator is one of the most popularly used estimators. AP can be directly calculated based on the sample prediction scores and is not subject to sampling bias. It is ideally suited to be used in stochastic optimization problems due to these advantages.

The second difficulty lies in the nested structure and the non-differential ranking functions in the AP. Traditional gradient-based gradient descent techniques cannot directly be used with the original concept of AP. Most existing optimization works use the surrogate function to replace the ranking function in the AP function [13, 17, 21, 26, 31, 35]. We can follow these works and substitute a surrogate loss for the ranking function in the AP function.

The third difficulty is that existing algorithms only focus on finite-sum settings and maintain the inner estimators u_t for each positive data point, which is not permitted in multi-party collaborative online learning. Therefore, despite recent developments, it is still unclear if there is a strategy to optimize AUPRC for multi-party collaborative imbalanced data mining. It is natural to ask the following question:

Can we design serverless multi-party stochastic optimization algorithms to directly maximize AUPRC with guaranteed convergence?

In this paper, we provide an affirmative answer to the aforementioned question. We propose the new algorithms for multi-party collaborative AUPRC maximization and provide systematic analysis. Our main contributions can be summarized as follows:

- We cast the AUPRC maximization problem into non-convex conditional stochastic optimization problem by substituting a surrogate loss for the indicator function in the definition of AP. Unlike existing methods that just focus on finite-sum settings, we consider the stochastic online setting.
- We propose the first multi-party collaborative learning algorithm, ServerLess biAsed sTochastic gradiEnt (SLATE), to solve our new objective. It can be used in an online environment and has no reliance on specific local data points. In addition, with different network topologies, our algorithm can also be used for distributed learning and federated learning.
- Furthermore, we propose a stochastic method (*i.e.*, SLATE-M) based on the momentum-based variance-reduced technique to reduce the convergence complexity in multi-party collaborative learning. Our method can reach iteration complexity of $O\left(1/\epsilon^5\right)$, which matches the lower bound proposed in the single-machine conditional stochastic optimization.
- We conduct extensive experiments on various datasets compared with previous stochastic multi-party optimization algorithms to verify the effectiveness of our methods.

2 RELATED WORK

2.1 AUROC Maximization

There is a long line of research that investigated the imbalanced data mining with AUROC metric [13, 21, 40, 43, 48], which highlight the value of the AUC metric in imbalanced data mining. Earlier works about AUROC focused on linear models with pairwise surrogate losses [18]. Furthermore, Ying et al. [40] solved the AUC square surrogate loss using a stochastic gradient descent ascending approach and provided a minimax reformulation of the loss to address the scaling problem of AUC optimization. Later, Liu et al. [21] studied the application of AUROC in deep learning and reconstructed deep AUC as a minimax problem, which offers a strategy to resolve the stochastic AUC maximization problem with a deep neural network as the predictive model. Furthermore, some methods were

proposed for multi-party AUROC maximization. Yuan et al. [43] and Guo et al. [13] reformulated the federated deep AUROC maximization as non-convex-strongly-concave problem in the federated setting. However, the analyses of methods in [43] and [13] rely on the assumption of PL condition on the deep models. Recently, [42] developed the compositional deep AUROC maximization model and [46] extend it to federated learning.

2.2 AUPRC Maximization

Early works about AUPRC optimization mainly depend on traditional optimization techniques. Recently, Qi et al. [26] analyzed AUPRC maximization with deep models in the finite-sum setting. They use a surrogate loss to replace the ranking function in the AP function and maintain biased estimators of the surrogate ranking functions for each positive data point. They proposed the algorithm to directly optimize AUPRC and show a guaranteed convergence. Afterward, Wang et al. [31] presented adaptive and non-adaptive methods (*i.e.* ADAP and MOAP) with a new strategy to update the biased estimators for each data point. The momentum average is applied to both the outer and inner estimators to track individual ranking scores. More recently, algorithms proposed in [30] reduce convergence complexity with the parallel speed-up and Jiang et al. [17], Wu et al. [35] introduced the momentum-based variance-reduction technology into AUPRC maximization to reduce the convergence complexity. While we developed distributed AUPRC optimization concurrently with [12], they pay attention to X-Risk Optimization in federated learning. Because X-Risk optimization is a sub-problem in conditional stochastic optimization and federated learning could be regarded as decentralized learning with a specific network topology (seen 3.1), our methods could also be applied to their problem.

Overall, existing methods mainly focus on finite-sum single-machine setting [17, 26, 30, 31, 35]. To solve the biased stochastic gradient, they maintain an inner state for local each data point. However, this strategy limits methods to be applied to real-world big data applications because we cannot store an inner state for each data sample in the online environment. In addition, we cannot extend them directly from the single-machine setting to multi-party setting, because under non-IID assumption, the data point on each machine is different and this inner state can only contain the local data information and make it difficult to train a global model.

In the perspective of theoretical analysis, Hu et al. [15] studied the general condition stochastic optimization and proposed two single-machine algorithms with and without using the variance-reduction technique (SpiderBoost) named BSGD and BSpiderboost, and established the lower bound at ε^{-5} in the online setting.

AUPRC is widely utilized in binary classification tasks. It is simple to adapt it for multi-class classifications. If a task has multiple classes, we can assume that each class has a binary classification task and adopt the one vs. the rest classification strategy. We can then calculate average precision based on all classification results.

2.3 Serverless Multi-Party Collaborative Learning

Distributed learning has wide applications in data mining and machine learning problems. Multi-party collaborative learning in this paper has a more general definition that does not rely on the IID assumption of data to guarantee the convergence analysis. In the last years, many serverless multi-party collaborative learning approaches have been put out because they avoid the communication bottlenecks or constrained bandwidth between each worker node and the central server, and also provide some level of data privacy [41]. Lian et al. [20] offered the first theoretical backing for serverless multi-party collaborative training. Then serverless multi-party collaborative training attracts attention [22, 23, 29, 39] and the convergence rate has been improved using many different strategies, including variance extension [29], variance reduction [25, 44], gradient tracking [23], and many more. In addition, serverless multi-party collaborative learning has been applied to various applications, such as reinforcement learning [45], robust training [38], generative adversarial nets (GAN) [22], robust principal component analysis [33] *etc.* However, none of them focus on imbalanced data mining. The serverless multi-party collaborative learning setting in this paper is different to federated learning [24] which uses server with different communication mechanism to periodically average the model parameters and is mainly designed for indirectly aggregating data from numerous devices, such as in IoT applications, not for training large-scale deep learning models with big data.

3 PRELIMINARY

3.1 Serverless Multi-Party Collaborative Learning

Notations: We use $\mathbf{x}$ to denote a collection of all local model parameters $\mathbf{x}_n$, where $n \in [N]$, *i.e.*, $\mathbf{x} = [x_1^\top, x_2^\top, \ldots, x_N^\top]^\top \in \mathbb{R}^{Nd}$. Similarly, we define $\mathbf{u}, \mathbf{v}$ as the concatenation of $\mathbf{u}_n, \mathbf{v}_n$ for $n \in [N]$. In addition, $\otimes$ denotes the Kronecker product, and $\|\cdot\|$ denotes the ℓ_2 norm for vectors, respectively. $\mathcal{D}_n^+$ denotes the positive dataset on the N worker nodes and $\mathcal{D}$ denotes the whole dataset on the n worker nodes. The network system of N worker nodes $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is represented by double stochastic matrix $\underline{\mathbf{W}} = \{\underline{w}_{ij}\} \in \mathbb{R}^{N\times N}$, which is defined as follows: (1) if there exists a link between node i and node j, then $\underline{w}_{ij} > 0$, otherwise $\underline{w}_{ij} = 0$, (2) $\underline{\mathbf{W}} = \underline{\mathbf{W}}^\top$ and (3) $\underline{\mathbf{W}}\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^\top\underline{\mathbf{W}} = \mathbf{1}^\top$. We define the second-largest eigenvalue of $\mathbf{W}$ as λ and $\mathbf{W} := \underline{\mathbf{W}} \otimes \mathbf{I}_d$. We denote the exact averaging matrix as $\mathbf{J} = \frac{1}{N}(\mathbf{1}_n\mathbf{1}_n^\top) \otimes \mathbf{I}_d$ and $\lambda = \|\mathbf{W} - \mathbf{J}\|$. Taking ring network topology as an example. In the ring network, where each node can only exchange information with its two neighbors. The corresponding W is in the form of

$$\underline{\mathbf{W}} = \begin{pmatrix} 1/3 & 1/3 & & & & 1/3 \\ 1/3 & 1/3 & 1/3 & & & \\ & 1/3 & 1/3 & \ddots & & \\ & & \ddots & \ddots & 1/3 & \\ & & & 1/3 & 1/3 & 1/3 \\ 1/3 & & & & 1/3 & 1/3 \end{pmatrix} \in \mathbb{R}^{N\times N}$$

If we change the network topology, serverless multi-Party collaborative learning could become different types of multi-party collaborative training. If $\mathbf{W}$ is $\frac{1}{N}\mathbf{1}\mathbf{1}^\top$, it is converted to distributed learning with the average operation in each iteration. If we choose $\mathbf{W}$ as the Identity matrix and change it to $\frac{1}{N}\mathbf{1}\mathbf{1}^\top$ every q iteration, it would be federated learning.

3.2 AUPRC

AUPRC can be defined as the following integral problem [1]:

$$\text{AUPRC} = \int_{-\infty}^{\infty} \Pr(y = 1 \mid h(\mathbf{x}; \mathbf{z}) \geq c) d \Pr(h(\mathbf{x}; \mathbf{z}) \leq c \mid y = 1)$$

where $h(\mathbf{x}; \mathbf{z})$ is the prediction score function, $\mathbf{x}$ is the model parameter, $\xi = (\mathbf{z}, y)$ is the data point, and $Pr(y = 1|h(\mathbf{x}; \mathbf{z}) \geq c)$ is the precision at the threshold value of c.

To overcome the problem of the continuous integral, we use AP as the estimator to approximate AUPRC, which is given by [3]:

$$\text{AP} = \mathbb{E}_{\xi \sim \mathcal{D}^+} Precision\left(h\left(\mathbf{x}; \mathbf{z}\right)\right) = \mathbb{E}_{\xi \sim \mathcal{D}^+} \frac{\mathrm{r}^+\left(\mathbf{x}\right)}{\mathrm{r}\left(\mathbf{x}\right)}, \tag{1}$$

where $\mathcal{D}^+$ denotes the positive dataset, and samples $\xi = (\mathbf{z}, y)$ are drawn from positive dataset $\mathcal{D}^+$ where $\mathbf{z} \in \mathcal{Z}$ represents the data features and $y = +1$ is the positive label. r^+ denotes the positive data rank ratio of prediction score (*i.e.*, the number of positive data points with no less prediction score than that of ξ including itself over total data number) and r denotes its prediction score rank among all data points (*i.e.*, the number of data points with no less prediction score than that of ξ including itself over total data number). $\mathcal{D}$ denotes the whole datasets and $\xi' = (\mathbf{z}', y') \sim \mathcal{D}$ denote a random data drawn from an unknown distribution $\mathcal{D}$, where $\mathbf{z}' \in \mathcal{Z}$ represents the data features and $y' \in \mathcal{Y} = \{-1, +1\}$. Therefore, (1) is the same as:

$$\text{AP} = \mathbb{E}_{\xi \sim \mathcal{D}^+} \frac{\mathbb{E}_{\xi' \sim \mathcal{D}} \mathbf{I}(h(\mathbf{x}; \mathbf{z}') \geq h(\mathbf{x}; \mathbf{z})) \cdot \mathbf{I}\left(y' = 1\right)}{\mathbb{E}_{\xi' \sim \mathcal{D}} \mathbf{I}(h(\mathbf{x}; \mathbf{z}') \geq h(\mathbf{x}; \mathbf{z}))}$$

We employ the following squared hinge loss:

$$\ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right) = (max\{s - h(\mathbf{x}; \mathbf{z}) + h(\mathbf{x}; \mathbf{z}'), 0\})^2 \tag{2}$$

as the surrogate for the indicator function $\mathbf{I}(h(\mathbf{x}; \mathbf{z}') \geq h(\mathbf{x}; \mathbf{z}))$, where s is a margin parameter, that is a common choice used by previous studies [17, 26, 31]. As a result, the AUPRC maximization problem can be formulated as:

$$AP = \mathbb{E}_{\xi \sim \mathcal{D}^+} \frac{\mathbb{E}_{\xi' \sim \mathcal{D}} \mathbf{I}\left(y' = 1\right) \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right)}{\mathbb{E}_{\xi' \sim \mathcal{D}} \quad \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right)}$$

In the finite-sum setting, it is defined as :

$$AP = \frac{1}{|\mathcal{D}^+|} \sum_{\xi \sim \mathcal{D}^+} \frac{\frac{1}{|\mathcal{D}|} \sum_{\xi \sim \mathcal{D}} \mathbf{I}\left(y' = 1\right) \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right)}{\frac{1}{|\mathcal{D}|} \sum_{\xi \sim \mathcal{D}} \quad \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right)}$$

For convenience, we define the elements in $g(\mathbf{x})$ as the surrogates of the two prediction score ranking function $\mathrm{r}^+\left(\mathbf{x}\right)$ and $\mathrm{r}\left(\mathbf{x}\right)$ respectively. Define the following equation:

$$g\left(\mathbf{x}; \xi, \xi'\right) = \begin{bmatrix} g^1\left(\mathbf{x}; \xi, \xi'\right) \\ g^2\left(\mathbf{x}; \xi, \xi'\right) \end{bmatrix} = \begin{bmatrix} \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right) \mathbf{I}\left(y' = 1\right) \\ \ell\left(\mathbf{x}; \mathbf{z}, \mathbf{z}'\right) \end{bmatrix}$$

and $g(\mathbf{x}; \xi) = \mathbb{E}_{\xi' \sim \mathcal{D}} g\left(\mathbf{x}; \xi, \xi'\right) \in \mathbb{R}^2$, and assume $f(\mathbf{u}) = -\frac{u_1}{u_2} : \mathbb{R}^2 \mapsto \mathbb{R}$ for any $\mathbf{u} = [u_1, u_2]^\top \in \mathbb{R}^2$. Then, we could reformulate the optimization objective into the following stochastic optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} F(\mathbf{x}) &= \mathrm{E}_{\xi \sim \mathcal{D}^+}\left[f(g(\mathbf{x}; \xi)\right] \\ &= \mathrm{E}_{\xi \sim \mathcal{D}^+}\left[f(\mathrm{E}_{\xi' \sim \mathcal{D}} g(\mathbf{x}; \xi, \xi'))\right] \end{aligned} \tag{3}$$

It is similar to the two-level conditional stochastic optimization [15], where the inner layer function depends on the data points sampled from both inner and outer layer functions. Given that $f(\cdot)$ is a nonconvex function, problem (3) is a noncvonex optimiztion problem. In this paper, we considers serverless multi-party collaborative non-convex optimization where N worker nodes cooperate to solve the following problem:

$$\min_{\mathbf{x}} F(\mathbf{x}) = \min_{\mathbf{x}} \frac{1}{N} \sum_{n=1}^{n} F_n(\mathbf{x}) \tag{4}$$

where $F_n(\mathbf{x}) = \mathbb{E}_{\xi_n \sim \mathcal{D}_n^+} f(\mathbb{E}_{\xi_n' \sim \mathcal{D}_n} g_n(\mathbf{x}; \xi_n))$ and $\xi_n' = (\mathbf{z}_n', y_n') \sim \mathcal{D}_n$ and $\xi_n = (\mathbf{z}_n, y_n) \sim \mathcal{D}^+$ We consider heterogeneous data setting in this paper, which refers to a situation where $\mathcal{D}_i$ and $\mathcal{D}_j$ are different ($i \neq j$) on different worker nodes.

In order to design the method, we first consider how to compute the gradient of $F(\mathbf{x})$.

$$\begin{aligned} \nabla F_n(\mathbf{x}) =& \mathbb{E}_{\xi_n \sim \mathcal{D}_n^+} \nabla g_n(\mathbf{x}; \xi_n)^\top \nabla f\left(g_n(\mathbf{x}; \xi_n)\right) \\ =& \mathbb{E}_{\xi_n \sim \mathcal{D}_n^+} \nabla g_n(\mathbf{x}; \xi_n)^\top \left(\frac{-1}{g_n^2(\mathbf{x}; \xi_n)}, \frac{g_n^1(\mathbf{x}; \xi_n)}{\left(g_n^2(\mathbf{x}; \xi_n)\right)^2}\right)^\top \end{aligned}$$

where

$$\begin{aligned} \nabla g_n(\mathbf{x}; \xi_n) &= \begin{bmatrix} \nabla g_n^1(\mathbf{x}; \xi_n) \\ \nabla g_n^2(\mathbf{x}; \xi_n) \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{E}_{\xi_n' \sim \mathcal{D}_n} \mathbf{I}\left(y_n' = 1\right) \nabla \ell\left(\mathbf{x}; \mathbf{z}_n, \mathbf{z}_n'\right) \\ \mathbb{E}_{\xi_n' \sim \mathcal{D}_n} \nabla \ell\left(\mathbf{x}; \mathbf{z}_n, \mathbf{z}_n'\right) \end{bmatrix} \end{aligned}$$

We can notice that it is different from the standard gradient since there are two levels of functions and the inner function also depends on the sample data from the outer layer. Therefore, the stochastic gradient estimator is not an unbiased estimation for the full gradient. Instead of constructing an unbiased stochastic estimator of the gradient [28], we consider a biased estimator of $\nabla F_n(x)$ using one sample ξ from $\mathcal{D}_n^+$ and m sample ξ' from $\mathcal{D}_n$ as $\mathcal{B}_n$ in the following form:

$$\begin{aligned} &\nabla \hat{F}_n\left(\mathbf{x}; \xi_n, \mathcal{B}_n\right) \\ =& (\frac{1}{m} \sum_{\xi' \in \mathcal{B}_n} \nabla g_n(\mathbf{x}; \xi_n, \xi_n'))^\top \nabla f(\frac{1}{m} \sum_{\xi_n' \in \mathcal{B}_n} g_n(\mathbf{x}; \xi_n, \xi_n')) \end{aligned} \tag{5}$$

where $\mathcal{B}_n = \left\{\xi'^j\right\}_{j=1}^m$. It is observed that $\nabla \hat{F}_n\left(\mathbf{x}; \xi_n, \mathcal{B}_n\right)$ is the gradient of an empirical objective such that

$$\hat{F}_n\left(\mathbf{x}; \xi_n, \mathcal{B}_n\right) := f_n\left(\frac{1}{m} \sum_{\xi_n' \in \mathcal{B}_n} g_n(\mathbf{x}; \xi_n, \xi_n')\right).$$

4 ALGORITHMS

In this section, we propose the new serverless multi-party collaborative learning algorithms for solving the problem (4). Specifically, we use the gradient tracking technique (which could be ignored in practice) and propose a ServerLess biAsed sTochastic gradiEnt (SLATE). We further propose an accelerated version of SLATE with momentum-based variance reduction [7] technology (SLATE-M).

4.1 Serverless Biased Stochastic Gradient (SLATE)

Based on the above analysis, we design a serverless multi-party collaborative algorithms with biased stochastic gradient and is

Algorithm 1 SLATE Algorithm

1: **Input:** T, step size η inner batch size m and mini-batch size b; $\mathbf{u}_{n,0} = 0$ and $\mathbf{v}_{n,0} = 0$ for $n \in \{1, \cdots, N\}$
2: **Initialize:** $x_{n,0} = \frac{1}{N}\sum_{k=1}^{N} x_{n,0}$.
3: **for** $t = 0, 1, \ldots, T$ **do**
4: **for** $n = 1, 2, \ldots, N$ **do**
5: Draw b samples $\mathcal{B}_{n,t}^{+} = \{\xi_{n,t}^{i}\}_{i=1}^{b}$ from $\mathcal{D}_n^{+}$
6: Draw m samples $\mathcal{B}_{n,t} = \left\{\xi_{n,t}^{\prime j}\right\}_{j=1}^{m}$ from $\mathcal{D}_n$,
7: $\mathbf{u}_{n,t} = \frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t}; \xi_{n,t}^{i}, \mathcal{B}_{n,t})$ as in (6)
8: $\mathbf{v}_{n,t} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{v}_{t-1}^{r} + \mathbf{u}_{t}^{r} - \mathbf{u}_{t-1}^{r})$
9: $\mathbf{x}_{n,t+1} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{x}_{t}^{n} - \eta \mathbf{v}_{n,t})$
10: **end for**
11: **end for**
12: **Output:** x chosen uniformly random from $\{\bar{\mathbf{x}}_t\}_{t=1}^{T}$.

named SLATE. Algorithm 1 shows the algorithmic framework of the SLATE. Step 8 could be ignored in practice.

At the beginning of Algorithm 1, one simply initializes local model parameters $\mathbf{x}$ for all worker nodes. Given the couple structure of problem (4). We can assign the value of gradient estimator $\mathbf{u}_{n,0}$ and gradient tracker $\mathbf{v}_{n,0}$ as 0.

At the Lines 5-6 of Algorithm 1, we draw b samples as $\mathcal{B}_{n,t}^{+}$ from positive dataset $\mathcal{D}_n^{+}$ and m samples as $\mathcal{B}_{n,t}$from full data sets $\mathcal{D}_n$ on each node, respectively. We use a biased stochastic gradient to update the gradient estimator $\mathbf{u}_{n,t}$ according to the (6).

$$\mathbf{u}_{n,t} = \tag{6}$$
$$\sum_{\xi \in \mathcal{B}_{n,t}^{+}} \sum_{\xi' \in \mathcal{B}_{n,t}} \frac{\left(g^1\left(\mathbf{x}_{n,t}; \xi, \xi'\right) - g^2\left(\mathbf{x}_{n,t}; \xi, \xi'\right) \mathbf{I}\left(\mathrm{y}' = 1\right)\right) \nabla \ell\left(\mathbf{x}_{n,t}; \mathbf{z}, \mathbf{z}'\right)}{bm\left(g^2\left(\mathbf{x}_{n,t}; \xi, \xi'\right)\right)^2}$$

where $\xi = (\mathbf{z}, y)$ and $\xi' = (\mathbf{z}', y')$

Afterward, at the Line 8 of Algorithm 1, we adopt the gradient tracking technique [23] to reduce network consensus error, where we update the $\mathbf{v}_{n,t}$ and then do the consensus step with double stochastic matrix $\mathbf{W}$ as:

$$\mathbf{v}_{n,t} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{v}_{t-1}^{r} + \mathbf{u}_{t}^{r} - \mathbf{u}_{t-1}^{r})$$

Finally, at the Line 9 of Algorithm 1, we update the model with gradient tracker $\mathbf{v}_{n,t}$, following the consensus step among worker nodes with double stochastic matrix $\mathbf{W}$:

$$\mathbf{x}_{n,t+1} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{x}_{r,t} - \eta \mathbf{v}_{r,t})$$

The output $\bar{\mathbf{x}}_t$ is defined as: $\bar{\mathbf{x}}_t = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_{n,t}$.

4.2 SLATE-M

Furthermore, we further propose an accelerated version of SLATE (SLATE-M) based on the momentum-based variance reduced technique, which has the better convergence complexity. The details of the algorithm are shown in Algorithm 2. Step 11 could be ignored in practice.

At the beginning, similar to the SLATE, one initialize local model parameters $\mathbf{x}$ for all worker nodes, as seen in Lines 1-2 in Algorithm 2.

Algorithm 2 SLATE-M Algorithm

1: **Input:** T, step size η, momentum coefficient α, inner batch size m and mini-batch size b, and initial batch size B;
2: **Initialize:** $\mathbf{x}_{n,0} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{x}_{n,0}$
3: Draw B samples of $\{\xi_{n,0}^{i}\}_{i=1}^{B}$ from $\mathcal{D}_n^{+}$, and draw m samples $\mathcal{B}_{n,0} = \left\{\xi_{n,0}^{\prime j}\right\}_{j=1}^{m}$ from $\mathcal{D}_n$, $\mathbf{u}_{n,0} = \frac{1}{B}\sum_{i=1}^{B} \nabla \hat{F}_n(\mathbf{x}_{n,0}; \xi_{n,0}^{i}, \mathcal{B}_{n,0}) \forall n \in [N]$
4: $\mathbf{v}_{n,0} = \sum_{r=1}^{N} \underline{w}_{nr}\mathbf{u}_{r,0} \forall n \in [N]$
5: $\mathbf{x}_{n,1} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{x}_{n,0} - \eta \mathbf{v}_{n,0}) \forall n \in [N]$
6: **for** $t = 1, 2, \ldots, T$ **do**
7: **for** $n = 1, 2, \ldots, N$ **do**
8: Draw b samples of $\{\xi_{n,1}^{0}, \cdots, \xi_{n,1}^{b}\}$ from $\mathcal{D}_n^{+}$
9: Draw m samples $\mathcal{B}_n = \left\{\xi^{\prime j}\right\}_{j=1}^{m}$ from $\mathcal{D}_n$,
10: $\mathbf{u}_{n,t} = \frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t}; \xi_{n,t}^{i}, \mathcal{B}_{n,t}) + (1-\alpha)(\mathbf{u}_{n,t-1} - \frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t-1}; \xi_{n,t}^{i}, \mathcal{B}_{n,t}))$
11: $\mathbf{v}_{n,t} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{v}_{t-1}^{r} + \mathbf{u}_{t}^{r} - \mathbf{u}_{t-1}^{r})$
12: $\mathbf{x}_{n,t+1} = \sum_{r=1}^{N} \underline{w}_{nr}(\mathbf{x}_{t}^{n} - \eta \mathbf{v}_{n,t})$
13: **end for**
14: **end for**
15: **Output:** x chosen uniformly random from $\{\bar{\mathbf{x}}_t\}_{t=1}^{T}$.

Different from SLATE, we initialize the $\mathbf{u}_{n,0}$ with initial batch size B and $\mathbf{v}_{n,0} \forall n \in [N]$, which can be seen in Lines 3-4 in Algorithm 2. Then we do the consensus step to update the model parameters $\mathbf{x}_n$. The definition of $\hat{F}_n(\mathbf{x}_{n,0}; \xi_{n,0}^{i}, \mathcal{B}_{n,0})$ is similar to (6) as below:

$$\frac{1}{|\mathcal{B}_{n,t}|}\hat{F}_n(\mathbf{x}_{n,t}; \xi_{n,t}^{i}, \mathcal{B}_{n,t}) = \tag{7}$$
$$\sum_{\xi \in \mathcal{B}_{n,t}^{+}} \sum_{\xi' \in \mathcal{B}_{n,t}} \frac{\left(g^1\left(\mathbf{x}_{n,t}; \xi, \xi'\right) - g^2\left(\mathbf{x}_{n,t}; \xi, \xi'\right) \mathbf{I}\left(\mathrm{y}' = 1\right)\right) \nabla \ell\left(\mathbf{x}_{n,t}; \mathbf{z}, \mathbf{z}'\right)}{|\mathcal{B}_{n,t}| m\left(g^2\left(\mathbf{x}_{n,t}; \xi, \xi'\right)\right)^2}$$

where $|\mathcal{B}_{n,t}|$ denotes the size of batch $\mathcal{B}_{n,t}$ and $\xi = (\mathbf{z}, y)$ and $\xi' = (\mathbf{z}', y')$.

Afterwards, similar to SLATE, each iteration, we draw b samples from positive dataset $\mathcal{D}_n^{+}$ and m samples from full data sets $\mathcal{D}_n$ on each worker node, respectively to construct the biased stochastic gradient, seen in Line 8-9 of Algorithm 2.

The key different between SLATE and SLATE-M is that we update gradient estimator $\mathbf{u}_{n,t}$ in SLATE-M with the following variance reduction method:

$$\mathbf{u}_{n,t} = \frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t}; \xi_{n,t}^{i}, \mathcal{B}_{n,t}) + (1-\alpha)(\mathbf{u}_{n,t-1} - \frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t-1}; \xi_{n,t}^{i}, \mathcal{B}_{n,t}))$$

where $\frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t}; \xi_{n,t}^{i}, \mathcal{B}_{n,t})$ and $\frac{1}{b}\sum_{i=1}^{b} \nabla \hat{F}_n(\mathbf{x}_{n,t-1}; \xi_{n,t}^{i}, \mathcal{B}_{n,t})$ are defined in (7)

Finally, we update gradient tracker $\mathbf{v}_{n,t}$ and model parameters $\mathbf{x}_{n,t}$ as in Lines 11-12 in Algorithm 2.

5 THEORETICAL ANALYSIS

We will discuss some mild assumptions and present the convergence results of our algorithms (SLATE and SLATE-M).

5.1 Assumptions

In this section, we introduce some basic assumptions used for theoretical analysis.

Assumption 1. *$\forall n \in [N]$, we assume (i) there is $C(>0)$ that $\ell(x; z_n, z_n) > C$; (ii) there is $M(>0)$ that $0 < \ell(x; z_n, z_n') < M$; (iii) $\ell(x; z_n, z_n')$ is Lipschitz continuous and smooth with respect to model $\mathbf{x}$ for any $\xi_n = (z_n, y_n) \sim \mathcal{D}^+, \xi_n' = (z_n', y_n') \sim \mathcal{D}$.*

Assumption 2. *$\forall n \in [N]$, we assume there exists a positive constant σ, such that $\|\nabla g(\mathbf{x}; \xi, \xi')\|^2 \leq \sigma^2 \forall \xi \sim \mathcal{D}_n^+, \xi' \sim \mathcal{D}_n$.*

Assumptions 1 and 2 are a widely used assumption in optimization analysis of AUPRC maximization [17, 26, 31]. They can be easily satisfied when we choose a smooth surrogate loss function $\ell(\mathbf{x}; \mathbf{z}, \mathbf{z}')$ and a bounded score function model $h(\mathbf{x}; \cdot)$.

Furthermore, based on Assumptions 1 and 2, we can build the smoothness and lipschitz continuity of objective function in the problem (4).

Lemma 1. *(Lemma 1 in [31]) Suppose Assumptions 1 and 2 hold, then $\forall \mathbf{x}, \|g_n(\mathbf{x}; \xi)\|^2 \leq \sigma_g^2$, $g_n(\mathbf{x}; \xi)$ is L_g-Lipschitz and S_g-smooth for $\mathbf{xi} \sim \mathcal{D}_n^+$, and $\forall u \in \Omega$, $f(u)$ is L_f-Lipschitz and S_f-smooth. $\forall \mathbf{x}, F_n(\mathbf{x})$ is L_F-Lipschiz and S_F-smooth.*

From the Lemma 1, we have f_n and g_n are S_f-smooth and S_h-smooth. This implies that for an samgle $\xi_n \sim \mathcal{D}_n^+$ there exist $S_f > 0$ and $S_g > 0$ such that

$$\mathbb{E}\|\nabla f_n(x_1) - \nabla f_n(x_2)\| \leq S_f \|x_1 - x_2\|$$
$$\mathbb{E}\|\nabla g_n(y_1, \xi_n) - \nabla g_n(y_2, \xi_n)\| \leq S_g \|y_1 - y_2\|$$

And f_n and g_n are L_f-Lipchitz continuous and L_g-Lipchitz continuous. This implies that there exist $L_f > 0$ and $L_g > 0$ such that

$$\mathbb{E}\|\nabla f_n(x)\|^2 \leq L_f^2$$
$$\mathbb{E}\|\nabla g_n(y_1, \xi_n)\|^2 \leq S_g^2$$

In addition, we also have bounded variance of g_n. There exist $\sigma_g > 0$ such that

$$\mathbb{E}_{\xi_n \sim \mathcal{D}_n^+} \|g_n(\mathbf{x}; \xi_n, \xi_n') - \mathbb{E}_{\xi_n' \sim \mathcal{D}_n} g_n(\mathbf{x}; \xi_n, \xi_n'\|^2 \leq \sigma_g^2$$

which indicates that the inner function g_n has bounded variance. To control the estimation bias, we follow the analysis in single-machine conditional stochastic optimization [15].

Lemma 2. *(Proposition B.1 in [15]) Under Assumptions 1 and 2, on the n-th worker node, for a sample $\xi_n \sim \mathcal{D}_n^+$ and m samples $\mathcal{B}_n$ from $\mathcal{D}_n$,*
(a) $\mathcal{B}_n = \{\xi'^j\}_{j=1}^m$ and we have

$$\|\mathbb{E}\nabla \hat{F}_n(x; \xi_n, \mathcal{B}_n) - \nabla F_n(x)\|^2 \leq \frac{L_g^2 S_f^2 \sigma_g^2}{m} \tag{8}$$

(b) $\mathbb{E}\nabla \hat{F}_n(x; \xi_n, \mathcal{B}_n)$ are S_F-Lipschitz smooth
(c)

$$\left\|\nabla \left(f(\hat{g}_n(\mathbf{x}, \xi_n))\right) - \nabla \hat{F}(x)\right\|_2^2 \leq L_f^2 L_g^2 \tag{9}$$

Lemma 2 (a) provide a bound of biased stochastic gradient, which will be used in the following theoretical analysis.

Assumption 3. *The function $F_n(x)$ is bounded below,* i.e., $\inf_x F_n(x) > -\infty$.

5.2 The Communication Mechanism in Serverless Multi-Party Collaborative Training

The network system of N worker nodes $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is represented by double stochastic matrix $\underline{\mathbf{W}} = \{\underline{w}_{ij}\} \in \mathbb{R}^{N \times N}$ in the analysis.

For the ease of exposition, we write the $\mathbf{x}_t$ and $\mathbf{v}_t$-update in Algorithm 1 and Algorithm 2 in the following equivalent matrix form: $\forall t \geq 0$,

$$\mathbf{v}_t = \mathbf{W}(\mathbf{v}_{t-1} + \mathbf{u}_t - \mathbf{u}_{t-1}), \quad \mathbf{x}_{t+1} = \mathbf{W}(\mathbf{x}_{t-1} - \eta \mathbf{v}_t)$$

where $\mathbf{W} := \underline{\mathbf{W}} \otimes \mathbf{I}_d$ and $\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t$ are random vectors in $\mathbb{R}^{Nd}$ that respectively concatenate the local estimates $\{\mathbf{x}_{n,t}\}_{n=1}^N$ of a stationary point of F , gradient trackers $\{\mathbf{v}_{n,t}\}_{n=1}^N$, gradient estimators $\{\mathbf{u}_{n,t}\}_{n=1}^N$. With the exact averaging matrix $\mathbf{J}$, we have following quantities:

$$\mathbf{1} \otimes \bar{\mathbf{x}}_t := \mathbf{J}\mathbf{x}_t, \ \mathbf{1} \otimes \bar{\mathbf{u}}_t := \mathbf{J}\mathbf{u}_t, \ \mathbf{1} \otimes \bar{\mathbf{v}}_t := \mathbf{J}\mathbf{v}_t$$

Next, we enlist some useful results of gradient tracking-based algorithms for serveress multi-party collaborative stochastic optimization

Lemma 3. *(Lemma 1 in [39]) For double stochastic matrix, we have the following:*
(a) $\|\mathbf{W}\mathbf{x} - \mathbf{J}\mathbf{x}\| \leq \lambda \|\mathbf{x} - \mathbf{J}\mathbf{x}\|, \forall \mathbf{x} \in \mathbb{R}^{nd}$.
(b) $\bar{\mathbf{v}}_t = \bar{\mathbf{u}}_t, \forall t \geq 0$. As the update step in 1 and 2, we have

$$\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t - \eta \bar{\mathbf{v}}_t = \bar{\mathbf{x}}_t - \eta \bar{\mathbf{u}}_t$$

(c) According to the definition of network $\mathbf{W}$, we have the following inequalities: $\forall k \geq 0$,

$$\|\mathbf{x}_{t+1} - \mathbf{J}\mathbf{x}_{t+1}\|^2 \leq \frac{1+\lambda^2}{2} \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{2\eta^2\lambda^2}{1-\lambda^2} \|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 \tag{10}$$

$$\|\mathbf{x}_{t+1} - \mathbf{J}\mathbf{x}_{t+1}\|^2 \leq 2\lambda^2 \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + 2\eta^2\lambda^2 \|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 \tag{11}$$

$$\|\mathbf{x}_{t+1} - \mathbf{J}\mathbf{x}_{t+1}\| \leq \lambda \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \eta\lambda \|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\| \tag{12}$$

Then, we study the convergence properties of SLATE and SLATEM. We first discus the metric to measure convergence of our algorithms. Given that the loss function is nonconvex, we are unable to demonstrate convergence to an global minimum point. Instead, we establish convergence to an approximate stationary point, defined below:

Definition 1. A point x is called ϵ-stationary point if $\|\nabla f(x)\| \leq \epsilon$. Generally, a stochastic algorithm is defined to achieve an ϵ-stationary point in T iterations if $\mathbb{E}\|\nabla f(x_T)\| \leq \epsilon$.

5.3 Convergence Analysis of SLATE Algorithm

First, we study the convergence properties of our SLATE algorithm. The detailed proofs are provided in the supplementary materials.

Theorem 1. *Suppose the sequence $\{\bar{\mathbf{x}}_t\}_{t=1}^T$ be generated from Algorithm 1 and Assumptions 1, 2, and 3 hold, $0 < \eta \leq \min\{\frac{1-\lambda^2}{24\lambda^2 S_F}, \frac{1}{6S_F}\}$,*

SLATE in algorithm 1 has the following

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 \leq \frac{2\mathbb{E}[F(\bar{\mathbf{x}}_0)-F(\bar{\mathbf{x}}_T)]}{\eta T} + \left(\frac{1}{\lambda^2}+5N\right)\frac{32\lambda^2\eta^2L_f^2L_g^2S_F^2}{(1-\lambda^2)^2N} + \frac{2L_g^2S_f^2\sigma_g^2}{m} + \frac{2\eta S_F L_f^2 L_g^2}{N}$$

Corollary 1. *Based on the analysis in Theorem 1, by setting* $\eta = O(\sqrt{\frac{N}{T}})$ *, SLATE in Algorithm 1 has the following*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 \leq O(\frac{\mathbb{E}[F(\bar{\mathbf{x}}_0)-F(\bar{\mathbf{x}}_T)]}{(NT)^{1/2}}) + \left(\frac{1}{\lambda^2}+5\right)\frac{24\lambda^2L_f^2L_g^2S_F^2}{(1-\lambda^2)^2}O(\frac{N}{T}) + \frac{2L_g^2S_f^2\sigma_g^2}{m} + O(\frac{2S_FL_f^2L_g^2}{(NT)^{1/2}})$$

Based on the result in Theorem 1 and Corollary 1, we can get the convergence result of SLATE.

Remark 1. According to Corollary 1, without loss of generality, we let $m = O(\varepsilon^{-2})$, $b = O(1)$ and $\sqrt{T} > N$, we know to make $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 \leq \varepsilon^2$, we have iterations T should be as large as $O(N^{-1}\varepsilon^{-4})$.

In Algorithm 1, we sample $b+m$ data points to build the biased stochastic gradients $\mathbf{u}_{n,t}$, and need T iterations. Thus, our SLATE algorithm has a sample complexity of $m \cdot T = O(N^{-1}\varepsilon^{-6})$, for finding an ϵ-stationary point. In addition, the result also indicates the linear speedup of our algorithm with respect to the number of worker nodes.

5.4 Convergence Analysis of SLATE-M

In the subsection, we study the convergence properties of our SLATE-M algorithm. The details about proofs are provided in the supplementary materials.

Theorem 2. *Suppose the sequence* $\{\bar{\mathbf{x}}_t\}_{t=1}^T$ *be generated from Algorithm 2 and Assumptions 1, 2, and 3 hold,* $0 < \eta \leq min\{\frac{1}{4}, \frac{(1-\lambda^2)^2}{90\lambda^2}, \frac{\sqrt{1-\lambda^2}}{12\sqrt{7}\lambda}\}\frac{1}{S_F}$ *and* $\alpha = \frac{72S_F^2\eta^2}{Nb}$ *, SLATE-M in Algorithm 2 has the following*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla\mathbf{F}(\bar{\mathbf{x}}_t)\|^2 \leq \frac{2(\mathbf{F}(\bar{\mathbf{x}}_0)-\mathbf{F}(\bar{\mathbf{x}}_T))}{\eta T} + \frac{3L_g^2S_f^2\sigma_g^2}{m} + 3\frac{L_g^2L_f^2}{\alpha NBT} + \frac{6\alpha L_g^2L_f^2}{Nb} + \frac{96\lambda^2L_g^2L_f^2}{(1-\lambda^2)^3BT} + \frac{256\lambda^2\alpha^2L_f^2L_g^2}{(1-\lambda^2)^3} + \frac{64\lambda^4\mathbb{E}\left\|\nabla\hat{\mathbf{F}}_0\right\|^2}{(1-\lambda^2)^3NT}$$

Corollary 2. *Based on the analysis in the theorem 2, we choose* $b = O(1), \eta = O(\frac{N^{2/3}}{T^{1/3}}), \alpha = O(\frac{N^{1/3}}{T^{2/3}}), B = O(\frac{T^{1/3}}{N^{2/3}})$*, SLATE-M in Algorithm 2 has the following*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla\mathbf{F}(\bar{\mathbf{x}}_t)\|^2 \leq O(\frac{2(\mathbf{F}(\bar{\mathbf{x}}_0)-\mathbf{F}(\bar{\mathbf{x}}_T))}{(NT)^{2/3}} + \frac{3L_g^2S_f^2\sigma_g^2}{m} + O(\frac{3L_g^2L_f^2}{(NT)^{2/3}}) + O(\frac{6L_g^2L_f^2}{(NT)^{2/3}}) + \frac{352\lambda^2L_f^2L_g^2}{(1-\lambda^2)^3}O(\frac{N^{2/3}}{T^{4/3}}) + \frac{64\lambda^4\mathbb{E}\left\|\nabla\hat{\mathbf{F}}_0\right\|^2}{(1-\lambda^2)^3NT} \quad (13)$$

Based on the result in theorem 2, we can get the convergence result of SLATE-M.

Remark 2. According to Corollary 2, without loss of generality, Let $m = O(\varepsilon^{-2}), b = O(1)\ \eta = O(\frac{N^{2/3}}{T^{1/3}}), \alpha = O(\frac{N^{1/3}}{T^{2/3}})$, and $B = O(\frac{T^{1/3}}{N^{2/3}})$, we know to make $\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 \leq \varepsilon^2$, we have iterations T should be as large as $O(N^{-1}\varepsilon^{-3})$.

In Algorithm 2, in each iteration, we sample $b+m$ data points to build the biased stochastic gradients $\mathbf{u}_{n,t}$, and need T iterations. Thus, our SLATE-M algorithm has a sample complexity of $m \cdot T = O(N^{-1}\varepsilon^{-5})$, for finding an ϵ-stationary point, which also achieves the linear speedup of our algorithm with respect to the number of worker nodes.

Remark 3. The sample complexity of $O(N^{-1}\varepsilon^{-5})$ in SLATE-M matches the best convergence complexity achieved by the single-machine stochastic method for conditional stochastic optimization in the online setting, and also match the lower bound for the online stochastic algorithms [15].

6 EXPERIMENTS

In this section, we conduct extensive experiments on imbalanced benchmark datasets to show the efficiency of our algorithms. All experiments are run over a machine with AMD EPYC 7513 32-Core Processors and NVIDIA RTX A6000 GPU. The source code is available at https://github.com/xidongwu/D-AUPRC.

The goal of our experiments is two-fold: (1) to verify that (4) is the surrogate function of AUPRC and illustrate that directly optimizing the AUPRC in the multi-party collaborative training would improve the model performance compared with traditional loss optimization, and (2) to show the efficiency of our methods for AUPRC maximization.

6.1 Configurations

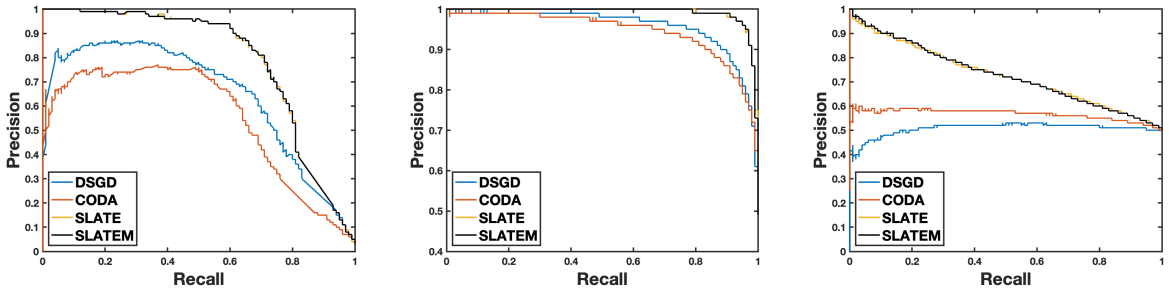
Datasets: We conduct experiments on imbalanced benchmark datasets from LIBSVM data [1]: w7a and w8a, and four typical image datasets: MNIST dataset, Fashion-MNIST dataset, CIFAR-10, and Tiny-ImageNet dataset (seen in Table 1). For w7a and w8a, we scale features to [0, 1]. For image datasets, following [26, 31], we construct the imbalanced binary-class versions as follows: Firstly, the first half of the classes (0 - 4) in the original MNIST, Fashion-MNIST, and CIFAR-10, and (0 - 99) in Tiny-ImageNet datasets are converted to be the negative class, and another half of classes are considered to be the positive class. Because the original distributions of image datasets are balanced, we randomly drop 80% of the positive examples in the training set to make them imbalanced and keep test sets of image datasets unchanged. Finally, we evenly partition each datasets into disjoint sets and distribute datasets among worker nodes.

Models: For w7a and w8a, we use two layers of neural networks with the dimension of the hidden layer as 28. The RELU is used as the activation function. For MNIST, Fashion MNIST and Cifar-10 data sets, we choose model architectures from [34] for our imbalanced binary image classification task, as shown in Table 5 and Table 6. For Tiny-ImageNet, we choose ResNet-18 [14] as the classifier. In our algorithms, We modify the output of all models to

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

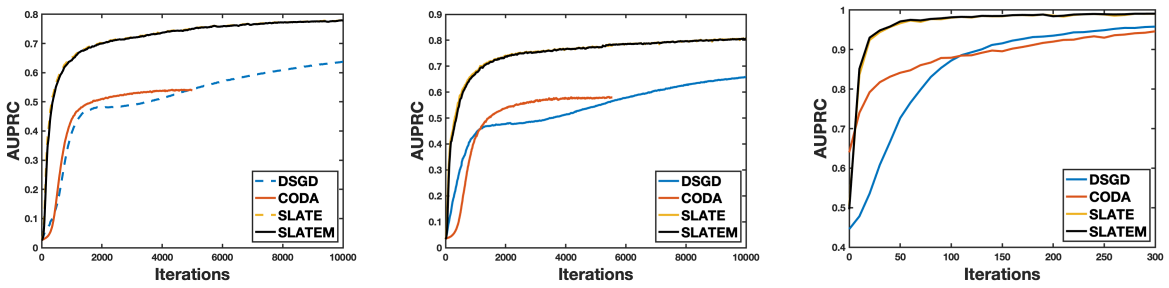
Table 1: Statistics of benchmark datasets

Data Set	Training examples	Testing examples	Feature Size	Proportion of positive data
w7a	24692	25057	300	2.99%
w8a	49749	14951	300	2.97 %
MNIST	60000	10000	28×28	16.7%
Fashion MNIST	60000	10000	28×28	16.7 %
CIFAR-10	50000	10000	$3 \times 32 \times 32$	16.7 %
Tiny-ImageNet	100000	10000	$3 \times 64 \times 64$	16.7 %

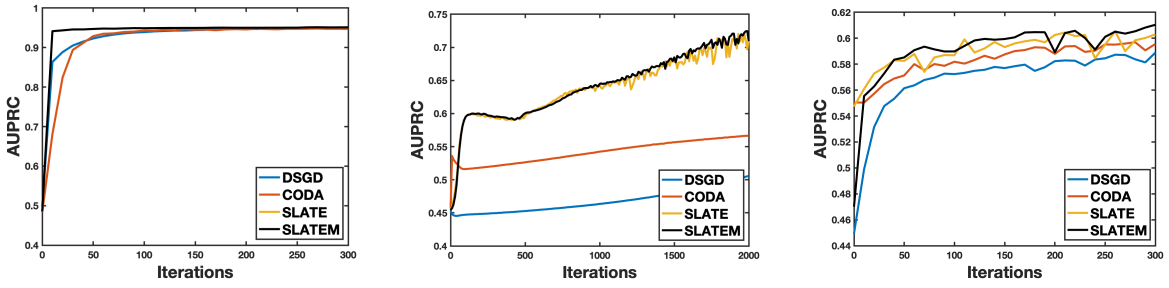


(a) w7a dataset (b) MNIST dataset (c) CIFAR-10 dataset

Figure 1: Precision-Recall curves of the models on the testing set



(a) w7a dataset (b) w8a dataset (c) MNIST dataset



(d) Fashion MNIST dataset (e) CIFAR-10 dataset (f) Tiny-ImageNet dataset

Figure 2: AP vs Iterations on the test set

1 and the sigmoid function is followed since we consider binary classification tasks.

In the experiments, the number of worker nodes is set as N = 20 and we use the ring-based topology as the communication network structure. [20].

6.2 Comparison with Existing Multi-Party Stochastic Methods

Baselines: We compare our algorithms with two baselines: 1) D-PSGD [20], a SGD-like serverless multi-party collaborative algorithm with the Cross-Entropy loss as the optimization objective.

Table 2: Final averaged AP scores on the testing data

Method	w7a	w8a	MNIST	Fashion MNIST	CIFAR-10	Tiny-ImageNet
D-PSGD	0.6372	0.6585	0.9592	0.9497	0.5058	0.5906
CODA	0.5414	0.5786	0.9460	0.9474	0.5668	0.5971
SLATE	0.7788	0.8072	0.9911	0.9515	0.7279	0.6032
SLATEM	0.7778	0.8063	0.9913	0.9515	0.7285	0.6131

Table 3: SLATE test accuracy on CIFAR-10 with margin parameter s, and positive batch size b (B=60)

Margin	0.1	0.3	0.5	0.7	0.9
b = 5	0.6270	0.5971	0.5992	0.5928	0.5765
b = 10	0.6910	0.5887	0.5956	0.5986	0.5989
b = 15	0.7248	0.5895	0.5902	0.5952	0.5979
b = 20	0.7279	0.6001	0.5870	0.5929	0.5962
b = 25	0.7216	0.6038	0.5876	0.5933	0.5962

Table 4: SLATE-M test accuracy on CIFAR-10 with margin parameter s, positive batch size b (B = 60), and α

Margin	0.1 (α = 0.1)	0.1 (α = 0.9)	0.3 (α = 0.1)	0.3 (α = 0.9)
b = 15	0.7273	0.7272	0.5853	0.5895
b = 20	0.7285	0.7289	0.5986	0.6001
b = 25	0.7167	0.7206	0.6024	0.6036

D-PSGD runs SGD locally and then computes the neighborhood weighted average by fetching model parameters from neighbors; 2) CODA, a typical federated learning algorithm for optimizing minimax formulated AUROC loss [13, 43]. CODA runs local SGDA with the periodic model average in the federated learning setting. We convert it into the serverless multi-party setting and run local SGDA, following the consensus step to update the models. Gradient tracking steps are ignored. In the experiments, we ignore the gradient tracking steps to reduce computation and communication costs.

Parameter tuning: We perform a grid search to tune all methods carefully. The total batch size m drawn from $\mathcal{D}$ is chosen in the set $\{20, 20, 20, 20, 60, 200\}$. For SLATE and SLATE-M, the positive batch size b in the total batch size is chosen in the set $\{2, 2, 3, 5, 20, 35\}$, and m - b negative data points. The squared hinge loss is used as (2), α is chosen from $\{0.1, 0.9\}$ and the margin parameter s is selected from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The step size is selected from the set $\{0.01, 0.005, 0.001\}$. For the D-PSGD, the step size is chosen in the set $\{0.01, 0.005, 0.001\}$. For the CODA, the step size for minimum variable is chose from the set $\{0.01, 0.005, 0.001\}$ and that for the maximum variable is chosen from the set $\{0.0001, 0.0005, 0.001\}$. Moreover, we use Xavier normal to initialize models.

Experimental results: Table 2 summarizes the final results on the test sets. In order to present the advantage of optimization of AUPRC, we plot the Precision-Recall curves of final models on testing sets of W7A, MNIST, and CIFAR-10 when training stop in Figure 1. Then we illustrate the convergence curve on test sets in Figure 2. Results show that our algorithms (*i.e.*, SLATE, SLATE-M) can outperform baselines in terms of AP with a great margin across each benchmark, regardless of model structure. The experiments verify that 1) the objective function (4) is a good surrogate loss function of AUPRC and directly optimizing the AUPRC in the multi-party collaborative training would improve the model performance compared with traditional loss optimization in the imbalanced data mining. 2) Although CODA, with minimax formulated AUROC loss, has a relatively better performance compared with D-PSGD in large-scale datasets (CIFAR-10 and Tiny-ImageNet), the results verify the previous results that an algorithm that maximizes AUROC does not necessarily maximize AUPRC. Therefore, designing the algorithms for AUPRC in multi-party collaborative training is necessary. 3) our algorithms can efficiently optimize the (4) and largely improve the performance in terms of AUPRC in multi-party collaborative imbalanced data mining. 4) In datasets CIFAR-10 and Tiny-ImageNet, SLATE-M has better performance compared with SLATE.

Ablation study: In this part, we study the effect of margin parameters, positive batch size, and α of SLATE-M. The results are listed in Table 3 and Table 4.

7 CONCLUSION

In this paper, we systematically studied how to design serverless multi-party collaborative learning algorithms to directly maximize AUPRC and also provided the theoretical guarantee on algorithm convergence. To the best of our knowledge, this is the first work to optimize AUPRC in the multi-party collaborative training. We cast the AUPRC maximization problem into non-convex two-level stochastic optimization functions under the multi-party collaborative learning settings as the problem (4), and proposed the first multi-party collaborative learning algorithm, ServerLess biAsed sTochastic gradiEnt (SLATE). Theoretical analysis shows that SLATE has a sample complexity of $O(\varepsilon^{-6})$ and shows a linear speedup respective to the number of worker nodes. Furthermore, we proposed a stochastic method (*i.e.*, SLATE-M) based on the momentum-based variance-reduced technique to reduce the convergence complexity for maximizing AP in multi-party collaborative optimization. Our methods reach iteration complexity of $O\left(1/\epsilon^5\right)$, which matches the best convergence complexity achieved by the single-machine stochastic method for conditional stochastic optimization in the online setting, and also matches the lower bound for the online stochastic algorithms. Unlike existing single-machine methods that just focus on finite-sum settings and must keep an inner state for each positive data point, we consider the stochastic online setting. The extensive experiments on various data sets compared with previous stochastic multi-party collaborative optimization algorithms validate the effectiveness of our methods. Experimental results also demonstrate that directly optimizing the AUPRC in the multi-party collaborative training would largely improve the model performance compared with traditional loss optimization.

REFERENCES

[1] Donald Bamber. 1975. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of mathematical psychology* 12, 4 (1975), 387–415.

[2] Runxue Bao, Xidong Wu, Wenhan Xian, and Heng Huang. 2022. Doubly sparse asynchronous learning for stochastic composite optimization. In *Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*. 1916–1922.

[3] Kendrick Boyd, Kevin H Eng, and C David Page. 2013. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 451–466.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

[5] Andrew Brown, Weidi Xie, Vicky Kalogeiton, and Andrew Zisserman. 2020. Smooth-ap: Smoothing the path towards large-scale image retrieval. In *European Conference on Computer Vision*. Springer, 677–694.

[6] Corinna Cortes and Mehryar Mohri. 2003. AUC optimization vs. error rate minimization. *Advances in neural information processing systems* 16 (2003).

[7] Ashok Cutkosky and Francesco Orabona. 2019. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems* 32 (2019).

[8] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*. 233–240.

[9] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. 2012. Large scale distributed deep networks. *Advances in neural information processing systems* 25 (2012).

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017).

[12] Zhishuai Guo, Rong Jin, Jiebo Luo, and Tianbao Yang. 2022. FeDXL: Provable Federated Learning for Deep X-Risk Optimization. *arXiv preprint arXiv:2210.14396* (2022).

[13] Zhishuai Guo, Mingrui Liu, Zhuoning Yuan, Li Shen, Wei Liu, and Tianbao Yang. 2020. Communication-efficient distributed stochastic auc maximization with deep neural networks. In *International Conference on Machine Learning*. PMLR, 3864–3874.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[15] Yifan Hu, Siqi Zhang, Xin Chen, and Niao He. 2020. Biased stochastic first-order methods for conditional stochastic optimization and applications in meta learning. *Advances in Neural Information Processing Systems* 33 (2020), 2759–2770.

[16] Yuelyu Ji, Yuhe Gao, Runxue Bao, Qi Li, Disheng Liu, Yiming Sun, and Ye Ye. 2023. Prediction of COVID-19 Patients' Emergency Room Revisit using Multi-Source Transfer Learning. In *2023 IEEE 11th International Conference on Healthcare Informatics (ICHI)*. IEEE.

[17] Wei Jiang, Gang Li, Yibo Wang, Lijun Zhang, and Tianbao Yang. 2022. Multi-block-Single-probe Variance Reduced Estimator for Coupled Compositional Optimization. *arXiv preprint arXiv:2207.08540* (2022).

[18] Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*.

[19] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 583–598.

[20] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems* 30 (2017).

[21] Mingrui Liu, Zhuoning Yuan, Yiming Ying, and Tianbao Yang. 2019. Stochastic auc maximization with deep neural networks. *arXiv preprint arXiv:1908.10831* (2019).

[22] Mingrui Liu, Wei Zhang, Youssef Mroueh, Xiaodong Cui, Jarret Ross, Tianbao Yang, and Payel Das. 2020. A decentralized parallel algorithm for training generative adversarial nets. *Advances in Neural Information Processing Systems* 33 (2020), 11056–11070.

[23] Songtao Lu, Xinwei Zhang, Haoran Sun, and Mingyi Hong. 2019. GNSD: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization. In *2019 IEEE Data Science Workshop (DSW)*. IEEE, 315–321.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR.

[25] Taoxing Pan, Jun Liu, and Jie Wang. 2020. D-SPIDER-SFO: A decentralized optimization algorithm with faster convergence rate for nonconvex problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1619–1626.

[26] Qi Qi, Youzhi Luo, Zhao Xu, Shuiwang Ji, and Tianbao Yang. 2021. Stochastic Optimization of Areas Under Precision-Recall Curves with Provable Convergence. *Advances in Neural Information Processing Systems* 34 (2021).

[27] Jianhui Sun, Mengdi Huai, Kishlay Jha, and Aidong Zhang. 2022. Demystify hyperparameters for stochastic optimization with transferable representations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1706–1716.

[28] Jianhui Sun, Ying Yang, Guangxu Xun, and Aidong Zhang. 2023. Scheduling Hyperparameters to Improve Generalization: From Centralized SGD to Asynchronous SGD. *ACM Transactions on Knowledge Discovery from Data* 17, 2 (2023).

[29] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. 2018. D^2: Decentralized training over decentralized data. In *International Conference on Machine Learning*. PMLR, 4848–4856.

[30] Bokun Wang and Tianbao Yang. 2022. Finite-Sum Coupled Compositional Stochastic Optimization: Theory and Applications. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 23292–23317.

[31] Guanghui Wang, Ming Yang, Lijun Zhang, and Tianbao Yang. 2021. Momentum Accelerates the Convergence of Stochastic AUPRC Maximization. *arXiv preprint arXiv:2107.01173* (2021).

[32] Yadi Wei, Rishit Sheth, and Roni Khardon. 2021. Direct loss minimization for sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2566–2574.

[33] Xidong Wu, Zhengmian Hu, and Heng Huang. 2023. Decentralized Riemannian Algorithm for Nonconvex Minimax Problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[34] Xidong Wu, Feihu Huang, Zhengmian Hu, and Heng Huang. 2022. Faster Adaptive Federated Learning. *arXiv preprint arXiv:2212.00974* (2022).

[35] Xidong Wu, Feihu Huang, and Heng Huang. 2022. Fast Stochastic Recursive Momentum Methods for Imbalanced Data Mining. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE.

[36] Yihan Wu, Aleksandar Bojchevski, and Heng Huang. 2022. Adversarial Weight Perturbation Improves Generalization in Graph Neural Network. *arXiv preprint arXiv:2212.04983* (2022).

[37] Yihan Wu, Hongyang Zhang, and Heng Huang. 2022. RetrievalGuard: Provably Robust 1-Nearest Neighbor Image Retrieval. In *International Conference on Machine Learning*. PMLR, 24266–24279.

[38] Wenhan Xian, Feihu Huang, Yanfu Zhang, and Heng Huang. 2021. A faster decentralized algorithm for nonconvex minimax problems. *Advances in Neural Information Processing Systems* 34 (2021), 25865–25877.

[39] Ran Xin, Usman Khan, and Soummya Kar. 2021. A hybrid variance-reduced method for decentralized stochastic non-convex optimization. In *International Conference on Machine Learning*. PMLR, 11459–11469.

[40] Yiming Ying, Longyin Wen, and Siwei Lyu. 2016. Stochastic online AUC maximization. *Advances in neural information processing systems* 29 (2016).

[41] Kun Yuan, Qing Ling, and Wotao Yin. 2016. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization* 26, 3 (2016), 1835–1854.

[42] Zhuoning Yuan, Zhishuai Guo, Nitesh Chawla, and Tianbao Yang. 2022. Compositional training for end-to-end deep AUC maximization. In *International Conference on Learning Representations*.

[43] Zhuoning Yuan, Zhishuai Guo, Yi Xu, Yiming Ying, and Tianbao Yang. 2021. Federated deep AUC maximization for hetergeneous data with a constant communication complexity. In *International Conference on Machine Learning*. PMLR, 12219–12229.

[44] Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth Serena Bentley. 2021. Gt-storm: Taming sample, communication, and memory complexities in decentralized non-convex learning. In *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. 271–280.

[45] Xin Zhang, Zhuqing Liu, Jia Liu, Zhengyuan Zhu, and Songtao Lu. 2021. Taming Communication and Sample Complexities in Decentralized Policy Evaluation for Cooperative Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021), 18825–18838.

[46] Xinwen Zhang, Yihan Zhang, Tianbao Yang, Richard Souvenir, and Hongchang Gao. 2023. Federated Compositional Deep AUC Maximization. *arXiv preprint arXiv:2304.10101* (2023).

[47] Yanfu Zhang, Runxue Bao, Jian Pei, and Heng Huang. 2022. Toward Unified Data and Algorithm Fairness via Adversarial Data Augmentation and Adaptive Model Fine-tuning. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE.

[48] Peilin Zhao, Steven CH Hoi, Rong Jin, and Tianbo YANG. 2011. Online AUC maximization. (2011).

Table 5: Model Architecture for the the MNIST dataset [34]

Layer Type	Shape
Convolution + ReLU	$5 \times 5 \times 20$
Max Pooling	2×2
Convolution + ReLU	$5 \times 5 \times 50$
Max Pooling	2×2
Fully Connected + ReLU	500
Fully Connected + ReLU	1

Table 6: Model Architecture for the Fashion MNIST dataset [34]

Layer Type	Shape
Convolution + ReLU	$3 \times 3 \times 5$
Max Pooling	2×2
Convolution + ReLU	$3 \times 3 \times 10$
Max Pooling	2×2
Fully Connected + ReLU	100
Fully Connected + ReLU	1

A SUPPLEMENTARY MATERIAL

A.1 Model Architecture

A.2 Basic Lemma

We draw one sample ξ_n from $\mathcal{D}_n^+$ and m sample ξ_n' from $\mathcal{D}_n$ as $\mathcal{B}_n$. We define

$$g_n(x;\xi_n) = \mathbb{E}_{\xi_n' \sim \mathcal{D}_n} g_n\left(x;\xi_n,\xi_n'\right)$$

$$\hat{g}_n(x,\xi_n) = \frac{1}{m}\sum_{\xi_n' \in \mathcal{B}_n} g_n(x,\xi_n;\xi_n')$$

$$\hat{F}_n(x;\xi_n,\mathcal{B}_n) = f(\hat{g}_n(x,\xi_n))$$

We also define

$$\begin{aligned}&\nabla F_n(x) = \nabla \mathbb{E}_{\xi_n} f(g_n(x,\xi_n)) = \mathbb{E}_{\xi_n}\left[\nabla\left(f(g_n(x,\xi_n))\right)\right]\\ =&\mathbb{E}_{\xi_n}\left[\nabla f(g_n(x,\xi_n))\cdot\nabla g_n(x,\xi_n)\right]\end{aligned}$$

$$\hat{F}_n(x) = \mathbb{E}\hat{F}_n(x;\xi_n,\mathcal{B}_n) = \mathbb{E}\left[f(\hat{g}_n(x,\xi_n))\right]$$

$$\nabla\hat{F}_n(x) = \mathbb{E}\nabla\hat{F}_n(x;\xi_n,\mathcal{B}_n) \tag{14}$$

For convenience, we denote

$$\bar{\mathbf{x}}_t = \frac{1}{N}\sum_{n=1}^N \mathbf{x}_{n,t},\quad \bar{\mathbf{v}}_t = \frac{1}{N}\sum_{n=1}^N \mathbf{v}_{n,t},\quad \bar{\mathbf{u}}_t = \frac{1}{N}\sum_{n=1}^N \mathbf{u}_{n,t}$$

$$F(\bar{\mathbf{x}}) = \frac{1}{N}\sum_{n=1}^N F_n(\bar{\mathbf{x}}), \nabla\hat{F}(\bar{\mathbf{x}}) = \frac{1}{N}\sum_{n=1}^N \nabla\hat{F}_n(\bar{\mathbf{x}}) \tag{15}$$

and $\nabla\hat{\mathbf{F}}_t = [\nabla\hat{F}_1(\mathbf{x}_1)^\top, \nabla\hat{F}_2(\mathbf{x}_2)^\top, \ldots, \nabla\hat{F}_N(\mathbf{x}_N)^\top]^\top \in \mathbb{R}^{nd}$.

Lemma 4. *(Lemma 6 in [39]) Let $\{V_t\}_{t\geq 0}, \{R_t\}_{t\geq 0}$ and $\{Q_t\}_{t\geq 0}$ be non-negative sequences and $C \geq 0$ be some constant such that $V_t \leq qV_{t-1} + qR_{t-1} + Q_t + C, \forall t \geq 1$, where $q \in (0,1)$ Then the following inequality holds: $\forall T \geq 1$,*

$$\sum_{t=0}^{T-1} V_t \leq \frac{V_0}{1-q} + \frac{1}{1-q}\sum_{t=0}^{T-2} R_t + \frac{1}{1-q}\sum_{t=1}^{T-1} Q_t + \frac{CT}{1-q} \tag{16}$$

B SLATE

Lemma 5. *Let Assumptions 1, 2 hold, and F is S_F-smooth, we have*

$$\begin{aligned}\mathbb{E}F(\bar{\mathbf{x}}_{t+1}) \leq \mathbb{E}F(\bar{\mathbf{x}}_t) - (\frac{\eta}{2} - \eta^2 S_F)\mathbb{E}\|\nabla\hat{F}(\bar{\mathbf{x}}_t)\|^2 - \frac{\eta}{2}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2\\ + \frac{\eta S_F^2}{N}\sum_{n=1}^N \|x_{n,t} - \bar{\mathbf{x}}_t\|^2 + \frac{\eta L_g^2 S_f^2 \sigma_g^2}{m} + \frac{\eta^2 S_F L_f^2 L_g^2}{N}\end{aligned} \tag{17}$$

Lemma 6. *Let Assumptions 1, 2 hold. We have: $\forall t \geq 0$,*

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{v}_{t+1} - \mathbf{J}\mathbf{v}_{t+1}\|^2\right] \leq \frac{1+\lambda^2}{2}\mathbb{E}\left[\|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2\right] + \frac{6\lambda^2\eta^2 S_F^2 N}{1-\lambda^2}\mathbb{E}\left\|\nabla\hat{F}_t\right\|^2\\ + \frac{24\lambda^2 S_F^2}{1-\lambda^2}\mathbb{E}\left[\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2\right] + \left(\frac{6\lambda^2\eta^2 S_F^2 N}{1-\lambda^2} + 3N + 2\right)L_f^2 L_g^2\end{aligned} \tag{18}$$

Based on previous lemmas, we start to prove the convergence of Theorem.

Proof. Recall Lemma 3, we have

$$\|\mathbf{x}_{t+1} - \mathbf{J}\mathbf{x}_{t+1}\|^2 \leq \frac{1+\lambda^2}{2}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{2\eta^2\lambda^2}{1-\lambda^2}\|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 \tag{19}$$

Putting (19) and lemma 6 into lemma 4, then we have

$$\sum_{t=0}^T \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 \leq \frac{4\lambda^2\eta^2}{(1-\lambda^2)^2}\sum_{t=0}^T \|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 \tag{20}$$

$$\begin{aligned}\sum_{t=0}^T \|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 \leq \frac{2}{1-\lambda^2}\|\mathbf{v}_0 - \mathbf{J}\mathbf{v}_0\|^2 + \left(\frac{6\lambda^2\eta^2 S_F^2}{1-\lambda^2} + 5N\right)\frac{2L_f^2 L_g^2 T}{1-\lambda^2}\\ + \frac{72\lambda^2 S_F^2}{(1-\lambda^2)^2}\sum_{t=0}^T \mathbb{E}\left[\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2\right] + \frac{24\lambda^2\eta^2 S_F^2 N}{(1-\lambda^2)^2}\sum_{t=0}^{t=T}\mathbb{E}\left\|\nabla\hat{F}(\bar{\mathbf{x}}_t)\right\|^2\end{aligned} \tag{21}$$

Then putting (20) into (21), we get

$$\begin{aligned}&\sum_{t=0}^T \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2\\ \leq&\frac{4\lambda^2\eta^2}{(1-\lambda^2)^2}\frac{2}{1-\lambda^2}\|\mathbf{v}_0 - \mathbf{J}\mathbf{v}_0\|^2 + \frac{96\lambda^4\eta^4 S_F^2 N}{(1-\lambda^2)^4}\sum_{t=0}^{t=T}\mathbb{E}\left\|\nabla\hat{F}(\bar{\mathbf{x}}_t)\right\|^2\\ &+ \frac{288\lambda^4\eta^2 S_F^2}{(1-\lambda^2)^4}\sum_{t=0}^T \mathbb{E}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \left(\frac{6\lambda^2\eta^2 S_F^2}{1-\lambda^2} + 5N\right)\frac{8\lambda^2\eta^2 L_f^2 L_g^2 T}{(1-\lambda^2)^3}\end{aligned}$$

Then we have

$$\begin{aligned}[1 - \frac{288\lambda^4\eta^2 S_F^2}{(1-\lambda^2)^4}]\sum_{t=0}^T \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 \leq \left(\frac{6\lambda^2\eta^2 S_F^2}{1-\lambda^2} + 5N\right)\frac{8\lambda^2\eta^2 L_f^2 L_g^2 T}{(1-\lambda^2)^3}\\ + \frac{96\lambda^4\eta^4 S_F^2 N}{(1-\lambda^2)^4}\sum_{t=0}^{t=T}\mathbb{E}\left\|\nabla\hat{F}(\bar{\mathbf{x}}_t)\right\|^2\end{aligned} \tag{22}$$

When $0 < \eta \leq \min\{\frac{1-\lambda^2}{24\lambda^2 S_F}, \frac{1}{6S_F}\}$, we have $[1 - \frac{192\lambda^4\eta^2 S_F^2}{(1-\lambda^2)^4}] \geq \frac{1}{2}$, Therefore, we have

$$\sum_{t=0}^{T} \|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 \leq \left(\frac{1}{\lambda^2} + 5N\right) \frac{16\lambda^2\eta^2 L_f^2 L_g^2 T}{(1-\lambda^2)^2} + \frac{192\lambda^4\eta^4 S_F^2 N}{(1-\lambda^2)^4} \sum_{t=0}^{t=T} \mathbb{E}\left\|\nabla \hat{F}_t\right\|^2 \quad (23)$$

Putting (23) into lemma 5, we have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 \leq \frac{2\mathbb{E}[F(\bar{\mathbf{x}}_0) - F(\bar{\mathbf{x}}_T)]}{\eta T} + \frac{2\eta S_F L_f^2 L_g^2}{N} \\
&+ \frac{2S_F^2}{NT}\sum_{t=0}^{T-1}\sum_{n=1}^{N}\|x_{n,t} - \bar{\mathbf{x}}_t\|^2 + \frac{2L_g^2 S_f^2 \sigma_g^2}{m} - \frac{(1-2\eta S_F)}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla \hat{F}(\bar{\mathbf{x}}_t)\|^2 \\
&\leq \frac{2\mathbb{E}[F(\bar{\mathbf{x}}_0) - F(\bar{\mathbf{x}}_T)]}{\eta T} - \frac{1}{T}[1 - 2\eta S_F - \frac{384\lambda^4\eta^4 S_F^4}{(1-\lambda^2)^4}]\sum_{t=0}^{T-1}\|\nabla \hat{F}(\bar{\mathbf{x}}_t)\|^2 \\
&+ \left(\frac{1}{\lambda^2} + 5N\right)\frac{32\lambda^2\eta^2 L_f^2 L_g^2 S_F^2}{(1-\lambda^2)^2 N} + \frac{2L_g^2 S_f^2 \sigma_g^2}{m} + \frac{2\eta S_F L_f^2 L_g^2}{N} \\
&\leq \frac{2\mathbb{E}[F(\bar{\mathbf{x}}_0) - F(\bar{\mathbf{x}}_T)]}{\eta T} + \left(\frac{1}{\lambda^2} + 5N\right)\frac{32\lambda^2\eta^2 L_f^2 L_g^2 S_F^2}{(1-\lambda^2)^2 N} + \frac{2L_g^2 S_f^2 \sigma_g^2}{m} \\
&+ \frac{2\eta S_F L_f^2 L_g^2}{N}
\end{aligned}$$

□

C PROOF OF SLATE-M ALGORITHM

Lemma 7. *Suppose the sequence $\{x_t\}_0^T$ are generated from SLATE-M in algorithm 2, we have*

$$\begin{aligned}
F(\bar{\mathbf{x}}_{t+1}) \leq F(\bar{\mathbf{x}}_t) - (\frac{\eta}{2} - \frac{\eta^2 S_F}{2})\|\bar{\mathbf{u}}_t\|^2 - \frac{\eta}{2}\|\nabla F(\bar{\mathbf{x}}_t)\|^2 + \frac{3\eta S_F^2}{2N}\|\mathbf{x}_t - \bar{\mathbf{x}}_t\|^2 \\
+ \frac{3\eta L_g^2 S_f^2 \sigma_g^2}{2m} + \frac{3\eta}{2}\|\frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}_n(\mathbf{x}_{n,t}) - \bar{\mathbf{u}}_t\|^2 \quad (24)
\end{aligned}$$

Lemma 8. *Assume that the stochastic partial derivatives u_t be generated from SLATE-M in Algorithm 2, we have*

$$\begin{aligned}
&\mathbb{E}\|\bar{\mathbf{u}}_{t+1} - \frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}_n(\mathbf{x}_{n,t+1})\|^2 \leq (1\alpha)^2\mathbb{E}\|\bar{\mathbf{u}}_t - \frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}_n(\mathbf{x}_{n,t})\|^2 \\
&+ \frac{2(1-\alpha)^2 S_F^2}{N^2 b}\mathbb{E}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 + \frac{2\alpha^2 L_g^2 L_f^2}{Nb} \\
&\mathbb{E}\|u_{n,t+1} - \nabla \hat{F}_n(\mathbf{x}_{n,t+1})\|^2 \leq (1-\alpha)^2\mathbb{E}\|\mathbf{u}_{n,t} - \nabla \hat{F}(\mathbf{x}_{n,t})\|^2 \\
&+ \frac{2(1-\alpha)^2 S_F^2}{b}\mathbb{E}\|x_{n,t+1} - \mathbf{x}_{n,t}\|^2 + \frac{2\alpha^2 L_g^2 L_f^2}{b} \quad (25)
\end{aligned}$$

Lemma 9. *Suppose sequence $\mathbf{v}_t$ are generated by Algorithm 2 and if $0 < \eta \leq \frac{1-\lambda^2}{2\sqrt{24}\lambda^2 S_F}$, we have*

$$\begin{aligned}
\mathbb{E}\|\mathbf{v}_{t+1} - \mathbf{J}\mathbf{v}_{t+1}\|^2 \leq \frac{3+\lambda^2}{4}\mathbb{E}\|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2 + \frac{21\lambda^2 N S_F^2 \eta^2}{1-\lambda^2}\mathbb{E}\|\bar{\mathbf{u}}_t\|^2 \\
+ \frac{63\lambda^2 S_F^2}{1-\lambda^2}\mathbb{E}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{7\lambda^2\alpha^2}{1-\lambda^2}\mathbb{E}\left\|\mathbf{u}_t - \nabla \hat{\mathbf{F}}_t\right\|^2 + 3\lambda^2 N\alpha^2 L_f^2 L_g^2
\end{aligned}$$

$$\mathbb{E}\|\mathbf{v}_0 - \mathbf{J}\mathbf{v}_0\|^2 \leq \lambda^2\mathbb{E}\left\|\mathbf{u}_0 - \nabla \hat{\mathbf{F}}_0\right\|^2 + \lambda^2\mathbb{E}\left\|\nabla \hat{\mathbf{F}}_0\right\|^2$$

Then we start the proof of Theorem 2.

Proof. Recall that

$$\begin{aligned}
&\mathbb{E}\|\bar{\mathbf{u}}_{t+1} - \frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}(x_{n,t+1})\|^2 \leq (1-\alpha)^2\mathbb{E}\|\bar{\mathbf{u}}_t - \frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}(\mathbf{x}_{n,t})\|^2 \\
&+ \frac{2(1-\alpha)^2 S_F^2}{N^2 b}\mathbb{E}\|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2 + \frac{2\alpha^2 L_g^2 L_f^2}{Nb} \quad (26)
\end{aligned}$$

We know that $\frac{1}{1-(1-\alpha)^2} \leq \frac{1}{\alpha}$ for $\alpha \in (0,1)$. Based on Lemma 4, we have: $\forall T \geq 2$,

$$\begin{aligned}
&\sum_{t=0}^{T-1}\mathbb{E}\left\|\bar{\mathbf{u}}_t - \frac{1}{N}\sum_{n=1}^{N}\nabla \hat{F}(x_{n,0})\right\|^2 \quad (27) \\
&\leq \frac{L_g^2 L_f^2}{\alpha NB} + \frac{12S_F^2}{N^2\alpha b}\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{6\eta^2 S_F^2}{N\alpha b}\sum_{t=0}^{T-1}\|\bar{\mathbf{u}}_t\|^2 + \frac{2\alpha L_g^2 L_f^2}{Nb}T
\end{aligned}$$

where B is the initial batch size. Similarly, we have the following: $\forall T \geq 2$,

$$\begin{aligned}
&\sum_{n=1}^{N}\sum_{t=0}^{T-1}\mathbb{E}\left\|\mathbf{u}_{n,t} - \nabla \hat{F}_n(\mathbf{x}_{n,t})\right\|^2 \quad (28) \\
&\leq \frac{NL_f^2 L_g^2}{\alpha B} + \frac{6NS_F^2\eta^2}{\alpha}\sum_{t=0}^{T-2}\mathbb{E}\|\bar{\mathbf{u}}_t\|^2 + \frac{12S_F^2}{\alpha b}\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{2\alpha NL_g^2 L_f^2}{b}T \\
&\sum_{t=0}^{T-1}\mathbb{E}\left[\|\mathbf{v}_t - \mathbf{J}\mathbf{v}_t\|^2\right] \\
&\leq \frac{4\lambda^2\mathbb{E}\left\|\nabla \hat{\mathbf{F}}_0\right\|^2}{1-\lambda^2} + \frac{4\lambda^2 NL_f^2 L_g^2}{\left(1-\lambda^2\right)B} + \frac{84\lambda^2 NS_F^2\eta^2}{\left(1-\lambda^2\right)^2}\sum_{t=0}^{T-1}\mathbb{E}\|\bar{\mathbf{u}}_t\|^2 \\
&+ \frac{252\lambda^2 S_F^2}{\left(1-\lambda^2\right)^2}\sum_{t=0}^{T-1}\mathbb{E}\|\mathbf{x}_t - \mathbf{J}\mathbf{x}_t\|^2 + \frac{12\lambda^2 N\alpha^2 L_f^2 L_g^2 T}{1-\lambda^2} \\
&+ \frac{28\lambda^2\alpha^2}{\left(1-\lambda^2\right)^2}\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}\left\|\mathbf{u}_{n,t} - \nabla \hat{\mathbf{F}}_n\left(\mathbf{x}_{n,t}\right)\right\|^2
\end{aligned}$$

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2 \overset{(d)}{\leq} \frac{2(\mathbf{F}(\bar{\mathbf{x}}_0) - \mathbf{F}(\bar{\mathbf{x}}_T))}{\eta T} + \frac{3L_g^2 S_f^2 \sigma_g^2}{m} + 3\frac{L_g^2 L_f^2}{\alpha NBT} \\
&+ \frac{6\alpha L_g^2 L_f^2}{Nb} + \frac{96\lambda^2 L_g^2 L_f^2}{\left(1-\lambda^2\right)^3 BT} + \frac{256\lambda^2\alpha^2 L_f^2 L_g^2}{(1-\lambda^2)^3} + \frac{64\lambda^4\mathbb{E}\left\|\nabla \hat{\mathbf{F}}_0\right\|^2}{(1-\lambda^2)^3 NT}
\end{aligned}$$

□

Then, we choose $b = O(1), \eta = O(\frac{N^{2/3}}{T^{1/3}}), \alpha = \frac{N^{1/3}}{T^{2/3}}, B = \frac{T^{1/3}}{N^{2/3}}$

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla \mathbf{F}(\bar{\mathbf{x}}_t)\|^2 \leq O(\frac{2(\mathbf{F}(\bar{\mathbf{x}}_0) - \mathbf{F}(\bar{\mathbf{x}}_T))}{(NT)^{2/3}} + \frac{3L_g^2 S_f^2 \sigma_g^2}{m} \\
&+ O(\frac{3L_g^2 L_f^2}{(NT)^{2/3}}) + O(\frac{6L_g^2 L_f^2}{(NT)^{2/3}}) + \frac{352\lambda^2 L_f^2 L_g^2}{(1-\lambda^2)^3}O(\frac{N^{2/3}}{T^{4/3}}) + \frac{64\lambda^4\mathbb{E}\left\|\nabla \hat{\mathbf{F}}_0\right\|^2}{(1-\lambda^2)^3 NT}
\end{aligned}$$