Communication-Efficient Robust Federated Learning with Noisy Labels

Junyi Li
Electrical and Computer Engineering
University of Pittsburgh
junyili.ai@gmail.com

Jian Pei
School of Computing Science
Simon Fraser University
jpei@cs.sfu.ca

Heng Huang*
Electrical and Computer Engineering
University of Pittsburgh
henghuanghh@gmail.com

ABSTRACT

Federated learning (FL) is a promising privacy-preserving machine learning paradigm over distributed located data. In FL, the data is kept locally by each user. This protects the user privacy, but also makes the server difficult to verify data quality, especially if the data are correctly labeled. Training with corrupted labels is harmful to the federated learning task; however, little attention has been paid to FL in the case of label noise. In this paper, we focus on this problem and propose a learning-based reweighting approach to mitigate the effect of noisy labels in FL. More precisely, we tuned a weight for each training sample such that the learned model has optimal generalization performance over a validation set. More formally, the process can be formulated as a Federated Bilevel Optimization problem. Bilevel optimization problem is a type of optimization problem with two levels of entangled problems. The non-distributed bilevel problems have witnessed notable progress recently with new efficient algorithms. However, solving bilevel optimization problems under the Federated Learning setting is under-investigated. We identify that the high communication cost in hypergradient evaluation is the major bottleneck. So we propose *Comm-FedBiO* to solve the general Federated Bilevel Optimization problems; more specifically, we propose two communication-efficient subroutines to estimate the hypergradient. Convergence analysis of the proposed algorithms is also provided. Finally, we apply the proposed algorithms to solve the noisy label problem. Our approach has shown superior performance on several real-world datasets compared to various baselines.

CCS CONCEPTS

• **Computing methodologies** → *Supervised learning*.

KEYWORDS

Data Cleaning, Federated Learning, Bilevel Optimization

ACM Reference Format:
Junyi Li, Jian Pei, and Heng Huang. 2022. Communication-Efficient Robust Federated Learning with Noisy Labels. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539328

*This work was partially supported by NSF IIS 1845666, 1852606, 1838627, 1837956, 1956002, IIA 2040588.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539328

1 INTRODUCTION

In Federated Learning (FL) [35], a set of clients jointly solve a machine learning problem under the coordination of a central server. To protect privacy, clients keep their own data locally and share model parameters periodically with each other. Several challenges of FL are widely studied in the literature, such as user privacy [35, 39, 53], communication cost [17, 21, 31, 50, 55], data heterogeneity [5, 13, 20, 29, 54] *etc.*. However, a key challenge is ignored in the literature: *the label quality of user data.* Data samples are manually annotated, and it is likely that the labels are incorrect. However, existing algorithms of FL *e.g.* FedAvg [35] treat every sample equally; as a result, the learned models overfit the label noise, leading to bad generalization performance. It is challenging to develop an algorithm that is robust to the noise from the labels. Due to privacy concerns, user data are kept locally, so the server cannot verify the quality of the label of the user data. Recently, several intuitive approaches [8, 52, 59] based on the use of a clean validation set have been proposed in the literature. In this paper, we take a step forward and formally formulate the noisy label problem as a bilevel optimization problem; furthermore, we provide two efficient algorithms that have guaranteed convergence to solve the optimization problem.

The basic idea of our approach is to identify noisy label samples based on its contribution to training. More specifically, we measure the contribution through the Shapley value [46] of each sample. Suppose that we have the training dataset $\mathcal{D}$ and a sample $s \in \mathcal{D}$. Then for any subset $\mathcal{S} \subset \mathcal{D}/\{s\}$, we first train a model with $\mathcal{S}$ only and measure the generalization performance of the learned model, then train over $S \cup \{s\}$ and calculate the generalization performance again. The difference in generalization performance of the two models reflects the quality of the sample label. If a sample has a correct label, the model will have better generalization performance when the sample is included in the training; in contrast, a mislabeled sample harms the generalization performance. Then we define the Shapley value of any sample as the average of the generalization performance difference in all possible subsets $\mathcal{S}$. However, the Shapley value of a sample is NP-hard to compute. As an alternative, we define a weight for each sample and turn the problem into finding weights that lead to optimal generalization performance. With this reformulation, we need to solve a bilevel optimization problem. Bilevel optimization problems [44, 48, 56] involve two levels of problems: an inner problem and an outer problem. Efficient gradient-based alternative update algorithms [15, 18, 26] have recently been proposed to solve non-distributed bilevel problems,

but efficient algorithms designed for the FL setting have not yet been shown. In fact, the most challenging step is to evaluate the hypergradient (gradient *w.r.t* the variable of the outer problem). In FL, hypergradient evaluation involves transferring the Hessian matrix, which leads to high communication cost. It is essential to develop a communication-efficient algorithm to evaluate the hypergradient and solve the Federated Bilevel Optimization problem efficiently.

More specifically, we propose two compression algorithms to reduce the communication cost for the hypergradient estimation: an iterative algorithm and a non-iterative algorithm. In the non-iterative algorithm, we compress the Hessian matrix directly and then solve a small linear equation to get the hypergradient. In the iterative algorithm, we formulate the hypergradient evaluation as solving a quadratic optimization problem and then run an iterative algorithm to solve this quadratic problem. To further save communication, we also compress the gradient of the quadratic objective function. Both the non-iterative and iterative algorithms effectively reduce the communication overhead of hypergradient evaluation. In general, the non-iterative algorithm requires communication cost polynomial to the stable rank of the Hessian matrix, and the iterative algorithm requires $O(log(d))$ (d is the dimension of the model parameters). Finally, we apply the proposed algorithms to solve the noisy label problem on real-world datasets. Our algorithms have shown superior performance compared to various baselines. We highlight the contribution of this paper below.

(1) We study Federated Learning with noisy labels problems and propose a learning-based data cleaning procedure to identify mislabeled data.
(2) We formalize the procedure as a Federated Bilevel Optimization problem. Furthermore, we propose two novel efficient algorithms based on compression, *i.e.* the Iterative and Non-iterative algorithms. Both methods reduce the communication cost of the hypergradient evaluation from $O(d^2)$ to be sub-linear of d.
(3) We show that the proposed algorithms have a convergence rate of $O(\epsilon^{-2})$ and validate their efficacy by identifying mislabeled data in real-world datasets.

Notations. ∇ denotes the full gradient, ∇_x is the partial derivative for variable x, and higher-order derivatives follow similar rules. $||\cdot||$ is ℓ_2-norm for vectors and the spectral norm for matrices. $||\cdot||_F$ represents the Frobenius norm. AB denotes the multiplication of the matrix between the matrix A and B. $\binom{n}{k}$ denotes the binomial coefficient. $[K]$ represents the sequence of integers from 1 to K.

2 RELATED WORKS

Federated Learning. FL is a promising paradigm for performing machine learning tasks on distributed located data. Compared to traditional distributed learning in the data center, FL poses new challenges such as heterogeneity [16, 20, 28, 38, 45], privacy [35, 39, 53] and communication bottleneck [17, 21, 31, 50, 55]. In addition, another challenge that receives little attention is the noisy data problem. Learning with noisy data, especially noisy labels, has been widely studied in a non-distributed setting [2–4, 37, 40, 41, 47, 51]. However, since the server cannot see the clients' data and the communication is expensive between the server and clients, algorithms developed in the non-distributed setting can not be applied to the Federated Learning setting. Recently, several works [8, 52, 59] have focused on FL with noisy labels. In [8], authors propose FOCUS: The server defines a credibility score for each client based on the mutual cross-entropy of two losses: the loss of the global model evaluated on the local dataset and the loss of the local model evaluated on a clean validation set. The server then uses this score as the weight of each client during global averaging. In [52], the server first trains a benchmark model, and then the clients use this model to exclude possibly corrupted data samples.

Gradient Compression. Gradient compression is widely used in FL to reduce communication costs. Existing compressors can be divided into quantization-based [31, 55] and sparsification-based [21, 50]. Quantization compressors give an unbiased estimate of gradients, but have a high variance [17]. In contrast, sparsification methods generate biased gradients, but have high compression rate and good practical performance. The error feedback technique [21] is combined with sparsification compressors to reduce compression bias. Sketch-based compression methods [17, 43] are one type of sparsification compressor. Sketching methods [1] originate from the literature on streaming algorithms, *e.g.* The Count-sketch [7] compressor was proposed to efficiently count heavy hitters in a data stream.

Bilevel Optimization. Bilevel optimization [56] has gained more interest recently due to its application in many machine learning problems such as hyperparameter optimization [33], meta learning [60], neural architecture search [32] *etc.* Various gradient-based methods are proposed to solve the bilevel optimization problem. Based on different approaches to the estimation of hypergradient, these methods are divided into two categories, *i.e.* Approximate Implicit Differentiation (AID) [11, 14, 15, 18, 22, 26, 58] and Iterative Differentiation (ITD) [9, 10, 34, 42]. ITD methods first solve the lower level problem approximately and then calculate the hypergradient with backward (forward) automatic differentiation, while AID methods approximate the exact hypergradient [11, 19, 30, 33]. In [12], authors compare these two categories of methods in terms of their hyperiteration complexity. Finally, there are also works that utilize other strategies such as penalty methods [36], and also other formulations *e.g.* the inner problem has multiple minimizers [25, 49]. A recent work [27] applied momentum-based acceleration to solve federated bilevel optimization problems.

3 PRELIMINARIES

Federated Learning. A general formulation of Federated Learning problems is:

$$\min_{x\in X} G(x) := \frac{1}{N}\sum_{i=1}^{M} N_i g_i(x) \tag{1}$$

There are M clients and one server. N_i is the number of samples in the i_{th} client and N is the total number of samples. g_i denotes the objective function on the i_{th} client. To reduce communication cost, a common approach is to perform local sgd; in other words, the client performs multiple update steps with local data, and the model averaging operation occurs every few iterations. A widely used algorithm that uses this approach is FedAvg [35].

Count Sketch. The count-sketch technique [7] was originally proposed to efficiently count heavy-hitters in a data stream. Later, it

was applied in gradient compression: It is used to project a vector into a lower-dimensional space, while the large-magnitude elements can still be recovered. To compress a vector $g \in \mathbb{R}^d$, it maintains counters $r \times c$ denoted as S. Furthermore, it generates sign and bucket hashes $\{h_j^s, h_j^b\}_{j=1}^r$. In the compression stage, for each element $g_i \in g$, it performs the operation $S[j, h_j^b(i)]$ += $h_j^s[i] * g_i$ for $j \in [r]$. In the decompression stage, it recovers g_i as median$(\{h_j^s[i] * S[j, h_j^b(i)]\}_{j=1}^r)$. To recover τ heavy-hitters (elements g_i where $||g_i||^2 \geq \tau||g||^2$) with probability at least $1-\delta$, the count sketch needs $r \times c$ to be $O(\tau^{-1}\log(d/\delta))$. More details of the implementation are provided in [7].

Bilevel Optimization. A bilevel optimization problem has the following form:

$$\min_{x \in \mathcal{X}} h(x) \coloneqq F(x, y_x) \ s.t. \ y_x = \arg\min_{y \in \mathbb{R}^d} G(x, y) \tag{2}$$

As shown in Eq. (2), a bilevel optimization problem includes two entangled optimization problems: the outer problem $F(x, y_x)$ and the inner problem $G(x, y)$. The outer problem relies on the minimizer y_x of the inner problem. Eq. (2) can be solved efficiently through gradient-based algorithms [18, 26]. There are two main categories of methods for hypergradient (the gradient *w.r.t* the outer variable x) evaluation: Approximate Implicit Differentiation (AID) and Iterative Differentiation (ITD). The ITD is based on automatic differentiation and stores intermediate states generated when we solve the inner problem. ITD methods are not suitable for the Federated Learning setting, where clients are stateless and cannot maintain historical inner states. In contrast, the AID approach is based on an explicit form of the hypergradient, as shown in Proposition 1:

Proposition 1. *(hypergradient) When y_x is uniquely defined and $\nabla^2_{yy} G(x, y_x)$ is invertible, the hypergradient has the following form:*

$$\nabla h(x) = \nabla_x F(x, y_x) - \nabla^2_{xy} G(x, y_x) v^* \tag{3}$$

where v^* is the solution of the following linear equation:

$$\nabla^2_{yy} G(x, y_x) v^* = \nabla_y F(x, y_x) \tag{4}$$

The proposition 1 is based on the chain rule and the implicit function theorem. The proof of Proposition 1 can be found in the bilevel optimization literature, such as [11].

4 FEDERATED LEARNING WITH NOISY LABELS

We consider the Federated Learning setting as shown in Eq. (1), *i.e.* a server and a set of clients. For ease of discussion, we assume that the total number of clients is M and that each client has a private data set $\mathcal{D}_i = \{s_j^i, j \in [N_i]\}$, $i \in [M]$ where $|\mathcal{D}_i| = N_i$. $\mathcal{D}$ denotes the union of all client datasets: $\mathcal{D} = \bigcup_{i=1}^M \mathcal{D}_i$. The total number of samples $|\mathcal{D}| = N$ and $N = \sum_{i=1}^M N_i$ (for simplicity, we assume that there is no overlap between the client data sets).

In Federated Learning, the local dataset $\mathcal{D}_i$ is not shared with other clients or the server; this protects the user privacy, but also makes the server difficult to verify the quality of data samples. A data sample can be corrupted in various ways; we focus on the noisy label issue. Current Federated Learning models are very sensitive to the label noise in client datasets. Take the widely used FedAvg [35] as an example; the server simply performs a weighted average on the client models in the global averaging step. As a result, if one client model is affected by mislabeled data, the server model will also be affected. To eliminate the effect of these corrupted data samples on training, we can calculate the contribution of each sample. Based on the contribution, we remove samples that have little or even negative contributions. More specifically, we define the following metric of sample contribution based on the idea of Shapley value [46]:

$$\phi_j^i = \frac{1}{N} \sum_{\mathcal{S} \subset \mathcal{D}/\{s_j^i\}} \binom{N-1}{|\mathcal{S}|}^{-1} \left(\Phi\left(\mathcal{A}\left(\mathcal{S} \cup \{s_j^i\}\right)\right) - \Phi\left(\mathcal{A}\left(\mathcal{S}\right)\right) \right) \tag{5}$$

where $\mathcal{A}$ is a randomized algorithm (*e.g.* FedAvg) that takes the dataset $\mathcal{S}$ as input and outputs a model. Φ is a metric of model gain, *e.g.* negative population loss of the learned model, or negative empirical loss of the learned model in a validation set. In Eq. (5), for each $S \subset \mathcal{D}/\{s_j^i\}$, we calculate the marginal gain when s_j^i is added to the training and then average over all such subsets $\mathcal{S}$. It is straightforward to find corrupted data if we can compute ϕ_j^i, however, the evaluation of ϕ_j^i is NP-hard. As an alternative, we define the weight $\lambda_j^i \in [0, 1]$ for each sample and λ_j^i should be positively correlated with the sample contribution ϕ_j^i: large λ represents a high contribution and small λ means little contribution. In fact, finding sample weights that reflect the contribution of a data sample can be formulated as solving the following optimization problem:

$$\max_{\lambda \in \Lambda} \Phi(\mathcal{A}(\mathcal{D}; \lambda)) \tag{6}$$

The above optimization problem can be interpreted as follows: The sample weights should be assigned so that the gain of the model Φ is maximized. It is then straightforward to see that only samples that have large contributions will be assigned with large weights. Next, we consider an instantiation of Eq. (6). Suppose that we choose Φ as the negative empirical loss over a validation set D_{val} at the server and $\mathcal{A}$ fits a model parameterized by ω over the data, then Eq. (6) can be written as:

$$\min_{\lambda \in \Lambda} \ell(\omega_\lambda; \mathcal{D}_{val}) \ s.t. \ \omega_\lambda = \arg\min_{\omega \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^M \sum_{j=1}^{N_i} \lambda_j^i \ell(\omega; s_j^i) \tag{7}$$

where ℓ is the loss function *e.g.* the cross entropy loss. Eq. (7) involves two entangled optimization problems: an outer problem and an inner problem, and ω_λ is the minimizer of the inner problem. This type of optimization problem is known as Bilevel Optimization Problems [56] as we introduce in the preliminary section. Following a similar notation as in Eq. (2), we write Eq. (7) in a general form:

$$\begin{aligned} &\min_{x \in \mathcal{X}} h(x) \coloneqq F(x, y_x) \\ &\quad s.t. \ y_x = \arg\min_{y \in \mathbb{R}^d} G(x, y) \coloneqq \frac{1}{N} \sum_{i=1}^M N_i g_i(x, y) \end{aligned} \tag{8}$$

Compared to Eq. (7), we set λ as x, ω as y; $\ell(\omega_\lambda; \mathcal{D}_{val})$ as $F(x, y_x)$, and $1/N_i \sum_{j=1}^{N_i} \lambda_j^i \ell(\omega; s_j^i)$ as $g_i(x, y)$. In the remainder of this section, our discussion will be based on the general formulation (8).

Algorithm 1 Communication-Efficient Federated Bilevel Optimization (**Comm-FedBiO**)

1: **Input:** Learning rate η, γ, initial state (x_0, y_0), number of sampled clients S
2: **for** $k = 0$ **to** $K-1$ **do**
3: Sample S clients and broadcast current model state (x_k, y_k);
4: **for** $m = 1$ to S clients in parallel **do**
5: Set $y_0^m = y_k$
6: **for** $t = 1$ to T **do**
7: $y_{t+1}^m = y_t^m - \gamma \nabla g_m(x_k, y_t^m)$
8: **end for**
9: **end for**
10: $y_{k+1} = y_k + \frac{1}{\sum_{m=1}^S N_m} \sum_{m=1}^S N_m (y_T^m - y_k)$
// Two ways to estimate $\hat{\nabla} h(x_k)$
11: Case 1: $\hat{\nabla} h(x_k)$ = Iterative-approx(x_k, y_{k+1})
12: Case 2: $\hat{\nabla} h(x_k)$ = Non-iterative-approx(x_k, y_{k+1})
13: $x_{k+1} = x_k - \eta \hat{\nabla} h(x_k)$
14: **end for**

We propose the algorithm **Comm-FedBiO** to solve Eq. (8) (Algorithm 1). Algorithm 1 follows the idea of alternative update of inner and outer variables in non-distributed bilevel optimization [15, 19, 26], however, it has two key innovations which are our contributions. First, since the inner problem of Eq. (8) is a federated optimization problem, we perform local sgd steps to save the communication. Next, we consider the communication constraints of federated learning in the hypergradient estimation. More specifically, we propose two communication-efficient hypergradient estimators, *i.e.* the subroutine *Non-iterative-approx* (line 11) and *Iterative-approx* (line 12).

To see the high communication cost caused by hypergradient evaluation. We first write the hypergradient based on Proposition 1,:

$$\begin{aligned} \nabla h(x) =& \nabla_x F(x, y_x) - \nabla_{xy}^2 G(x, y_x) v^* \\ v^* =& \left(\sum_{i=1}^M \frac{N_i}{N} \nabla_{yy}^2 g_i(x, y) \right)^{-1} \nabla_y F(x, y_x) \end{aligned} \tag{9}$$

where we use the explicit form of v^* and replace $G(x, y)$ with the federated form in Eq. (8). Eq. (9) includes two steps: calculating v^* and evaluating $\nabla h(x)$ based on v^*. For the second step, clients transfer $\nabla_{xy} g_i(x, y) v^*$ with communication cost $O(l)$ (the dimension of the outer variable x), we assume $l < d$ (d is the dimension of y). This is reasonable in our noisy label application: l is equal to the number of samples at a client and is very small in Federated Learning setting, while d is the weight dimension, which can be very large. Therefore, we focus on the first step when considering the communication cost. The inverse of the Hessian matrix in Eq. (9) can be approximated with various techniques such as the Neumann series expansion [11] or conjugate gradient descent [19]. However, clients must first exchange the Hessian matrix $\nabla_{yy}^2 g_i(x, y)$. This leads to a communication cost on the order of $O(d^2)$. In fact, it is not necessary to transfer the full Hessian matrix, and we can reduce the cost through compression. More specifically, we can exploit the sparse structure of the related properties; *e.g.* The Hessian matrix has only a few dominant singular values in practice. In Sections 4.1 and 4.2, we propose two communication-efficient estimators of hypergradient $\nabla h(x_k)$ based on this idea. In general, our estimators can be evaluated with the communication cost sublinear to the parameter dimension d.

Algorithm 2 Iterative Approximation of hypergradient (**Iterative-approx**)

1: **Input:** State (x, y), initial value v_0, learning rate α, number of sampled clients S
2: The server evaluates $\nabla_y F(x, y)$ and samples S clients uniformly and broadcasts state (x, y) to each client;
3: **for** $i = 0$ to $I-1$ **do**
4: **for** $m = 0$ to S in parallel **do**
5: Each client makes Hessian-vector product queries to compute $\nabla_{yy}^2 g_m v^i$, then send its sketch $S_{g_m}^i$ to the server;
6: **end for**
7: **Server:** $S_G^i = \frac{1}{\sum_{m=1}^S N_m} \sum_{m=1}^S N_m * S_{g_m}^i$
8: **Server:** $\Delta = U(\alpha S_G^i + S(e^i))$
9: **Server:** $v^{i+1} = v^i - (\Delta - \alpha \nabla_y F(x, y))$
10: **Server:** $S(e^{i+1}) = \alpha S_G^i + S(e^i) - S(\Delta)$
11: **end for**
12: **Output:** $\hat{\nabla} h(x) = \nabla_x F - \nabla_{xy}^2 G v^I$

4.1 hypergradient Estimation with iterative algorithm

In this section, we introduce an iterative hypergradient estimator. Instead of performing the expensive matrix inversion as in Eq. (9), we solve the following quadratic optimization problem:

$$\min_v q(v) := \frac{1}{2} v^T \nabla_{yy}^2 G(x, y_x) v - v^T \nabla_y F(x, y_x) \tag{10}$$

The equivalence is observed by noticing that:

$$\nabla q(v) = \nabla_{yy}^2 G(x, y_x) v - \nabla_y F(x, y_x)$$

If $q(v)$ is strongly convex ($\nabla_{yy}^2 G(x, y_x)$ is positive definite), the unique minimizer of the quadratic function $q(v)$ is exactly v^* as shown in Eq. (9). Eq. (10) is a simple positive definite quadratic optimization problem and can be solved with various iterative gradient-based algorithms. To further reduce communication cost, we compress the gradient $\nabla q(v)$. More specifically, $\nabla q(v)$ can be expressed as follows in terms of g_i:

$$\nabla q(v) = \frac{1}{N} \sum_{i=1}^M N_i \nabla_{yy}^2 g_i(x, y_x) v - \nabla_y F(x, y_x) \tag{11}$$

Therefore, clients must exchange the Hessian vector product to evaluate $\nabla q(v)$. This operation has a communication cost $O(d)$. This cost is considerable when we evaluate $\nabla q(v)$ multiple times to optimize Eq.(10). Therefore, we exploit compression to further reduce communication cost; *i.e.* clients only communicate the compressed Hessian vector product. Various compressors can be used for compression. In our paper, we consider the local Topk compressor and the Count Sketch compressor [7] in our paper. The local Top-k compressor is simple to implement, but it cannot recover the global Top-k coordinates, while the count-sketch is more complicated to implement, but it can recover the global Top-k coordinates under

certain conditions. In general, gradient compression includes two phases: compression at clients and decompression at the server. In the first phase, clients compress the gradient to a lower dimension with the compressor $S(\cdot)$, then transfer the compressed gradient to the server; In the second phase, the server aggregates the compressed gradients received from clients and decompresses them to recover an approximation of the original gradients. We denote the decompression operator as $U(\cdot)$. Then the update step of a gradient descent method with compression is as follows:

$$\begin{aligned} v^{i+1} &= v^i - C(\alpha \nabla q(v^i) + e^i), \\ e^{i+1} &= \alpha \nabla q(v^i) + e^i - C(\alpha \nabla q(v^i) + e^i) \end{aligned} \tag{12}$$

where $C(\cdot) \coloneqq U(S(\cdot))$ and α is the learning rate. Notice that we add an error accumulation term e^i. As shown by the update rule of e^i, it accumulates information that cannot be transferred due to compression and reintroduces information later in the iteration, this type of error feedback trick compensates for the compression error and is crucial for convergence. The update rule in Eq. (12) has communication cost sub-linear *w.r.t* parameter dimension d with either the Top-k compressor or the Count-sketch compressor. Furthermore, the iteration complexity of iterative algorithms is independent of the problem dimension, the overall communication cost of evaluating v^* is still sublinear *w.r.t* the dimension d. This is a great reduction compared to the $O(d^2)$ complexity when we transfer the Hessian directly as in Eq. (9). We term this hypergradient approximation approach the iterative algorithm, and the pseudocode is shown in Algorithm 2. Note that the server can evaluate $\nabla_y F(x, y)$, so we do not need to compress it, and we also omit the step of getting $\nabla^2_{xy} G v^I$ from the clients.

4.2 hypergradient Estimation with Non-iterative algorithm

In this section, we propose an efficient algorithm so that we can estimate v^* by solving the linear equation Eq. (4) directly. However, instead of transferring $\nabla^2_{yy} g_i(x, y_x)$, we transfer their sketch. More precisely, we solve the following linear equation:

$$S_2 \nabla^2_{yy} G(x, y_x) S_1^T \omega = S_2 \nabla_y F(x, y_x) \tag{13}$$

where $\hat{\omega} \in \mathbb{R}^{r_1}$ denotes the solution of Eq. (13). $S_1 \in \mathbb{R}^{r_1 \times d}$ and $S_2 \in \mathbb{R}^{r_2 \times d}$ are two random matrices. Then an approximation of the hypergradient $\nabla h(x)$ is:

$$\hat{\nabla} h(x) = \nabla_x F(x, y_x) - \nabla^2_{xy} G(x, y_x) S_1^T \hat{\omega} \tag{14}$$

To solve Eq. (13), clients first transfer $S_2 \nabla^2_{yy} g_i(x, y_x) S_1^T$ to the server with communication cost $O(r_1 r_2)$, then the server solves the linear system (13) locally. The server then transfers $\hat{\omega}$ to the clients and the clients transfer $\nabla_{xy} g_i(x, y_x) S_1^T \hat{\omega}$ back to the server, the server evaluates Eq. (14) to get $\hat{\nabla} h(x)$. The total communication cost is $O(r_1 r_2)$ (we assume that the dimension of the outer variable x is small).

We require S_1 and S_2 to have the following two properties: the approximation error $||\hat{\nabla} h(x) - \nabla h(x)||$ is small and the communication cost is much lower than $O(d^2)$, *i.e.* $r_1 r_2 \ll d^2$. We choose S_1 and S_2 as the following sketch matrices:

Algorithm 3 Non-iterative approximation of hypergradient (**Non-iterative-approx**)

1: **Input:** State (x, y), random seeds τ_1, τ_2, number of rows r_1, r_2, number of sampled clients S
2: **Server:** Sample S clients uniformly and broadcast model state (x, y) to each sampled client
3: **for** $m = 1$ to S in parallel **do**
4: Each client generates S_1, S_2 with random seeds τ_1, τ_2, compute $S_2 \nabla^2_{yy} g_j S_1^T$ and $\nabla^2_{xy} g_j S_1^T$ with Hessian-vector product queries
5: **end for**
6: **Server:** Collect and average sketches from clients to get $S_2 \nabla^2_{yy} G S_1^T$ and $\nabla^2_{xy} G S_1^T$ and solves Eq. (13) with linear regression to get $\hat{\omega}$
7: **Output:** $\hat{\nabla} h(x) = \nabla_x F - \nabla^2_{xy} G S_1^T \hat{\omega}$

Definition 4.1. *A distribution $\mathcal{D}$ on the matrices $S \in \mathbb{R}^{r \times n}$ is said to generate a (ϵ, δ)-sketch matrix for a pair of matrices A, B with n rows if:*

$$\Pr_{S \sim \mathcal{D}}[||A^T S^T S B - A^T B|| > \epsilon ||A||_F ||B||_F] \leq \delta$$

Corollary 2. *An $(\epsilon/l, \delta)$ sketch matrix S is a subspace embedding matrix for the column space of $A \in R^{n \times l}$.* i.e. *for all $x \in \mathbb{R}^l$* w.p. *at least $1 - \delta$:*

$$||SAx||_2^2 \in [(1-\epsilon)||Ax||_2^2, (1+\epsilon)||Ax||_2^2]$$

Corollary 3. *For any $\epsilon, \delta \in (0, 1/2)$, let $S \in \mathbb{R}^{r \times n}$ be a random matrix with $r > 18/(\epsilon^2 \delta)$. Furthermore, suppose that $\sigma \in \mathbb{R}^n$ is a random sequence where $\sigma(i)$ is randomly chosen from $\{-1, 1\}$ and $h \in \mathbb{R}^n$ is another random sequence where $h(i)$ is randomly chosen from $[r]$. Suppose that we set $S[h(i), i] = \sigma(i)$, for $i \in [n]$ and 0 for other elements; then S is a (ϵ, δ) sketch matrix.*

Approximately, the sketch matrices S are 'invariant' over matrix multiplication ($\langle SA, SB \rangle \approx AB$). An important property of a (ϵ, δ)-sketch matrix is the subspace embedding property in Corollary 2: The norm of the vectors in the column space of A is kept roughly after being projected by S. Many distributions generate sketch matrices, such as *sparse embedding matrice* [57]. We show one way to generate a sparse embedding matrix in Corollary 3. This corollary shows that we need to choose $O(\epsilon^{-2})$ number of rows for a sparse embedding matrix to be a (ϵ, δ) matrix. Finally, since we directly solve a (sketched) linear equation without using any iterative optimization algorithms, we term this hypergradient estimator as a non-iterative approximation algorithm. The pseudocode summarizing this method is shown in Algorithm 3. We omit the subscript of iterates when it is clear from the context. Note that the server sends the random seed to ensure that all clients generate the same sketch matrices S_1 and S_2.

5 CONVERGENCE ANALYSIS

In this section, we analyze the convergence property of Algorithm 1. We first state some mild assumptions needed in our analysis, then we analyze the approximation error of the two hypergradient estimation algorithms, *i.e.* the iterative algorithm and the non-iterative

algorithm. Finally, we provide the convergence guarantee of Algorithm 1.

5.1 Some Mild Assumptions

We first state some assumptions about the outer and inner functions as follows:

Assumption A. *The function F and G has the following properties:*

a) *$F(x,y)$ is possibly non-convex, $\nabla_x F(x,y)$ and $\nabla_y F(x,y)$ are Lipschitz continuous with constant L_F*
b) *$\|\nabla_x F(x,y)\|$ and $\|\nabla_y F(x,y)\|$ are upper bounded by some constant C_F*
c) *$G(x,y)$ is continuously twice differentiable, and μ_G-strongly convex* w.r.t *y for any given x*
d) *$\nabla_y G(x,y)$ is Lipschitz continuous with constant L_G*
e) *$\|\nabla^2_{xy} G(x,y)\|$ is upper bounded by some constant $C_{G_{xy}}$*

Assumption B. *$\nabla^2_{xy} G(x,y)$ and $\nabla^2_{yy} G(x,y)$ are Lipschitz continuous with constants $L_{G_{xy}}$ and $L_{G_{yy}}$, respectively.*

In Assumptions A and B, we make assumptions about the function G, it is also possible to make stronger assumptions about the local functions g_i. Furthermore, these assumptions are used in the bilevel optimization literature [11, 19], especially, we require higher-order smoothness in Assumption B as bilevel optimization is involved with the second-order information. The next two assumptions are needed when we analyze the approximation property of the two hypergradient estimation algorithms:

Assumption C. *For a constant $0 < \tau < 1$ and a vector $g \in \mathbb{R}^d$. If $\exists$ i, such that $(g_i)^2 \geq \tau ||g||^2$, then g has τ-heavy hitters.*

Assumption D. *The stable rank of $\nabla^2_{yy} G(x, y_x)$ is bounded by r_s,* i.e. *$\sum_{i=1}^d \sigma_i^2 \leq r_s \sigma_{max}^2$, where ($\sigma_{max}$) σ_i denotes the (max) singular values of the Hessian matrix.*

The heavy-hitter assumption C is commonly used in the literature to show the convergence of gradient compression algorithms. To bound the approximation error of our iterative hypergradient estimation error, we assume $\nabla q(v)$ defined in Eq. (11) to satisfy this assumption. Assumption D requires the Hessian matrix to have several dominant singular values, which describes the sparsity of the Hessian matrix. We assume Assumption D holds when we analyze the approximation error of the non-iterative hypergradient estimation algorithm.

5.2 Approximation Error of the iterative algorithm

In this section, we show the approximation error of the iterative algorithm. Suppose that we choose count-sketch as the compressor, we have the following theorem:

Theorem 4. *Assume Assumptions A and C hold. In Algorithm 2, set the learning rate $\alpha = \frac{8}{\mu_G (i+a)}$ with $a > \max\left(1, \frac{2-\tau}{\tau}(\sqrt{\frac{2}{2-\tau}} + 1)\right)$ as a shift constant. If the compressed gradient has dimension $O(\frac{log(dI/\delta)}{\tau})$, then with probability $1-\delta$ we have:*

$$E[||v^I - v^*||^2] \leq \frac{C_1}{I^3} + \frac{C_2}{I^2} + \frac{C_3(I+2a)}{I^2}$$

where C_1, C_2, and C_3 are constants.

Remark 1. The proof is included in Appendix A. As shown by Theorem 4, the approximation error of Algorithm 2 is of the order of $O(1/I)$, and the constants encompass compression errors. Finally, the communication cost is of the order of $O(log(d))$, which is sublinear *w.r.t* of dimension d.

5.3 Approximation Error of the Non-iterative algorithm

In this section, we show the approximation error of the non-iterative algorithm. More precisely, we have Theorem 5:

Theorem 5. *For any given $\epsilon, \delta \in (0, 1/2)$, if $S_1 \in \mathbb{R}^{r_1 \times d}$ is a $(\lambda_1 \epsilon, \delta/2)$ sketch matrix and $S_2 \in \mathbb{R}^{r_2 \times d}$ is a $(\lambda_2 \epsilon, \delta/2)$ sketch matrix. Under Assumptions A and D, with probability at least $1-\delta$, we have the following:*

$$||\hat{\nabla} h(x) - \nabla h(x)|| \leq \epsilon ||v^*||$$

where $\lambda_1 = \frac{5\mu_G}{7\sqrt{r_s} C_{G_{xy}} L_G}$, $\lambda_2 = \frac{1}{3(r_1+1)}$ are constants.

Proof sketch. The main step is to bound $||S_1^T \hat{\omega} - v^*||$, then the conclusion follows from the definition of the hypergradient. To bound $||S_1^T \hat{\omega} - v^*||$, we use the approximation matrix multiplication property of S_1 and the subspace embedding property of S_2 to have: $||S_1^T \hat{\omega} - v^*||_2 \leq C||v^*||_F ||\nabla^2_{yy} G(x, y_x)||_F$, where C is some constant. The last step is to use the stable rank and the smoothness assumption to bound $||\nabla^2_{yy} G(x, y_x)||_F$. The full proof is included in Appendix B.

Remark 2. Based on Corollary 3, we have an (ϵ, δ) sketch matrix that has $r = O(\epsilon^{-2})$ rows. Combining with Theorem 5, we have $r_1 = O(r_s)$ and $r_2 = O(r_s^2)$. So, to reach the approximation error ϵ, the number of rows of the sketch matrices is $O(r_s^2)$. This shows that the stable rank (the number of dominant singular values) correlates with the number of dimensions to be maintained after compression.

5.4 Convergence of the Comm-FedBiO algorithm

In this section, we study the convergence property of the proposed communication efficient federated bilevel optimization (**Comm-FedBiO**) algorithm. First, $h(x)$ is smooth based on Assumptions A and B, as stated in the following proposition:

Proposition 6. *Under Assumption A and B, $\nabla h(x)$ is Lipschitz continuous with constant L_h,* i.e.

$$||\nabla h(x_1) - \nabla h(x_2)|| \leq L_h ||x_1 - x_2||$$

where $\nabla h(x)$ is the hypergradient and is defined in Proposition 1.

The proof of Proposition 6 can be found in the Lemma 2.2 of [11]. We are ready to prove the convergence of Algorithm 1 in the following theorem. In this simplified version, we ignore the exact constants. A full version of the theorem is included in Appendix C.

Theorem 7. *Under Assumption A and B, if we choose the learning rate $\eta = \frac{1}{2L_h \sqrt{K+1}}$ in Algorithm 1,*

a) *Suppose that* $\{x_k\}_{k\geq 0}$ *is generated from the iterative Algorithm 2. Under Assumption C, for* $I = O(\sqrt{K})$*, we have the following:*

$$E[||\nabla h(x_k)||^2] \leq \frac{C_1}{K^{3/2}} + \frac{C_2}{K} + \frac{C_3}{\sqrt{K}}$$

where C_1, C_2, C_3 *are some constants*

b) *Suppose* $\{x_k\}_{k\geq 0}$ *are generated from the non-iterative Algorithm 3. Under Assumption D, for* $\epsilon = O(K^{-1/4})$*, it holds:*

$$E[||\nabla h(x_k)||^2] \leq \frac{C}{\sqrt{K}}$$

where C *is some constant.*

Proof sketch. Firstly, by the smoothness of $h(x)$, we can upper-bound $h(x_{k+1})$ as:

$$\begin{aligned} h(x_{k+1}) \leq & h(x_k) - \eta(\frac{1}{2} - \eta L_h)||\nabla h(x_k)||^2 \\ & + \eta(\frac{1}{2} + \eta L_h)||\hat{\nabla} h(x_k) - \nabla h(x_k)||^2 \end{aligned}$$

Next, we need to bound the error in the third term, where we can utilize the bound provided in Theorem 4 and Theorem 5. Finally, we find a suitable averaging scheme to obtain the bound for $E[||\nabla h(x_k)||^2]$.

Remark 3. The convergence rate for the nonconvex-strongly-convex bilevel problem without using variance reduction technique is $O(1/\sqrt{K})$ [11], thus both estimation algorithms achieve the same convergence rate as in the non-distributed setting. For the non-iterative algorithm, we need to scale $\epsilon = O(K^{-1/4})$, while the iterative algorithm instead scales the number of iterations as $O(\sqrt{K})$ at each hyper-iteration. Comparing these two methods: The iterative algorithm has to perform multiple rounds of communication, but it distributes the computation burden over multiple communication clients (multiple clients by sampling different clients at each step) and requires one Hessian vector product per round. But if the communication is very expensive, we could instead use the non-iterative algorithm, which requires one round of communication.

6 EMPIRICAL EVALUATIONS

In this section, we empirically validate our **Comm-FedBiO** algorithm. We consider three real-world datasets: MNIST [24], CIFAR-10 [23] and FEMNIST [6]. For MNIST and CIFAR-10. We create 10 clients, for each client, we randomly sample 500 images from the original training set. For the server, we sample 500 images from the training set to construct a validation set. For FEMNIST, the entire dataset has 3,500 users and 805,263 images. We randomly select 350 users and distribute them over 10 clients. On the server side, we randomly select another 5 users to construct the validation set. Next, for label noise, we randomly perturb the labels of a portion of the samples in each client, and the portion is denoted ρ. We consider two settings: i.i.d. and non-i.i.d. setting. For the i.i.d. setting, all clients are perturbed with the same ratio ρ and we set $\rho = 0.4$ in experiments, while for the non-i.i.d. setting, each client is perturbed with a random ratio from the range of $[0.2, 0.9]$. The code is written with Pytorch, and the Federated Learning environment is simulated via Pytorch.Distributed Package. We used servers with

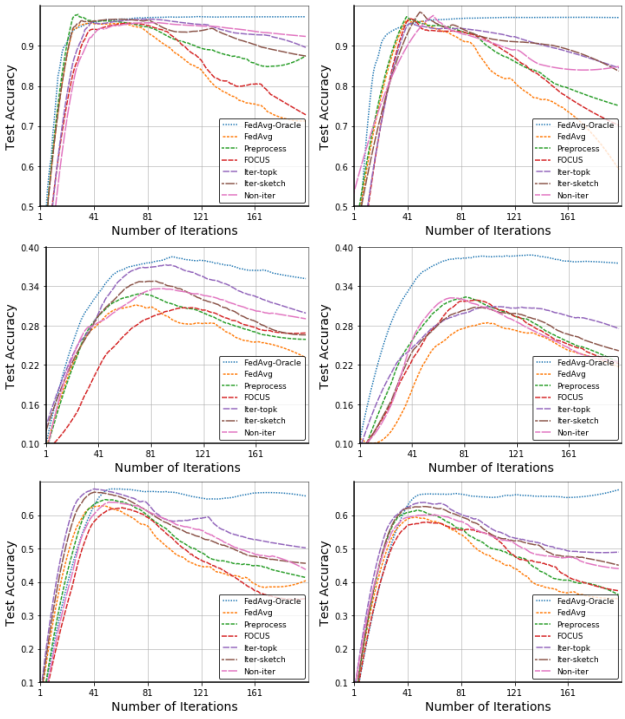

Figure 1: Test accuracy plots for our Comm-FedBiO (three variants: Iter-topK, Iter-sketch and Non-iter) and other baselines. The plots show the results for the MNIST dataset, the CIFAR-10 dataset, and the FEMNIST dataset from top to bottom. The plots in the left column show the i.i.d. case, and plots in the right column show the non-i.i.d. case. The compression rate of our algorithms is 20× in terms of the parameter dimension d**.**

AMD EPYC 7763 64-core CPU and 8 NVIDIA V100 GPUs to run our experiments.

For our algorithms, we consider three variants of *Comm-FedBiO* based on using different hypergradient approximation estimators: the non-iterative approximation method, the iterative approximation method with local Top-k and the iterative approximation method with Count-sketch compressor. We use Non-iter, Iter-topK and Iter-sketch as their short names. Furthermore, we also consider some baseline methods: a baseline that directly performs FedAvg [35] on the noisy dataset, an oracle method where we assume that clients know the index of clean samples (we denote this method as *FedAvg-Oracle*), the *FOCUS* [8] method which reweights clients based on a 'credibility score' and the *Preprocess* method [52] which uses a benchmark model to remove possibly mislabeled data before training.

We fit a model with 4 convolutional layers with 64 3×3 filters for each layer. The total number of parameters is about 10^5. We also use L_2 regularization with coefficient 10^{-3} to satisfy the strong convexity condition. Regarding hyper parameters, for three variants of our *Comm-FedBiO*, we set hyper-learning rates (learning rate for sample weights) as 0.1, the learning rate as 0.01, and the local iterations T as 5. We choose a minibatch of size 256 for MNIST and FEMNIST datasets and 32 for CIFAR10 datasets. For *FedAvg* and

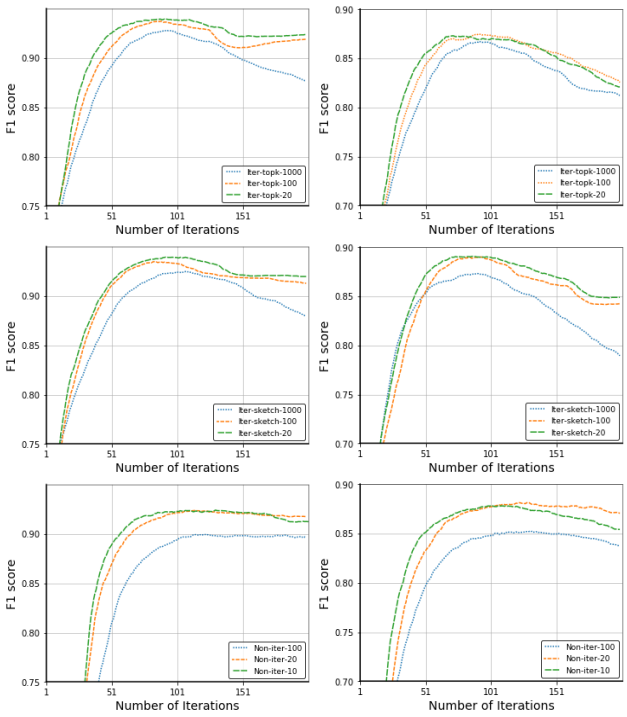

Figure 2: F1 score at different compression rates for the MNIST data set. The plots show the results for Iter-topK, Iter-sketch, and Non-iter from top to bottom. Plots in Left show the i.i.d case and in right show the non-i.i.d case.

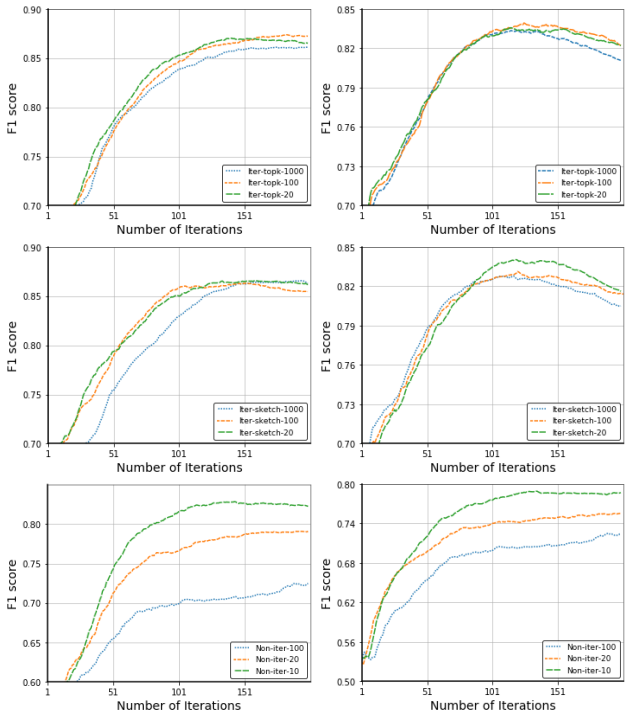

Figure 3: F1 score at different compression rates for the FEMNIST data set. The plots show results for Iter-topK, Iter-sketch and Non-iter from top to bottom. Plots on the left show the i.i.d. case, and in the right show the non-i.i.d case.

FedAvg-Oracle, we choose the learning rate, local iterations, and mini-batch size the same as in our *Comm-FedBiO*. For *FOCUS* [8], we tune its key parameter α to report the best results, for *Preprocess* [52], we tune its key parameter filtering threshold and report the best results.

We summarize the results in Figure 1. Due to the existence of noisy labels, *FedAvg* overfits the noisy training data quickly and the test accuracy decreases rapidly. On the contrary, our algorithm mitigates the effects of noisy labels and gets a much higher test accuracy than *FedAvg*, especially for the MNIST dataset, our algorithms get a test accuracy similar to that of the oracle model. Compared to *FedAvg*, our *Comm-FedBiO* performs the additional hypergradient evaluation operation at each global iteration (lines 11 - 13 in Algorithm 1). However, the additional communication overhead is negligible. In Figure 1, we need the communication cost $O(d/20)$, where d is the parameter dimension. Our algorithms are robust in labeling noise with almost no extra communication overhead. Finally, our algorithms also outperform the baselines *FOCUS* and *Preprocess*. The *FOCUS* method adjusts weights at the client level, so its performance is not good when all clients have a portion of mislabeled data, As for the *Preprocess* method, the benchmark model (trained over a small validation set) can screen out some mislabeled data, but its performance is very sensitive to performance of the benchmark model and a 'filtering threshold' hyperparameter.

Next, we verify that our algorithms are robust at different compression rates. The results are summarized in Figures 2 and 3. We use the $F1$ score to measure the efficacy of our algorithms in identifying mislabeled samples. We use 0.5 as the threshold for mislabeled data: for all samples with weights smaller than 0.5, we assume that they are mislabeled. Then the F1 score is computed between the ground-truth mislabeled samples and predicted mislabeled samples of our algorithms. In Figures 2 and 3, we show the results of i.i.d. and non-i.i.d. cases for the MNIST and FEMNIST datasets. The iterative algorithms (Iter-topK and Iter-sketch) achieve higher compression rates than the Non-iter method. For iterative algorithms, performance decreases at the compression rate 1000×, while the Non-iter method works well at around 10× to 100×. A major reason for this phenomenon is that the gradient $\nabla q(v)$ is highly sparse in experiments, whereas the Hessian matrix itself is much denser.

7 CONCLUSION

In this paper, we study the Federated Learning problem with noisy labels. We propose to use Shapley Value as a measure of the sample contribution. As Shapley Value is intractable, we then propose a Federated Bilevel Optimization formulation as its alternative. Next, we propose *Comm-FedBiO* to solve the Federated Bilevel Optimization problem, more specifically, we introduce two subroutines to estimate the hypergradient *i.e.* the Iterative and Non-iterative algorithms. We provide a theoretical convergence guarantee for both methods. In experiments, we validate our algorithms using real-world datasets. All empirical results show a superior performance of our proposed methods on various baselines.

REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

[2] R. Bao, B. Gu, and H. Huang. Efficient approximate solution path algorithm for order weight l_1-norm with accuracy guarantee. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 958–963. IEEE, 2019.

[3] R. Bao, B. Gu, and H. Huang. Fast oscar and owl regression via safe screening rules. In *International Conference on Machine Learning*, pages 653–663. PMLR, 2020.

[4] R. Bao, X. Wu, W. Xian, and H. Huang. Distributed dynamic safe screening algorithms for sparse regularization. *arXiv preprint arXiv:2204.10981*, 2022.

[5] A. K. R. Bayoumi, K. Mishchenko, and P. Richtarik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529, 2020.

[6] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[7] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 693–703. Springer, 2002.

[8] Y. Chen, X. Yang, X. Qin, H. Yu, B. Chen, and Z. Shen. Focus: Dealing with label quality disparity in federated learning. *arXiv preprint arXiv:2001.11359*, 2020.

[9] J. Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

[10] L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1165–1173. JMLR. org, 2017.

[11] S. Ghadimi and M. Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[12] R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pages 3748–3758. PMLR, 2020.

[13] F. Haddadpour and M. Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

[14] F. Huang and H. Huang. Biadam: Fast adaptive bilevel optimization methods. *arXiv preprint arXiv:2106.11396*, 2021.

[15] F. Huang and H. Huang. Enhanced bilevel optimization via bregman distance. *arXiv preprint arXiv:2107.12301*, 2021.

[16] F. Huang, J. Li, and H. Huang. Compositional federated learning: Applications in distributionally robust averaging and meta learning. *arXiv preprint arXiv:2106.11264*, 2021.

[17] N. Ivkin, D. Rothchild, E. Ullah, V. Braverman, I. Stoica, and R. Arora. Communication-efficient distributed sgd with sketching. *arXiv preprint arXiv:1903.04488*, 2019.

[18] K. Ji and Y. Liang. Lower bounds and accelerated algorithms for bilevel optimization. *arXiv preprint arXiv:2102.03926*, 2021.

[19] K. Ji, J. Yang, and Y. Liang. Provably faster algorithms for bilevel optimization and applications to meta-learning. *arXiv preprint arXiv:2010.07962*, 2020.

[20] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

[21] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261. PMLR, 2019.

[22] P. Khanduri, S. Zeng, M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. *arXiv preprint arXiv:2102.07367*, 2021.

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

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] J. Li, B. Gu, and H. Huang. Improved bilevel model: Fast and optimal algorithm with theoretical guarantee. *arXiv preprint arXiv:2009.00690*, 2020.

[26] J. Li, B. Gu, and H. Huang. A fully single loop algorithm for bilevel optimization without hessian inverse. *arXiv preprint arXiv:2112.04660*, 2021.

[27] J. Li, F. Huang, and H. Huang. Local stochastic bilevel optimization with momentum-based variance reduction. *arXiv preprint arXiv:2205.01608*, 2022.

[28] T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

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

[30] R. Liao, Y. Xiong, E. Fetaya, L. Zhang, K. Yoon, X. Pitkow, R. Urtasun, and R. Zemel. Reviving and improving recurrent back-propagation. *arXiv preprint arXiv:1803.06396*, 2018.

[31] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[32] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[33] J. Lorraine and D. Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018.

[34] D. Maclaurin, D. Duvenaud, and R. Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

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

[36] A. Mehra and J. Hamm. Penalty method for inversion-free deep bilevel optimization. *arXiv preprint arXiv:1911.03432*, 2019.

[37] A. Menon, B. Van Rooyen, C. S. Ong, and B. Williamson. Learning from corrupted binary labels via class-probability estimation. In *International conference on machine learning*, pages 125–134. PMLR, 2015.

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

[39] K. Nandakumar, N. Ratha, S. Pankanti, and S. Halevi. Towards deep neural network training on encrypted data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[40] K. Nishi, Y. Ding, A. Rich, and T. Hollerer. Augmentation strategies for learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8022–8031, 2021.

[41] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[42] F. Pedregosa. Hyperparameter optimization with approximate gradient. *arXiv preprint arXiv:1602.02355*, 2016.

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

[44] S. Sabach and S. Shtern. A first order method for solving convex bilevel optimization problems. *SIAM Journal on Optimization*, 27(2):640–660, 2017.

[45] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 3, 2018.

[46] L. S. Shapley. *Notes on the N-person Game–I: Characteristic-point Solutions of the Four-person Game.* Rand Corporation, 1951.

[47] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

[48] M. Solodov. An explicit descent method for bilevel convex optimization. *Journal of Convex Analysis*, 14(2):227, 2007.

[49] D. Sow, K. Ji, Z. Guan, and Y. Liang. A constrained optimization approach to bilevel optimization with multiple inner minima. *arXiv preprint arXiv:2203.01123*, 2022.

[50] S. U. Stich, J.-B. Cordonnier, and M. Jaggi. Sparsified sgd with memory. *arXiv preprint arXiv:1809.07599*, 2018.

[51] D. Tanaka, D. Ikami, T. Yamasaki, and K. Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5552–5560, 2018.

[52] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5020–5027. IEEE, 2021.

[53] S. Wagh, D. Gupta, and N. Chandran. Securenn: 3-party secure computation for neural network training. *Proc. Priv. Enhancing Technol.*, 2019(3):26–49, 2019.

[54] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[55] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *arXiv preprint arXiv:1705.07878*, 2017.

[56] R. A. Willoughby. Solutions of ill-posed problems (an tikhonov and vy arsenin). *SIAM Review*, 21(2):266, 1979.

[57] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *arXiv preprint arXiv:1411.4357*, 2014.

[58] J. Yang, K. Ji, and Y. Liang. Provably faster algorithms for bilevel optimization. *arXiv preprint arXiv:2106.04692*, 2021.

[59] S. Yang, H. Park, J. Byun, and C. Kim. Robust federated learning with noisy labels. *arXiv preprint arXiv:2012.01700*, 2020.

[60] L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR, 2019.

A PROOF FOR ITERATIVE ALGORITHM

In the iterative algorithm, we optimize Eq. (10). The full version of Theorem 4 in the main text is stated as follows:

THEOREM 8. *(Theorem 4) Under Assumption A, C and* $||v^i|| \leq D_v$, $\alpha = \frac{8}{\mu_G(i+a)}$, *with* $a > \max(1, \frac{2-\tau}{\tau}(\sqrt{\frac{2}{2-\tau}}+1))$ *being some shift constant. Then, if we use the count-sketch size* $O(log(dI/\delta)/\tau)$, *with probability* $1-\delta$, *we have the following:*

$$E[||v^I - v^*||^2] \leq \frac{C_1}{I^3} + \frac{C_2}{I^2} + \frac{C_3}{I^2}$$

where $G_q^2 = 2L_G^2 D_v^2 + 2C_F^2$, $C_1 = 3a^3 D_v^2/4$, $C_2 = (384(2L_G + \mu_G)G_q^2)/(\mu_G^3 \tau(1-(1-\frac{\tau}{2})(1+\frac{1}{a})^2))$, $C_3 = 12(I+2a)G_q^2/\mu_G^2$

PROOF. We first show that $E||\nabla q(v^i)||^2$ is upper bounded: $||\nabla q(v^i)||^2 = ||\nabla^2_{yy} G(x, y_x)v^i - \nabla_y F(x, y_x))||^2 \leq 2L_G^2 D_v^2 + 2C_F^2$. We denote $G_q^2 = 2L_G^2 D_v^2 + 2C_F^2$. Next, following the analysis in [21, 50], we consider the virtual sequence $\tilde{v}^i = v^i - e^i$, where we have:

$$\begin{aligned}\tilde{v}^i &= v^i - e^i = v^i - \alpha_{i-1}\nabla q(v^{i-1}) - e^{i-1} + C(\alpha_{i-1}\nabla q(v^{i-1}) + e^{i-1}) \\ &= v^{i-1} - e^{i-1} - \alpha_{i-1}\nabla q(v^{i-1}) = \tilde{v}^{i-1} - \alpha_{i-1}\nabla q(v^{i-1})\end{aligned}$$

Then we have:

$$\begin{aligned}&||\tilde{v}^i - v^*||^2 = ||\tilde{v}^{i-1} - \alpha_{i-1}\nabla q(v^{i-1}) - v^*||^2 \\ =&||\tilde{v}^{i-1} - v^*||^2 - 2\alpha_{i-1}\langle \tilde{v}^{i-1} - v^*, \nabla q(v^{i-1})\rangle + \alpha_{i-1}^2||\nabla q(v^{i-1})||^2 \\ \leq&||\tilde{v}^{i-1} - v^*||^2 - 2\alpha_{i-1}\langle \tilde{v}^{i-1} - v^{i-1}, \nabla q(v^{i-1})\rangle \\ &+ 2\alpha_{i-1}\langle v^* - v^{i-1}, \nabla q(v^{i-1})\rangle + \alpha_{i-1}^2 G_q^2\end{aligned} \tag{15}$$

In the last inequality, we use the fact that $\nabla q(v^{i-1})$ is upper-bounded. Then, since $q(v)$ is strongly convex, we have $q(v^*) \geq q(v^{i-1}) + \langle v^* - v^{i-1}, \nabla q(v^{i-1})\rangle + \frac{\mu_G}{2}||v^* - v^{i-1}||^2$. Furthermore, by the triangle inequality, we have: $||v^* - v^{i-1}||^2 \geq \frac{1}{2}||v^* - \tilde{v}^{i-1}||^2 - ||\tilde{v}^{i-1} - v^{i-1}||^2$. Combine these two inequalities, we can upper bound the third term in Eq. (15) and have:

$$\begin{aligned}||\tilde{v}^i - v^*||^2 \leq&(1 - \frac{\mu_G\alpha_{i-1}}{2})||\tilde{v}^{i-1} - v^*||^2 - 2\alpha_{i-1}\langle e^{i-1}, \nabla q(v^{i-1})\rangle \\ &+ \mu_G\alpha_{i-1}||e^{i-1}||^2 - 2\alpha_{i-1}(q(v^{i-1}) - q(v^*)) + \alpha_{i-1}^2 G_q^2\end{aligned} \tag{16}$$

Now, we bound $\langle e^{i-1}, \nabla q(v^{i-1})\rangle$, first by the triangle inequality, we have: $-\langle e^{i-1}, \nabla q(v^{i-1})\rangle \leq ||e^{i-1}||||\nabla q(v^{i-1})|| \leq (L_{G_y}||e^{i-1}||^2 + \frac{1}{4L_{G_y}}||\nabla q(v^{i-1})||^2)$. Then, by the smoothness of $q(v)$, we have

$$-\langle e^{i-1}, \nabla q(v^{i-1})\rangle \leq (L_{G_y}||e^{i-1}||^2 + \frac{1}{2}(q(v^{i-1}) - q(v^*)))$$

combine the two inequalities with Eq. (16), we have:

$$\begin{aligned}q(v^{i-1}) - q(v^*) \leq&\frac{(1-\mu_G\alpha_{i-1}/2)}{\alpha_{i-1}}||\tilde{v}^{i-1} - v^*||^2 - \frac{1}{\alpha_{i-1}}||\tilde{v}^i - v^*||^2 \\ &+ (2L_{G_y} + \mu_G)||e^{i-1}||^2 + \alpha_{i-1}G_q^2\end{aligned} \tag{17}$$

Note that we rearrange the terms and move $q(v^{i-1}) - q(v^*)$ to the left. Now, we bound the term $||e^i||^2$. By Assumption C, and the count sketch memory complexity in [7], suppose that we use the count sketch compressor and the compressed gradients have dimension $O(log(d/\delta)/\tau)$, we can recover the τ heavy hitters with probability at least $1-\delta$. Since we transfer compressed gradients I times, we have the communication cost of $O(log(dI/\delta)/\tau)$ by a union bound. Then for all $i \in [I]$, we have:

$$\begin{aligned}||e^i||^2 =&||\alpha_{i-1}\nabla q(v^{i-1}) + e^{i-1} - C(\alpha_{i-1}\nabla q(v^{i-1}) + e^{i-1})||^2 \\ \leq&(1-\tau)||\alpha_{i-1}\nabla q(v^{i-1}) + e^{i-1}||^2 \\ \leq&(1-\tau)(\alpha_{i-1}^2(1+\frac{1}{\gamma})||\nabla q(v^{i-1})||^2 + (1+\gamma)||e^{i-1}||^2) \\ \leq&(1-\tau)(1+\gamma)||e^{i-1}||^2 + (1-\tau)\alpha_{i-1}^2(1+\frac{1}{\gamma})G_q^2\end{aligned} \tag{18}$$

Next, we choose $\gamma = \frac{\tau}{2(1-\tau)}$, then we can prove

$$||e^i||^2 \leq \frac{(1-\tau)(2-\tau)(1+\frac{1}{a})^2\alpha_i^2 G_q^2}{\tau(1-(1-\frac{\tau}{2})(1+\frac{1}{a})^2)}$$

by induction, we omit the derivation here due to space limitation. By $a > \frac{2-\tau}{\tau}(\sqrt{\frac{2}{2-\tau}}+1)$, so the denominator is positive. Inserting the bound for $||e^i||^2$ back to Eq. (17), we have:

$$\begin{aligned}q(v^{i-1}) - q(v^*) \leq&\frac{(1-\frac{\mu_G\alpha_{i-1}}{2})}{\alpha_{i-1}}||\tilde{v}^{i-1} - v^*||^2 - \frac{1}{\alpha_{i-1}}||\tilde{v}^i - v^*||^2 \\ &+ \frac{2(2L_{G_y} + \mu_G)\alpha_{i-1}^2 G_q^2}{\tau(1-(1-\frac{\tau}{2})(1+\frac{1}{a})^2)} + \alpha_{i-1}G_q^2\end{aligned}$$

Finally, we average v^i with weight $w_i = (i+a)^2$, choose $\alpha = \frac{8}{\mu_G(i+a)}$, then by Lemma 3.3 in [50] and the strong convexity of $q(v)$, we get the upper bound of $||v^I - v^*||^2$ as shown in the Theorem. □

B PROOF FOR NON-ITERATIVE ALGORITHM

The proof for Corollary 2 is included in Theorem 9 in [57]. The full version of Theorem 5 in the main text is as follows:

THEOREM 9. *(Theorem 5) For any given* $\epsilon, \delta \in (0, 1/2)$, *if* $S_1 \in \mathbb{R}^{r_1 \times d}$ *is a* $(\lambda_1\epsilon, \delta/2)$ *sketch matrix and* $S_2 \in \mathbb{R}^{r_2 \times d}$ *is a* $(\lambda_2\epsilon, \delta/2)$ *sketch matrix. Under Assumption D, with probability at least* $1-\delta$ *we have:*

$$||\hat{\nabla}h(x) - \nabla h(x)|| \leq \epsilon||v^*||$$

where $\lambda_1 = \frac{5\mu_G}{7\sqrt{r_s}C_{G_{xy}}L_{G_y}}$, $\lambda_2 = \frac{1}{3(r_1+1)}$.

PROOF. For convenience, we denote $H_{yy} = \nabla^2_{yy}G(x, y_x)$, $H_{xy} = \nabla^2_{xy}G(x, y_x)$, $g_y = \nabla_y F(x, y_x)$, $g_x = \nabla_x F(x, y_x)$, $g = \nabla h(x)$, $\hat{g} = \hat{\nabla}h(x)$, and denote $\epsilon_1 = \lambda_1\epsilon$, $\epsilon_2 = (r_1+1)\lambda_2\epsilon$. Furthermore, we denote $v^* = \arg\min_v ||H_{yy}v - g_y||_2^2$, $\hat{\omega} = \arg\min_\omega ||S_2H_{yy}S_1^T\omega - S_2g_y||_2^2$, $v_{s_1} = \arg\min_v ||H_{yy}S_1^TS_1v - g_y||_2^2$ and $\omega_{s_1} = \arg\min_\omega ||H_{yy}S_1^T\omega - g_y||_2^2$. Then we have the following.

$$\begin{aligned}&(1-\epsilon_2)||H_{yy}S_1^T\hat{\omega} - g_y|| \leq ||S_2H_{yy}S_1^T\hat{\omega} - S_2g_y|| \\ &\leq ||S_2H_{yy}S_1^T\omega_{s_1} - S_2g_y|| \leq (1+\epsilon_2)||H_{yy}S_1^T\omega_{s_1} - g_y||\end{aligned} \tag{19}$$

The second inequality is by the definition of $\hat{\omega}$, the first and third inequality use the fact that S_2 is a $(\lambda_2\epsilon, \delta/2)$ sketch matrix and by Corollary 2, it is a $\lambda_2\epsilon \times (r_1+1) = \epsilon_2$ subspace embedding matrix over the column space $[H_{yy}S_1^T, g_y]$, so Eq. (19) holds with

probability $1-\delta/2$. Next, since S_1 is a $(\epsilon_1, \delta/2)$ sketching matrix, with probability $1-\delta/2$, we have:

$$\begin{aligned}&||H_{yy}S_1^TS_1v^* - H_{yy}v^*|| \le \epsilon_1||v^*||_F||H_{yy}||_F\\ &\rightarrow ||(H_{yy}S_1^TS_1v^* - g_y) - (H_{yy}v^* - g_y)|| \le \epsilon_1||v^*||_F||H_{yy}||_F\\ &\rightarrow ||H_{yy}S_1^TS_1v^* - g_y|| \le ||H_{yy}v^* - g_y|| + \epsilon_1||v^*||_F||H_{yy}||_F\end{aligned} \tag{20}$$

In the last step, we use the triangle inequality $||x||-||y|| \le ||x-y||$. Combining Eq. (20) and the definition of v_{s_1}, also noticing that $span(S_1v) \subset span(\omega)$, we have:

$$\begin{aligned}&||H_{yy}S_1^T\omega_{s_1} - g_y|| \le ||H_{yy}S_1^TS_1v_{s_1} - g_y||\\ &\le ||H_{yy}S_1^TS_1v^* - g_y|| \le ||H_{yy}v^* - g_y|| + \epsilon_1||v^*||_F||H_{yy}||_F\end{aligned} \tag{21}$$

Finally we combine Eq. (19) and (21) to get:

$$||H_{yy}S_1^T\hat{\omega} - g_y|| \le \frac{(1+\epsilon_2)}{(1-\epsilon_2)}(||H_{yy}v^* - g_y|| + \epsilon_1||v^*||_F||H_{yy}||_F) \tag{22}$$

By the union bound, Eq. (22) holds with probability $1-\delta$. Since H_{yy} is positive definite (invertible), we have $||H_{yy}v^* - g_y|| = 0$. Eq. (22) can be simplified further as:

$$||H_{yy}S_1^T\hat{\omega} - g_y|| \le \frac{\epsilon_1(1+\epsilon_2)}{(1-\epsilon_2)}||v^*||_F||H_{yy}||_F$$

Moreover, $G(x,y)$ is μ_G-strongly convex (Assumption A,), we have:

$$\begin{aligned}&||H_{yy}S_1^T\hat{\omega} - g_y||^2 = ||H_{yy}S_1^T\hat{\omega} - g_y - (H_{yy}v^* - g_y)||^2\\ &=||H_{yy}(S_1^T\hat{\omega} - v^*)||^2 \ge \mu_G^2||S_1^T\hat{\omega} - v^*||^2\end{aligned} \tag{23}$$

Combining the above two equations, we have the following.

$$||S_1^T\hat{\omega} - v^*|| \le \frac{\epsilon_1(1+\epsilon_2)}{\mu_G(1-\epsilon_2)}||v^*||_F||H_{yy}||_F \tag{24}$$

As for $||H_{yy}||_F$, by Assumption C, we have $||H_{yy}||_F = \sqrt{\sum_i \sigma_i^2} \le \sqrt{r_s}\sigma_{max} = \sqrt{r_s}L_{G_y}$. Then Eq. (24) can be simplified to $||S_1^T\hat{\omega} - v^*|| \le \frac{\sqrt{r_s}L_{G_y}\epsilon_1(1+\epsilon_2)}{\mu_G(1-\epsilon_2)}||v^*||$. Then we have the following:

$$\begin{aligned}||\hat{g} - g|| &= ||H_{xy}S_1^T\hat{\omega} - H_{xy}v^*|| = ||H_{xy}(S_1^T\hat{\omega} - v^*)||\\ &\le \frac{\sqrt{r_s}C_{G_{xy}}L_{G_y}\epsilon_1(1+\epsilon_2)}{\mu_G(1-\epsilon_2)}||v^*||\end{aligned} \tag{25}$$

By choice of $\epsilon_1 = \frac{5\mu_G\epsilon}{7\sqrt{r_s}C_{G_{yx}}L_{G_y}}$ and $\epsilon_2 = \frac{\epsilon}{3} < \frac{1}{6}$, *i.e.* $\frac{1+\epsilon_2}{1-\epsilon_2} = 1 + \frac{2\epsilon_2}{1-\epsilon_2} < \frac{7}{5}$. So, we get $||\hat{g} - g|| < \epsilon||v^*||$ with probability at least $1-\delta$. The proof is complete. □

C PROOF FOR CONVERGENCE ANALYSIS

The full version of Theorem 7 in the main text is provided as follows:

Theorem 10. *(Theorem 7) Under Assumption A, B, we pick the learning rate* $\eta = \frac{1}{2L_h\sqrt{K+1}}$*, then we have:*

a) *Suppose that* $\{x_k\}_{k\ge 0}$ *is generated from the non-iterative Algorithm 3, under Assumption D and* $\epsilon = (K+1)^{-1/4}$*, we have the following:*

$$E[||\nabla h(x_k)||^2] \le \left(32L_h\left(h(x_0) - h(x^*)\right) + \frac{16C_F^2}{\mu_G^2}\right)\frac{1}{\sqrt{K}}$$

b) *Suppose* $\{x_k\}_{k\ge 0}$ *are generated from the non-iterative Algorithm 4, under Assumption C,* $I = L_h\sqrt{K+1}$*, we have:*

$$E[||\nabla h(x_k)||^2] \le \frac{C_1}{K^{3/2}} + \frac{C_2}{K} + \frac{C_3}{\sqrt{K}}$$

where C_1, C_2 *and* C_3 *are some constants.* $C_1 = 12C_{G_{xy}}^2a^3D_v^2/L_h^3$, $C_2 = (6144(2L_{G_y} + \mu_G)C_{G_{xy}}^2G_q^2)/(\mu_G^3\tau(1-(1-\frac{\tau}{2})(1+\frac{1}{a})^2)L_h^2) + 384aC_{G_{xy}}^2G_q^2/\mu_G^2L_h^2$, $C_3 = 32L_h(h(x_0) - h(x^*)) + (192C_{G_{xy}}^2G_q^2)/\mu_G^2L_h$.

Proof. As stated in Proposition 6, $h(x)$ is L_h smooth: $h(x_{k+1}) \le h(x_k) + \langle\nabla h(x_k), x_{k+1} - x_k\rangle + \frac{L_h}{2}||x_{k+1} - x_k||^2$, and by the update rule of outer variable $x_{k+1} = x_k - \eta\hat{\nabla}h(x_k)$, we have:

$$\begin{aligned}h(x_{k+1}) \le& h(x_k) - \eta\langle\nabla h(x_k), \hat{\nabla}h(x_k)\rangle + \frac{L_h}{2}\eta^2||\hat{\nabla}h(x_k)||^2\\ \le& h(x_k) - \eta||\nabla h(x_k)||^2 + \eta\langle\nabla h(x_k), \nabla h(x_k) - \hat{\nabla}h(x_k)\rangle\\ &+\frac{L_h}{2}\eta^2||\hat{\nabla}h(x_k) - \nabla h(x_k) + \nabla h(x_k)||^2\end{aligned}$$

Using the triangle inequality and the Cauchy-Schwarz inequality, we have $\langle\nabla h(x_k), \nabla h(x_k) - \hat{\nabla}h(x_k)\rangle \le \frac{1}{2}||\nabla h(x_k)||^2 + \frac{1}{2}||[\hat{\nabla}h(x_k) - \nabla h(x_k)||^2$ and $||\hat{\nabla}h(x_k) - \nabla h(x_k) + \nabla h(x_k)||^2 \le 2||\hat{\nabla}h(x_k) - \nabla h(x_k)||^2 + 2||\nabla h(x_k)||^2$. We combine these two inequalities with the above inequality.

$$h(x_{k+1}) \le h(x_k) - \eta(\frac{1}{2} - \eta L_h)||\nabla h(x_k)||^2 + \eta(\frac{1}{2} + \eta L_h)e_k \tag{26}$$

where we denote $||\hat{\nabla}h(x_k) - \nabla h(x_k)||^2$ as e_k. Next we prove the two cases respectively.

Case (a): By Theorem 5 and $||v^*|| = ||\nabla_{yy}^2G(x,y_x)^{-1}\nabla_yF(x,y_x)|| \le \frac{C_F}{\mu_G}$, we have: $e_k \le \epsilon^2||v_k^*||^2 \le \epsilon^2C_{F_y}^2/\mu_G^2$. Combine this inequality with Eq. (26) and telescope from 1 to K, we have::

$$\sum_{k=0}^{K-1}\eta(\frac{1}{2} - \eta L_h)||\nabla h(x_k)||^2 \le h(x_0) - h(x^*) + \sum_{k=0}^{K-1}\eta(\frac{1}{2} + \eta L_h)\frac{\epsilon^2C_F^2}{\mu_G^2}$$

We select x_k with probability proportional to $\eta(\frac{1}{2} - \eta L_h)$, and pick $\eta = \frac{1}{2L_h\sqrt{K+1}}$, $\epsilon = (K+1)^{-1/4}$ we have:

$$E[||\nabla h(x_k)||^2] \le \frac{1}{\sqrt{K}}\left(32L_h(h(x_0) - h(x^*)) + \frac{16C_F^2}{\mu_G^2}\right)$$

where we use $\sum_{k=0}^{K-1}\eta(\frac{1}{2} - \eta L_h) \ge \frac{\sqrt{K}}{32L_h}$, $\sum_{k=0}^{K-1}\epsilon^2\eta(\frac{1}{2} + \eta L_h) \le \frac{1}{2L_h}$.

Case (b): for the iterative algorithm, we notice that $e_k = ||\nabla_xF(x,y_x) - \nabla_{xy}^2G(x,y_x)v^I - \nabla_xF(x,y_x) - \nabla_{xy}^2G(x,y_x)v^*||^2 \le C_{G_{xy}}^2||(v^I - v^*)||^2$. Combine this inequality with Eq. (26) and telescope from 1 to K as case (a), we have:

$$\begin{aligned}&\sum_{k=0}^{K-1}\eta(\frac{1}{2} - \eta L_h)||\nabla h(x_k)||^2\\ &\le h(x_0) - h(x^*) + \sum_{k=0}^{K-1}\eta(\frac{1}{2} + \eta L_h)C_{G_{xy}}^2||(v^I - v^*)||^2\end{aligned}$$

By choosing the learning rate *eta* and I as in the Theorem, then combine Theorem 4, it is straightforward to get the upper bound of $||\nabla h(x_k)||^2$ as stated in the Theorem. □