Resolving the Tug-of-War: A Separation of Communication and Learning in Federated Learning

Junyi Li
Computer Science
University of Maryland College Park
junyili.ai@gmail.com

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

Abstract

Federated learning (FL) is a promising privacy-preserving machine learning paradigm over distributed data. In this paradigm, each client trains the parameter of a model locally and the server aggregates the parameter from clients periodically. Therefore, we perform the learning and communication over the same set of parameters. However, we find that learning and communication have fundamentally divergent requirements for parameter selection, akin to two opposite teams in a tug-of-war game. To mitigate this discrepancy, we introduce FedSep, a novel two-layer federated learning framework. FedSep consists of separated communication and learning layers for each client and the two layers are connected through decode/encode operations. In particular, the decoding operation is formulated as a minimization problem. We view FedSep as a federated bilevel optimization problem and propose an efficient algorithm to solve it. Theoretically, we demonstrate that its convergence matches that of the standard FL algorithms. The separation of communication and learning in FedSep offers innovative solutions to various challenging problems in FL, such as Communication-Efficient FL and Heterogeneous-Model FL. Empirical validation shows the superior performance of FedSep over various baselines in these tasks.

1 Introduction

In Federated Learning (FL) [42], a set of clients jointly solve a machine learning task under the coordination of a central server. The process of FL involves two category of operations: Learning and Communication. For the learning operation, each client optimizes their local objectives, and for the communication operation, clients exchange local parameters to facilitate knowledge sharing. In the existing FL pipeline, both Learning and Communication operations hinge on the same set of parameters. However, these two operations have fundamentally divergent requirements, akin to two teams engaged in a tug-of-war, each pulling in opposite directions. In fact, on the communication side, it is imperative that the parameters of all clients reside in a uniform space. Moreover, to mitigate the high communication costs, a major bottleneck in FL, it is beneficial to maintain these parameters within a low-dimensional space. In contrast, on the learning side, given the heterogeneity of devices, including variations in hardware and data distribution, it is advantageous to allow the parameter space to differ across clients. Furthermore, to accommodate the implementation of state-of-the-art large-scale machine learning models (such as Transformers [56]), a high-dimensional parameter space is desirable. In summary, a huge discrepancy of requirements to parameter selection exists between the communication and the learning in FL.

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

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

To mitigate this discrepancy, we opt to 'break the rope' in this tug-of-war scenario. More precisely, we propose a two-layer structure on clients: a communication layer and a learning layer. As shown in Figure 1, we denote this framework as **FedSep**: The communication layer connects with the server and uploads/downloads parameter, while the learning layer performs local objective optimization. Then the communication layer and the learning layer are connected through encode/decode operations. The decode operation maps the parameter of the communication layer to the parameter of the learning layer and the encode operation performs the mapping in the opposite direction. More specifically, we formulate the decode operation as solving a minimization problem and does not assume an explicit relation between the parameter of communication layer and the parameter of the learning layer. In summary, we aim to use two distinct sets of parameters in FedSep to resolve the 'competition' between the communication and learning operations, meanwhile, the two set of parameters are close connected through decode/encode operations. Mathematically, FedSep can be formulated as solving a federated bilevel problem. The upper level problem corresponds to the learning problems on clients, while the lower level problem is the minimization problem in the decode operation. We propose an efficient algorithm to solve this bilevel problem, which is composed of three consecutive steps: Decode the communication parameter, Learning on local problems and Encode the learning parameter to the communication parameter. The convergence of the algorithm is guaranteed.

To demonstrate the flexibility of FedSep in incorporating the contradictory requirements of the communication and learning, we study two real-world FL tasks. In the first task, we study the communication-efficient FL task. In this task, we set the dimension of the communication parameter to be much smaller than the learning parameter, furthermore, we decode the communication parameter through a LASSO problem. In the second task, we study the model-heterogeneous FL task. In this task, the learning parameters on different clients have different dimension. We set the communication parameter to be the parameter of the server model, and the learning parameter be a subset of the communication parameter (server model). In particular, the decode operation is to select the subset adaptively based on the client's local data. We empirically verify the superior performance of our methods compared to other baselines. Finally, we summarize the main **contributions** of our work as follows:

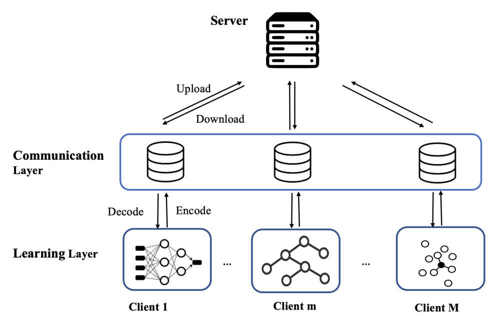

Figure 1: The structure of FedSep. FedSep follows the one-server-multiple-clients structure, in particular, each client has two layers: the communication layer and the learning layer. The two layers are connected through the decode and encode operations.

1. We propose a novel two-layer Federated Learning framework, *i.e.* **FedSep**, where the communication layer and the learning layer are separated on clients;
2. We let the decode operation from the communication layer to the learning layer be solving a minimization problem, and therefore, the **FedSep** framework has a bilevel optimization formulation. We propose an efficient algorithm to solve the bilevel problem and show that its convergence rate matches that of the standard Federated Learning algorithms;
3. We apply the **FedSep** framework to solve two important FL tasks: Communication-Efficient FL and Model-Heterogeneous FL. Numerical experiments show the superior performance of the methods based on FedSep.

Notations. ∇ denotes the full gradient, ∇_x is the partial derivative for variable x, and higher-order derivatives follow similar rules. $||\cdot||_2$ is ℓ_2-norm for vectors and the spectral norm for matrices. AB denotes the multiplication of the matrix between the matrix A and B. $[K]$ represents the sequence of integers from 1 to K.

2 Related Works

Federated Learning. FL is a novel paradigm for performing machine learning tasks over distributed data, but it also poses challenges such as heterogeneity [27], privacy [44] and communication bottleneck [58]. In particular, communication cost is one of the major bottlenecks in FL. A widely used approach to reduce communication burden is through compression. Various compressors [58, 39, 54, 28] have been studied in the literature. Next, model-heterogeneous FL is also studied to solve the device heterogeneity issue. There are two main categories of heterogeneous model methods: knowledge-distillation (KD) [38, 24, 10] based and partial-training (PT) [12, 6, 1] based. KD-based methods treat the server model as the teacher and the clients' model as the student. In contrast, PT-based methods select a subset of the server model for each client, and in the aggregation phase, the server simply aggregates the updated sub-models.

Bilevel Optimization. Bilevel problem is a type of two-layer optimization problem [59]. In the machine learning community, Hyper-parameter Optimization problems [31, 8, 4, 13] are one of the mostly studied bilevel problems. Early bilevel optimization algorithms solved the lower problem exactly to update the upper variable. Recently, researchers developed algorithms that solve the lower problem approximately with a fixed number of steps and use the back-propagation technique to compute the approximate hyper-gradient [14, 41, 16, 46, 51]. More recently, single-loop algorithms [18, 20, 26, 33, 29, 9, 62, 34, 11, 22, 23] based on alternative updates of lower and upper level problems are proposed. However, less work studies bilevel problems under the federated setting. [55, 36, 35] studies a type of federated bilevel problems where the lower level problems are federated. In particular, [36] solves the noisy label problem in FL. Note that our FedSep is formulated as a type of federated bilevel problems, however, our problem is different from that in [55, 36]. Firstly, our problem has a unique lower problem for each client, moreover, FedSep impose unique constraints not covered by the existing methods. Please refer to Section B in Appendix for more related works.

3 Separating Communication and Learning in Federated Learning

In this section, we propose **FedSep**, a novel two-layer Federated Learning Framework. As shown in Figure 1, FedSep has the one-server-multiple-clients structure as in the classical Federated learning framework, we assume M clients in the training. The difference between the FedSep and classical FL framework lies in the client. Different from the classical FL, FedSep includes two separated layers on clients: a communication layer and a learning layer. The two layers have different responsibilities. As the name shows, the learning layer is responsible of finishing the learning task, *e.g.* fitting a neural network over a dataset. We assume that the learning layer on client $m \in [M]$ has parameter $y^{(m)} \in \mathbb{R}^{d^{(m)}}$. Note that the choice of $y^{(m)}$ should be adapted to the specific setting of each client, such as the data distribution, hardware resources *etc.*. Next, the communication layer performs communication with the server (other clients), which includes the **upload** and **download** operations. We assume the communication layer on client m has parameter $x \in \mathbb{R}^p$. Note that the parameter of communication layer has identical formulation for all clients. Finally, the communication layer and the learning layer are connected through the **encode** and **decode** operations.

To model the various potential relationship between the communication and learning layer, we abstract the decode operation as solving a minimization problem, and the encode operation be its reverse operation. As a result, we have the following bilevel formulation for the FedSep framework.

$$\min_{x\in\mathbb{R}^p} h(x) := \frac{1}{M}\sum_{m=1}^{M} h^{(m)}(x) := \frac{1}{M}\sum_{m=1}^{M} f^{(m)}(y_x^{(m)}), \;\; y_x^{(m)} = \underset{y^{(m)}\in\mathbb{R}^{d^{(m)}}}{\arg\min}\, g^{(m)}(x, y^{(m)}) \tag{1}$$

In Eq. (1), M is the number of clients; $f^{(m)}(y), m \in [M]$ denotes the local problem solved by the client m; $g^{(m)}(x, y)$ is the minimization problem solved in the decode operation by the client m; x is the parameter of the communication layer; $y^{(m)}$ denotes the parameter of the learning layer on the client m.

Next, we propose a three-stage algorithm to solve Eq. (1). As shown by Algorithm 1, we perform T global steps, and at each step $t \in T$, we randomly choose a subset of clients $\mathcal{M}_t$ to perform the training and then the server performs aggregation. The **Client training** is divided into three stages: **Decode stage** (lines 7-12), **Learning stage** (lines 14-17), and **Encode Stage** (line 19).

Decode Stage. In Eq. (1), the decode operation solves the lower optimization problem $g^{(m)}(x, y)$ to obtain $y_x^{(m)}$. We denote the decode operator on the m_{th} client by $Dec^{(m)}\{\cdot\}$, thus we have: $y_x^{(m)} = Dec^{(m)}\{x\}$. In Algorithm 1, we solve $g^{(m)}(x, y)$ approximately with I_{dec} steps of stochastic gradient

Algorithm 1 Separating Communication and Learning in FL (**FedSep**)

1: **Input:** Initial states x_1; learning rates γ, η; mini-batchsize b_x, b_y
2: **for** $t = 1$ **to** T **do**
3: Randomly sample a subset $\mathcal{M}_t$ of clients;
4: **for** $m \in \mathcal{M}_t$ in parallel **do**
5: // Decode stage, estimate $y_x^{(m)} = Dec^{(m)}\{x\}$;
6: Receive the global state x_t from the server and set $x^{(m)} = x_t$, and initialize $y_0^{(m)}$;
7: **for** $i = 1$ to I_{dec} **do**
8: Randomly sample a minibatch of b_y samples $\mathcal{B}_y$;
9: $y_i^{(m)} = y_{i-1}^{(m)} - \gamma \nabla_y g^{(m)}(x^{(m)}, y_{i-1}^{(m)}, \mathcal{B}_y)$
10: **end for**
11: Set $\hat{y}_0^{(m)} = y_{I_{dec}}^{(m)}$ as the estimation of $Dec^{(m)}(x^{(m)})$;
12: // Learning stage, optimize $f^{(m)}(y)$;
13: **for** $i = 1$ to I **do**
14: Randomly sample a minibatch of b_y samples $\mathcal{B}_{\hat{y}}$;
15: $\hat{y}_i^{(m)} = \hat{y}_{i-1}^{(m)} - \eta \nabla f^{(m)}(\hat{y}_{i-1}^{(m)}; \mathcal{B}_{\hat{y}})$;
16: **end for**
17: // Encode stage, encode the update of the learning layer back to the communication layer;
18: Set $\Delta \hat{y}^{(m)} = \hat{y}_I^{(m)} - \hat{y}_0^{(m)}$ and compute $\Delta \hat{x}_t^{(m)} = \widetilde{Enc}^{(m)}\{\Delta \hat{y}^{(m)}\}$, where $\widetilde{Enc}^{(m)}\{\cdot\}$ is defined in Eq. (3) and we choose $|\mathcal{B}_x| = b_x$.
19: **end for**
20: $x_{t+1} = x_t - \frac{1}{|\mathcal{M}_t|} \sum_{m \in \mathcal{M}_t} \eta_g \Delta \hat{x}_t^{(m)}$
21: **end for**

descent (Line 8-11) and use the output as an approximation of $y_x^{(m)}$, *i.e.*. $y_{I_{dec}}^{(m)} = \widetilde{Dec}^{(m)}(x^{(m)}) \approx y_x^{(m)}$, where $\widetilde{Dec}^{(m)}(\cdot)$ represents the approximation of the exact decoder $Dec^{(m)}\{\cdot\}$.

Learning Stage. Next, we perform I steps of stochastic gradient descent to solve the local learning problem $f^{(m)}(y)$ (Line 14 - 17). This stage is similar to the local gradients in classical FL framework. Note we get $\hat{y}_I^{(m)}$ as the updated learning parameter $y^{(m)}$, more formally, we use $\Delta \hat{y}^{(m)} = \hat{y}_I^{(m)} - \hat{y}_0^{(m)} = \hat{y}_I^{(m)} - y_{I_{dec}}^{(m)}$ to represent the update in the learning stage.

Encode Stage. After the learning stage, we get the update of the local learning problem $\Delta \hat{y}^{(m)}$. Suppose we denote the encode operator $Enc^{(m)}\{\cdot\}$, then we get the update of the communication parameter as $\Delta \hat{x}^{(m)} = Enc^{(m)}(\Delta \hat{y}^{(m)})$ (Line 19). In particular, we choose the following encode operator:

$$Enc^{(m)}\{\cdot\} := -\nabla_x y_x^{(m)} = \nabla_{xy} g^{(m)}(x, y_x^{(m)}) \big(\nabla_{yy} g^{(m)}(x, y_x^{(m)})\big)^{-1} \tag{2}$$

where $y_x^{(m)} = Dec^{(m)}(x)$ is the output of the exact decode operation and the second equality can be derived following the implicit function theorem under mild assumptions [18]. To reduce the computation complexity, we use Neumann series to approximate the inverse operation in Eq. (2). More specifically, we have the following approximation: $\big(\nabla_{yy} g^{(m)}(x, y)\big)^{-1} \approx \tau \sum_{q=0}^{Q} (I - \tau \nabla_{yy} g^{(m)}(x, y))^q$, where Q and τ are some constants. Since in the decode stage, we use $y_x^{(m)} \approx y_{I_{dec}}^{(m)}$, we get a stochastic approximation of Eq. (2) as:

$$\widetilde{Enc}^{(m)}\{\cdot\} := \sum_{q=0}^{Q} \tau \nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x)(I - \tau \nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x))^q \tag{3}$$

Remark 1. We use Eq. (2) as the encode operator due to the following fact about the hyper-gradient:

$$\nabla h^{(m)}(x) = \nabla_x y_x^{(m)}(\nabla f^{(m)}(y_x^{(m)}))$$

Note that suppose $\Delta \hat{y}^{(m)} = -\nabla f^{(m)}(y_x^{(m)})$, which means the learning layer performs one step of gradient descent with learning rate 1 in the learning stage of Algorithm 1, then we have: $\nabla h^{(m)}(x) =$

$Enc\{\Delta\hat{y}^{(m)}\} = \Delta x^{(m)}$. Therefore, we update the communication parameter x such that it optimize the overall problem $\nabla h^{(m)}(x)$.

Server Aggregation. Finally, the server aggregates the local updates of communication parameter $\Delta\hat{x}^{(m)}, m \in \mathcal{M}_t$ to get the new communication parameter as shown in Line 21 of Algorithm 1.

Difference with existing bilevel algorithms. Bilevel optimization problems are widely studied in the literature [18, 26]. However, FedSep cannot directly apply existing algorithms. In a standard bilevel algorithm, each step of update to the upper level variable requires (approximately) solving of the corresponding lower level problem. However, in each epoch of FedSep, we only solve the lower level problem once and perform multiple steps of update to the upper level variable.

3.1 Convergence Analysis

In this section, we provide the convergence analysis of Algorithm 1. Before we state the convergence result, we first make the following assumptions.

Assumption 3.1. Function $f^{(m)}(y)$ is possibly non-convex and $g^{(m)}(x, y)$ is μ-strongly convex *w.r.t* y for any given x.

Assumption 3.2. Function $f^{(m)}(y)$ is L-smooth and has C_f-bounded gradient.

Assumption 3.3. Function $g^{(m)}(x, y)$ is L-smooth; $\nabla_{xy} g^{(m)}(x, y)$ and $\nabla_{y^2} g^{(m)}(x, y)$ are Lipschitz continuous with constants L_{xy} and L_{y^2}, respectively.

Assumption 3.4. We have unbiased stochastic first-order and second-order gradient oracle with bounded variance.

Note that Assumption 3.1-Assumption 3.3 are standard assumptions used to analyze bilevel problems [26, 18]. Assumption 3.4 is standard in analyzing stochastic optimization problems. For a full version of Assumption 3.4, please refer to Assumption C.1 in the Appendix.

In the learning stage of Algorithm 1, we perform I steps of updates to optimize the learning problem, and the update has the form of: $\Delta\hat{y}^{(m)} = \hat{y}_I^{(m)} - \hat{y}_0^{(m)} = \sum_{i=0}^{I-1} -\eta\nabla f^{(m)}(\hat{y}_i^{(m)}; \mathcal{B}_{\hat{y}})$, therefore we have: $\Delta\hat{x}_t^{(m)} = \widetilde{Enc}^{(m)}\{\Delta\hat{y}^{(m)}\} = \sum_{i=0}^{I-1} \eta\widetilde{Enc}^{(m)}\{-\nabla f^{(m)}(\hat{y}_i^{(m)}; \mathcal{B}_{\hat{y}})\}$. We denote $\Delta\hat{x}_{t,i}^{(m)} = \widetilde{Enc}^{(m)}\{\nabla f^{(m)}(\hat{y}_i^{(m)}; \mathcal{B}_{\hat{y}})\}$. **Then the main effort of the proof is bounding the estimation error of $\boldsymbol{\Delta\hat{x}_{t,i}^{(m)}}$ to the hyper-gradient $\boldsymbol{\nabla h^{(m)}(x)}$.**

More specifically, we first show that $\Delta\hat{x}_{t,i}^{(m)}$ estimates a term $\mu_i^{(m)}$ with bounded variance and bias as stated in the following proposition:

Proposition 3.5. *Suppose Assumptions 3.1-3.3 and 3.4 hold and $\tau < \frac{1}{L}$, we have $\|\mathbb{E}_\xi[\Delta\hat{x}_{t,i}^{(m)}] - \mu_i^{(m)}\| \leq G_1$, where $\mu_i^{(m)} = -\nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big(\nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)})$, and $\mathbb{E}\|\Delta\hat{x}_{t,i}^{(m)} - \mathbb{E}_\xi[\Delta\hat{x}_{t,i}^{(m)}]\|^2 \leq G_2^2$. G_1 and G_2 are some constants related to Q in Eq.* (3) *and mini-batch size.*

Please refer to Proposition C.3 for the specific form of the constants in Proposition 3.5. Next, we have that $\mu_i^{(m)} \approx \nabla h^{(m)}(x)$ as shown in the following proposition:

Proposition 3.6. *Suppose Assumptions 3.2 and 3.3 hold, the following statements hold:*

a) $\|\mu_i^{(m)} - \nabla h^{(m)}(x)\|^2 \leq 2\hat{L}^2\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + 2\kappa^2 L^2\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2$, where $\hat{L} = O(\kappa^2)$.

b) $h^{(m)}(x)$ is Lipschitz continuous in x with constant $\bar{L}$ i.e., for any given $x_1, x_2 \in \mathbb{R}^p$, we have $\|\nabla h^{(m)}(x_2) - \nabla h^{(m)}(x_1)\|^2 \leq \bar{L}^2\|x_2 - x_1\|$, where $\bar{L} = O(\kappa^3)$.

where we denote the condition number as $\kappa = L/\mu$.

Please refer to Proposition C.2 for the proof. Proposition 3.6 shows that $\mu_i^{(m)}$ estimates the true hyper-gradient $\nabla h^{(m)}(x)$ with two types of errors: the estimation error of the decode operation ($\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2$) and the drift of the learning process ($\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2$), where Lemma C.4 and

Lemma C.5 show bounds for these two errors. Furthermore, Proposition 3.6.b) shows that $h^{(m)}(x)$ is smooth. Next, we are ready to show the convergence of Algorithm 1:

Theorem 3.7. *Suppose Assumptions 3.1-3.4 hold, and we run T iterations of Algorithm 1, with the learning rates satisfy $\gamma < \frac{1}{L}$, $\eta\eta_g < \frac{1}{2I\bar{L}}$, then we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_{\xi}[\bar{\Delta}\hat{x}_{t,i}]\|^2\right)$$
$$\leq \frac{2h(x_1)}{TI\eta\eta_g} + \frac{\eta\eta_g\bar{L}G_2^2}{b_x M} + 12I^2\kappa^2L^2\eta^2C_f + \frac{4I^2\kappa^2L^2\eta^2\sigma^2}{b_y}$$
$$+ \frac{4(3\kappa^2L^2 + \hat{L}^2)\gamma\sigma^2}{\mu b_y} + 2G_1^2 + 2(3\kappa^2L^2 + \hat{L}^2)(1-\mu\gamma)^{I_{dec}}C_0$$

where the expectation $\mathbb{E}_\xi$ is w.r.t *the noise of stochastic gradient. b_x, b_y denotes the batch size, and G_1, G_2 and C_0 are some constants.*

Please refer to Theorem C.8 in the Appendix for the proof. We can further control the noise terms in Theorem 3.7 by choosing the learning rates carefully:

Corollary 3.8. *Suppose we choose the learning rates as $\gamma = \min(\frac{1}{2L}, (\frac{1}{C_\gamma T})^{1/2})$, $\eta = \min\left(1, \left(\frac{8Ib_xM\bar{L}h(x_1)}{TG_2^2}\right)^{1/2}, \left(\frac{4\bar{L}h(x_1)}{C_\eta I^2 T}\right)^{1/3}\right)$ and $\eta_g = \frac{1}{2I\bar{L}}$, then we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big(\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\|\mathbb{E}_{\xi}[\bar{\Delta}\hat{x}_{t,i}]\|^2\big) = O\left(\frac{\kappa^3}{T} + \left(\frac{\kappa^5}{T}\right)^{1/2} + \left(\frac{\kappa^6}{T^2}\right)^{1/3} + \tilde{G}\right)$$

where $\tilde{G} = \kappa^2(1-\tau\mu)^{2(Q+1)} + \kappa^4(1-\mu\gamma)^{I_{dec}}$, C_η and C_γ are some constants.

To reach an ϵ stationary point, we need to run Algorithm 1 with $T = O(\kappa^5\epsilon^{-2})$ number of iterations, furthermore, we need $Q = O(\kappa\log(\frac{\kappa}{\epsilon}))$, $I_{dec} = O(\kappa\log(\frac{\kappa}{\epsilon}))$. Note that this matches the iteration complexity of standard FL algorithms [60] up to some logarithm factors (Q and I_{dec}). Note that Corollary 3.8 shows that the following term converges to 0: $\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_{\xi}[\bar{\Delta}\hat{x}_{t,i}]\|^2$. In fact, for the first term, it shows that x_t reaches to a stationary point of the bilevel problem $h(x)$, meanwhile, since we have:

$$\mathbb{E}_{\xi}[\bar{\Delta}\hat{x}_{t,i}] = \frac{1}{M}\sum_{m=1}^{M}\sum_{q=0}^{Q}\tau\nabla_{xy}g^{(m)}(x, y_{I_{dec}}^{(m)}) \times (I - \tau\nabla_{yy}g^{(m)}(x, y_{I_{dec}}^{(m)}))^q\nabla f^{(m)}(\hat{y}_i^{(m)})$$

Therefore, if $\|\nabla_{xy}g^{(m)}(x,y)\|$ is lower-bounded and $\|\nabla_{yy}g^{(m)}(x,y)\|$ is upper-bounded, we also get the learning parameter $\hat{y}_i^{(m)}$ converges to the stationary point of the local learning problem $f^{(m)}(y)$.

4 Application of FedSep to Real-world FL Problems

In FedSep, the separation of the communication layer and the learning layer makes it able to solve various challenges in Federated Learning. In this section, we apply FedSep to solve two challenging problems in Federated Learning: communication-efficient FL and model-heterogeneous FL.

4.1 Communication-efficient Federated Learning

Communication-cost is one of the major bottlenecks in Federated Learning due to slow connections between clients and the server. In Federated Learning, compression is commonly employed to mitigate communication costs, where clients compress the local updates before transferring to the server. A common compressor is choosing the Top-K coordinates, however, this approach is only good at reducing the upload communication cost. Since clients have different Top-K coordinates, the aggregated updates are often dense. More complicated approaches can save both upload and download communication cost, such as the Count-sketch based compressor [48]. In fact, we can develop a simple yet effective approach to reduce the communication cost based on FedSep.

In our FedSep framework, suppose we have the learning parameter $\theta \in \mathbb{R}^d$, and the communication parameter $\omega \in \mathbb{R}^p$. We choose $p \ll d$ to have a communication efficient federated learning algorithm. More specifically, we consider the following formulation:

$$\min_{\omega \in \mathbb{R}^p} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\theta_\omega^{(m)}; \mathcal{D}_{tr}^{(m)}) \ s.t.\ \theta_\omega^{(m)} = \underset{\theta \in \mathbb{R}^d}{\arg\min} \frac{1}{2} \big\| S^{(m)}\theta - \omega \big\|_2^2 + \beta \big\|\theta\big\|_1 \tag{4}$$

where $\mathcal{D}_{tr}^{(m)}$ denotes the training distribution of the m_{th} client; $\mathcal{L}(\cdot)$ denotes the loss function. In particular, $S^{(m)} \in \mathbb{R}^{p \times d}$ ($p \ll d$) is a random sketch matrix whose coordinates are sampled from Gaussian distribution. We choose the quadratic optimization problem (LASSO) as the decode function.

Eq. (4) is a special case of Eq. (1): θ corresponds to the learning parameter y, ω corresponds to the communication parameter x, $\mathcal{L}(\theta; \mathcal{D}_{tr}^{(m)})$ corresponds to the learning problem $f^{(m)}(y)$ and the decoding problem is a LASSO problem. We can solve it through Algorithm 1.

The LASSO problem of the decode operation involves a non-smooth L_1 regularization term and we solve it with the standard proximal gradient method [3]. For the encode operation, we have:

$$Enc^{(m)}\{\cdot\} = -\nabla_\omega \theta_\omega^{(m)} = \gamma S^{(m)} U \big(I - \big(I - \gamma (S^{(m)})^T S^{(m)}\big) U\big)^{-1} \tag{5}$$

where I is the identity matrix and $U = Diag\{\mathbb{I}_{\gamma\beta}(\theta_\omega^{(m)} + \gamma (S^{(m)})^T (\omega - S^{(m)} \theta_\omega^{(m)}))\}$ where $\mathbb{I}_{\gamma\beta}(\cdot)$ is an indicator operator, which outputs 1 if the absolute value of the input is greater than $\gamma\beta$. Please refer to Appendix A.1 for more details of the proximal gradient method and the encode operator.

Remark 2. If we set $\beta = 0$ in Eq. (4), *i.e.* we remove the sparsity constraints, both encode and decode operators have explicit solutions. More specifically, we have the decode operator $Dec^{(m)}\{\cdot\} := \big((S^{(m)})^T S^{(m)}\big)^{-1} (S^{(m)})^T$ and the encode operator $Enc^{(m)}\{\cdot\} := S^{(m)} \big((S^{(m)})^T S^{(m)}\big)^{-1}$. If we choose $S^{(m)} = S$ for all $m \in [M]$, we get a linear compressor.

4.2 Model-Heterogeneous Federated Learning

In practical Federated Learning applications, the scale of the model is inherently limited by the on-device resources of the participating clients, which often exhibit significant diversity. As a result, we can choose different scale of models based on the available resources of each individual client. One widely-used approach is the sub-model extraction, *i.e.* each client selects a part of the server model to perform local training. Different strategies can be used to do extraction: fixed [12, 2], random [6] and roll [1]. In contrast, we provide a data-dependent approach to select the sub-models. As shown by the following formulation:

$$\begin{aligned} &\min_{\omega \in \mathbb{R}^p} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}(\theta_\omega^{(m)}; \mathcal{D}_{tr}^{(m)}) \\ &s.t.\ \theta_\omega^{(m)} = a_\omega^{(m)} \odot \omega,\ a_\omega^{(m)} = \underset{a \in \{0,1\}^p}{\arg\min} \mathcal{L}(a \odot \omega; \mathcal{D}_{val}^{(m)}) + \beta \mathcal{R}(T(a), p^{(m)} T_{tol}) \end{aligned} \tag{6}$$

where $\mathcal{D}_{tr}^{(m)}$ denotes the training distribution; $\mathcal{L}(\cdot)$ denotes the loss function and θ and ω are the learning parameter. Note that ω denotes the parameter of the full model and θ is the parameter of the sub-model. In particular, we denote a mask vector $a_\omega^{(m)}$, whose value is from $\{0, 1\}$. $\odot$ represents the coordinate-wise multiplication.

For the decode operation, we have $\theta_\omega^{(m)} = a_\omega^{(m)} \odot \omega$. So we first need to find an optimal mask $a_\omega^{(m)}$, instead of solving the complicated integer programming as in Eq. (6), we solve the following relaxed continuous problem:

$$a_\omega^{(m)} = \underset{a \in [0,1]^p}{\arg\min} \mathcal{L}(a \odot \omega; \mathcal{D}_{val}^{(m)}) + \beta \mathcal{R}(T(a), p^{(m)} T_{tol}) \tag{7}$$

Eq. (7) includes two parts. $\mathcal{L}(a \odot \omega; \mathcal{D}_{val}^{(m)})$ measures the loss of the extracted sub-model over $\mathcal{D}_{val}^{(m)}$. The second part $\mathcal{R}(T(a), p^{(m)} T_{tol})$ is called resource loss [17]. T_{tol} denotes FLOPS of

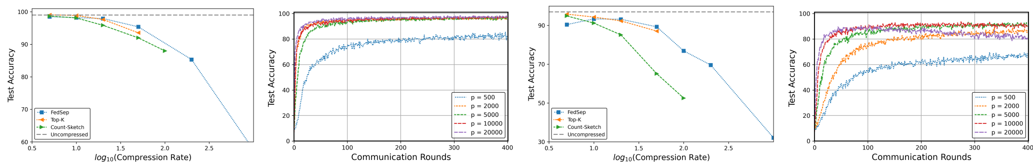

Figure 2: Test Accuracy w.r.t Communication Rate for FedSep and other baseline methods for MNIST Dataset. The first two plots show results under the I.I.D case, and the last two plots show results for the Non-I.I.D case. p in the second and the fourth plot is the dimension of the communication parameter. The local learning steps are set as $I = 5$.

the server model and $T(a)$ denotes FLOPS of the sub-model. Therefore, $p^{(m)} \in [0, 1]$ controls the size of the sub-model. Following [17], we choose $\mathcal{R}(\cdot)$ as: $\mathcal{R}(T(a), p^{(m)}T_{tol}) = \log(max(T(a), p^{(m)}T_{tol})/p^{(m)}T_{tol})$. In summary, we choose the 'best' sub-model which keeps the loss value small and meanwhile, satisfies the FLOPS constraint. Note that this is in contrast to the data-independent sub-model selection methods [12, 6, 1].

Next for the encode operator $Enc\{\cdot\}$, we follow the definition of Eq. (2) to have:

$$Enc^{(m)}\{\cdot\} = -\nabla_{\omega}\theta_{\omega}^{(m)} = -\nabla_{\omega}\big(a_{\omega}^{(m)} \odot \omega\big) = -Diag\{a_{\omega}^{(m)}\} - Diag\{\omega\}\nabla_{\omega}a_{\omega}^{(m)} \tag{8}$$

Where the last equality follows from the chain rule. Finally, combine Eq. (7) and Eq. (8), we can solve Eq. (6) with Algorithm 1.

Remark 3. For the encode operator Eq. (8), we have $Enc^{(m)}(\cdot) \approx -Diag\{a_{\omega}^{(m)}\}$ by omitting the second term in Eq. (8), where $\Delta\theta$ denotes the updates to the sub-model θ. As shown in Line 21 of Algorithm 1, we have the update of the server model be $\Delta\omega = \frac{1}{|\mathcal{M}_t|}\sum_{m\in\mathcal{M}_t}\eta_g Diag\{a_{\omega}^{(m)}\}\Delta\theta$. In other words, the server simply aggregates the updates of the sub-model, this is the aggregation rule widely used in the partial training literature [12, 6, 1].

5 Numerical Experiments

In this section, we perform numerical experiments to validate the efficacy of FedSep in solving communication-efficient FL and model-heterogeneous FL as discussed in Section 4. The code is written with Pytorch [45], and the Federated Learning environment is simulated via Pytorch.Distributed Package. We used servers with AMD EPYC 7763 64-core CPU and 8 NVIDIA V100 GPUs to run our experiments.

5.1 Experiments for Communication-Efficient Federated Learning

In this section, we provide numerical experiments for the communication-efficient FL task discussed in Section 4.1. We first introduce the experimental settings.

Dataset. We consider MNIST [32] and CIFAR-10 [30] in our experiments. We create the federated version of these datasets by evenly distributing the training samples among 10 clients. We consider an I.I.D setting and a Non-I.I.D setting. For the I.I.D setting, each client has data samples from all 10 classes, while for the Non-I.I.D setting, each client only has data samples from 2 classes.

Model. For experiments related to MNIST, we use a four-layer convolutional neural network. The first three layers contain 32 kernels of size 3×3 and the fourth layer contains 32 kernels of size 2×2. For the experiments related to CIFAR-10, we increase the number of kernels to 64. The total number of parameters of the model is around 10^5. For the sketch matrix $S^{(m)}$ of our FedSep, we let all clients choose the same sketch matrix, and we test the dimension of the communication parameter $p \in \{10, 100, 500, 1000, 2000, 5000, 10000, 20000\}$.

Baselines and Hyper-parameters. For baselines, we first compare with the uncompressed baseline (the communication parameter and the learning parameter are in the same dimension); Next we compare with the local Top-K compressor (with error feedback [53]) and the Count-Sketch compressor [48]. Please refer to the Appendix for the hyper-parameter selection of FedSep and baselines.

Table 1: **Test accuracy comparison between FedSep with other model-heterogeneous FL baseline methods.** High data heterogeneity represents $K = 2$ for CIFAR-10 and $K = 20$ for CIFAR-100; Lower data heterogeneity represents $K = 5$ for CIFAR-10 and $K = 50$ for CIFAR-100.

	Method	**High Data Heterogeneity**		**Low Data Heterogeneity**	
		CIFAR-10	**CIFAR-100**	**CIFAR-10**	**CIFAR-100**
KD-based	FedDF [38]	73.81 (± 0.42)	31.87 (± 0.46)	76.55 (± 0.32)	37.87 (± 0.31)
	DS-FL [24]	65.27 (± 0.53)	29.12 (± 0.51)	68.44 (± 0.47)	33.56 (± 0.55)
	Fed-ET [10]	**78.66 (± 0.31)**	**35.78 (± 0.45)**	**81.13 (± 0.28)**	**41.58 (± 0.36)**
PT-based	HeteroFL [12]	63.90 (± 2.74)	52.38 (± 0.80)	73.19 (± 1.71)	57.44 (± 0.42)
	Federated Dropout [6]	46.64 (± 3.05)	45.07 (± 0.07)	76.20 (± 2.53)	46.40 (± 0.21)
	ZeroFL [47]	64.61 (± 2.18)	51.39 (± 0.45)	83.31 (± 0.78)	53.62 (± 0.51)
	FedDST [5]	67.65 (± 1.27)	54.21 (± 0.34)	84.57 (± 0.28)	54.97 (± 0.44)
	Flash [2]	67.08 (± 1.46)	54.92 (± 0.29)	84.61 (± 0.37)	55.04 (± 0.32)
	FedRolex [1]	69.44 (± 1.50)	56.57 (± 0.15)	84.45 (± 0.36)	58.73 (± 0.33)
	FedSep (Ours)	**71.13 (± 0.94)**	**58.16 (± 0.25)**	**84.61 (± 0.37)**	**61.41 (± 0.29)**
	Homogeneous (smallest)	38.82 (± 0.88)	12.69 (± 0.50)	46.86 (± 0.54)	19.70 (± 0.34)
	Homogeneous (largest)	**75.74 (± 0.42)**	**60.89 (± 0.60)**	**84.48 (± 0.58)**	**62.51 (± 0.20)**

The experimental results are summarized in Figure 2 for MNIST (Figure 3 in the Appendix for CIFAR-10). As shown by the figures, FedSep outperforms other baselines. In particular, our FedSep achieves much better performance in the high-compression-rate regime and can get similar performance as other baselines in the low-compression-rate regime. Please refer to the Appendix for more ablation studies.

5.2 Experiments for Model-Heterogeneous Federated Learning

In this section, we provide numerical experiments for the task of a model-heterogeneous FL task discussed in Section 4.2. First, we introduce the experimental settings.

Dataset. We consider CIFAR-10 and CIFAR-100 [30] in our experiments. We create Federated datasets by evenly distributing images among clients; we have 100 clients in our experiments. We create data heterogeneity by controlling the number of classes K a client can have [1]. For CIFAR-10, we test $K \in \{2, 4, 5, 8\}$, while for CIFAR-100, we test $K \in \{5, 10, 20, 50\}$. Note that smaller values of K mean higher heterogeneity of the data. For FedSep, the validation and training set are the same.

Model. We choose ResNet-18 [19] in the experiments. Following the setting in [1, 12], we replace the batch normalization layer with static batch normalization and add a scalar module after each convolution layer. Instead of using the coordinate mask as defined in Eq. (6), we select kernels in each convolutional layer of ResNet-18; furthermore, we parameterize the mask a with a neural network $HN(\phi)$ [17], *i.e.* $a = HN(\phi)$; finally, we reuse the masks and only update the masks every two epochs, this stabilizes the training. This reduces the dimension of a that we need to optimize. In experiments, the size of the submodel p (Eq. (6)) of a client is randomly chosen from $\{1, 0.5, 0.25, 0.125, 0.0625\}$, where the ratio p is *w.r.t* the server model. For our FedSep, after optimizing Eq. (7), we select the top p percent kernels of each convolutional layer by the value of the leaned mask a, and then we only use the selected kernels to perform the training. In the encode operation, we only evaluate the first term of Eq. (8) as stated in Remark 3.

Baselines and Hyper-parameters. We compare with both state-of-the-art Knowledge Distillation-based methods: FedDF [38], DS-FL [24] and Fed-ET [10], and Partial Training Based methods: HeteroFL [12], ZeroFL [47], Federated Dropout [6], FedDST [5], Flash [2] and FedRolex [1]. Fjord [21] gets similar performance as HeteroFL, so we do not include it in the comparision. For ZeroFL, FedDST and Flash, we consider their heterogeneous-model version by varying the compression rate among clients.

The experimental results are summarized in Table 1. As shown in the table, our FedSep gets comparable performance with the Knowledge Distillation based methods, which needs additional public data, furthermore, FedSep outperforms the partial training based methods, including the state-of-the-art FedRolex [1] method. This result shows the effectiveness of the adaptive sub-model extraction strategy used by our FedSep. Our method chooses the most important part of the model at each step for clients, thus our method converges faster than other data-independent methods. For more experimental results, please refer to the Appendix.

6 Conclusion

In this work, we propose a novel federated learning framework, *i.e.* FedSep with separated communication and learning components. Based on the observation that communication and learning set opposite requirements to the parameter, we let clients to have two layers: a communication layer and a learning layer. We formulate the decode operation from the communication layer to the learning layer as solving a minimization problem, therefore, FedSep has a bilevel structure. We propose an efficient algorithm to solve this bilevel problem and also prove that the algorithms converge to a stationary point with rate $O(\epsilon^{-2})$. Finally, we apply FedSep to solve the communication-efficient FL task and the model heterogeneous FL task. Numerical experiments show the superior performance of our FedSep over various baselines.

References

[1] S. Alam, L. Liu, M. Yan, and M. Zhang. Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction. *arXiv preprint arXiv:2212.01548*, 2022.

[2] S. Babakniya, S. Kundu, S. Prakash, Y. Niu, and S. Avestimehr. Federated sparse training: Lottery aware model compression for resource constrained edge. *arXiv preprint arXiv:2208.13092*, 2022.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[4] Y. Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[5] S. Bibikar, H. Vikalo, Z. Wang, and X. Chen. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6080–6088, 2022.

[6] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 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] D. Chen and M. T. Hagan. Optimal use of regularization and cross-validation in neural network modeling. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 2, pages 1275–1280. IEEE, 1999.

[9] T. Chen, Y. Sun, and W. Yin. A single-timescale stochastic bilevel optimization method. *arXiv preprint arXiv:2102.04671*, 2021.

[10] Y. J. Cho, A. Manoel, G. Joshi, R. Sim, and D. Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*, 2022.

[11] M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *arXiv preprint arXiv:2201.13409*, 2022.

[12] E. Diao, J. Ding, and V. Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

[13] C. B. Do, C.-S. Foo, and A. Y. Ng. Efficient multiple hyperparameter learning for log-linear models. In *NIPS*, volume 2007, pages 377–384. Citeseer, 2007.

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

[15] M. C. Ferris and O. L. Mangasarian. Finite perturbation of convex programs. *Applied Mathematics and Optimization*, 23(1):263–273, 1991.

[16] 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.

[17] S. Gao, F. Huang, W. Cai, and H. Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021.

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

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

[21] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *Advances in Neural Information Processing Systems*, 34:12876–12889, 2021.

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

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

[24] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2021.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

[31] J. Larsen, L. K. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71. IEEE, 1996.

[32] Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

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

[34] J. Li, B. Gu, and H. Huang. A fully single loop algorithm for bilevel optimization without hessian inverse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7426–7434, 2022.

[35] J. Li, F. Huang, and H. Huang. Communication-efficient federated bilevel optimization with local and global lower level problems. *arXiv preprint arXiv:2302.06701*, 2023.

[36] J. Li, J. Pei, and H. Huang. Communication-efficient robust federated learning with noisy labels. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 914–924, 2022.

[37] 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.

[38] T. Lin, L. Kong, S. U. Stich, and M. Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

[39] 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.

[40] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

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

[42] 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.

[43] M. Mohri, G. Sivek, and A. T. Suresh. Agnostic federated learning. *arXiv preprint arXiv:1902.00146*, 2019.

[44] 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.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

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

[47] X. Qiu, J. Fernandez-Marques, P. P. Gusmao, Y. Gao, T. Parcollet, and N. D. Lane. Zerofl: Efficient on-device training for federated learning with local sparsity. *arXiv preprint arXiv:2208.02507*, 2022.

[48] 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.

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

[50] 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.

[51] A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots. Truncated back-propagation for bilevel optimization. *arXiv preprint arXiv:1810.10667*, 2018.

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

[53] S. U. Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

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

[55] D. A. Tarzanagh, M. Li, C. Thrampoulidis, and S. Oymak. Fednest: Federated bilevel, minimax, and compositional optimization. *arXiv preprint arXiv:2205.02215*, 2022.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[57] 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.

[58] 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.

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

[60] B. Woodworth. The minimax complexity of distributed optimization. *arXiv preprint arXiv:2109.00534*, 2021.

[61] I. Yamada, M. Yukawa, and M. Yamagishi. Minimizing the moreau envelope of nonsmooth convex functions over the fixed point set of certain quasi-nonexpansive mappings. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 345–390. Springer, 2011.

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

A More Experimental Results

A.1 Communication-efficient FL

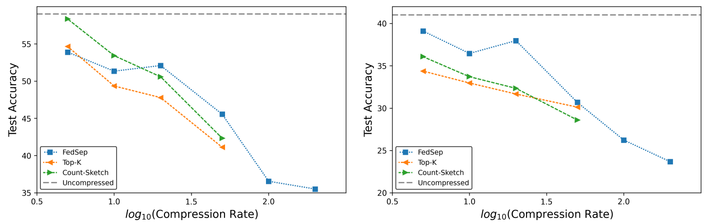

Figure 3: Test Accuracy w.r.t Communication Rate for FedSep and other baseline methods for CIFAR-10 Dataset. The left figure shows results under the I.I.D case, and the right figure show results for the Non-I.I.D case. The local learning steps are set as $I = 5$.

In this section, we include more details related to the Communication-efficient FL. We first introduce details of some definitions.

Soft-threshold Operator. We use the definition of the soft threshold operator $Q_{\gamma\beta}(\cdot)$ in Section 4.1, formally, we have the following definition:

$$Q_{\gamma\lambda}(x) = sign(x)max(abs(x) - \gamma\lambda, 0) \tag{9}$$

Proximal Gradient Descent. we solve $\theta_\omega^{(m)}$ through the following update rule:

$$\theta^+ = Q_{\gamma\beta}\big(\theta + \gamma(S^{(m)})^T(\omega - S^{(m)}\theta)\big) \tag{10}$$

where $Q_{\gamma\beta}(\cdot)$ is the soft-threshold operator and its definition is in Eq. (9) and γ is some constant. We denote the generalized gradient based on Eq. (10) as:

$$G^*(\theta) = \frac{1}{\gamma}\big(\theta - \theta^+\big) \tag{11}$$

In summary, for the decode operation $Dec\{\cdot\}$ in Eq. (4), we solve decode problem through gradient descent with the gradient operator defined by Eq. (11). As for the encode operation, we have $G^*(\theta_\omega^{(m)}) = 0$ by definition of $\theta_\omega^{(m)}$, then take derivation over ω on both sides, and use the chain rule, we get the expression in Eq. (5).

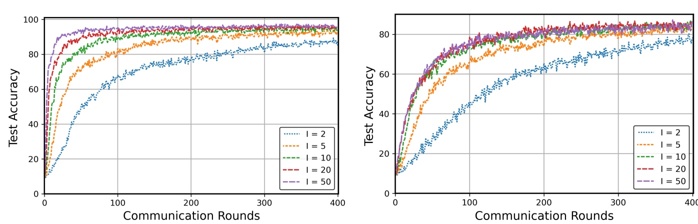

Figure 4: Test Accuracy w.r.t Communication Round for FedSep over the MNIST Dataset. The left figures shows results under the I.I.D case, and the right figure show results for the Non-I.I.D case. We vary the value of local step $I \in \{2, 5, 10, 20, 50\}$ in the figure, and we fix the dimension of the communication parameter as $p = 2000$.

Next, we introduce more details of experimental settings.

Baselines and Hyper-Parameter. We first compare with the uncompressed baseline (the communication parameter and the learning parameter are in the same dimension); Next we compare with the local Top-K compressor and the Count-Sketch compressor [48]. For the count-sketch, we tune the size of sketch table; a typical good value for the length of the sketch table is half of the communication parameter dimension. For our FedSep, we tune the values of β and γ. A typical good value for

γ is $[0.5, 5]$ and it depends on the dimension of the communication parameter p, for β, we choose $[0, 0.01]$. We use the SGD optimizer to optimize $\mathcal{L}(\theta_{\omega}^{(m)}; \mathcal{D}_{tr}^{(m)})$ for all methods, and set the learning rate at 0.1, and set $I_{dec} = 10$ for the decode operation, *i.e.* for solving the Lasso problem.

More Experimental Results. The experimental results for the CIFAR-10 dataset is included in Figure 3. We observe that FedSep also outperforms other baselines as in the MNIST data set. Finally, in Figure 4, we vary the number of local steps I under a fixed compression rate. As shown in the figure, our FedSep can benefit from more local steps. More specifically, for the homogeneous case, our FedSep can benefit from more local steps as large as $I = 50$, while for the heterogeneous case, increasing local steps brings little benefit to $I > 10$.

A.2 Model-Heterogeneous FL

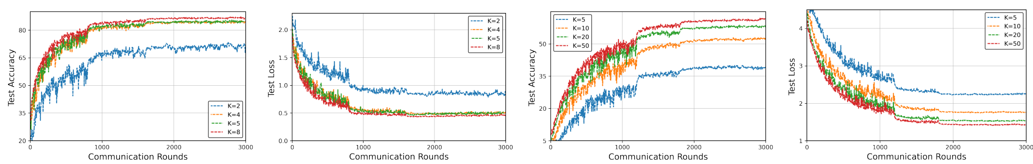

Figure 5: Test Accuracy and Loss w.r.t Communication Rounds for FedSep under different levels of data heterogeneity. K is the number of classes each client has. The first two plots show results for CIFAR-10, then the last two plots show results for CIFAR-100.

In this section, we introduce more details of the experimental setting for Model-Heterogeneous FL task and present more experimentsl results.

Hyper-parameters. For our FedSep, we perform grid search to search the best parameter setting. More specifically, we decode problem Eq. (7) with $\beta = 10$ and AdamW [40] with learning rate 0.01. Regarding the learning problem, for CIFAR-10, we use the Adam optimizer with learning rate 10^{-4} when $K = \{2, 4\}$ and 2×10^{-4} when $K \in \{5, 8\}$, for CIFAR-100, we use the Adam optimizer with learning rate 2×10^{-4} when $K = \{5, 10, 20, 50\}$. We decrease learning by a factor of 0.25 at the $\{800, 1600\}$ steps for the CIFAR-10 dataset and at the $\{1200, 1800\}$ global steps for the CIFAR-100 dataset.

More Experimental Results. Next, we test our FedSep at different levels of heterogeneity. The results are summarized in Figure 5. As shown by the figures, our FedSep is quite robust when clients have highly heterogeneous data.

B More 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 [27, 50, 43, 37], privacy [44, 42, 57] and communication bottleneck [58, 39, 54, 28, 25]. In particular, communication cost is one of the major bottlenecks in FL. A widely used approach to reduce communication burden is through compression: clients compress parameters before transmitting to the server. Various compressors [58, 39, 54, 28] have been studied in the literature. In particular, the local Top-K compressor selects the K largest coordinates of the parameter to transmit. From the view of FedSep: the vector of Top-K coordinates is the communication parameter, while the encode operation is finding these coordinates. Count Sketch [7] based compressors [25, 48] are also widely studied. Since the count sketch is linear, from the point of view of FedSep: count sketch performs a linear mapping between the communication parameter and the learning parameter. Next, model-heterogeneous FL is also studied to solve the device heterogeneity issue. There are two main categories of heterogeneous model methods: knowledge-distillation (KD) [38, 24, 10] based and partial-training (PT) [12, 6, 1] based. KD-based methods treat the server model as the teacher and the clients' model as the student. A disadvantage of this category is the dependence on a public dataset. In contrast, PT-based methods select a subset of the server model for each client, and in the aggregation phase, the server simply aggregates the updated sub-models. We also develop a PT-based method where we select the sub-models adaptively based on the clients' local data.

Bilevel Optimization. Bilevel problem is a type of two-layer optimization problem. It includes an upper-level problem and a lower-level problem, and the upper-level problem relies on the solution of the lower-level problems. The study of bilevel problems dates back at least to the 1960s [59] and is followed by [15, 52, 61, 49]. In the machine learning community, Hyper-parameter Optimization problems [31, 8, 4, 13] are one of the most studied bilevel problems. Early bilevel optimization algorithms solved the lower problem exactly to update the upper variable. Recently, researchers developed algorithms that solve the lower problem approximately with a fixed number of steps and use the back-propagation technique to compute the approximate hyper-gradient [14, 41, 16, 46, 51]. More recently, single-loop algorithms [18, 20, 26, 29, 9, 62, 34, 11, 22, 23] based on alternative updates of lower and upper level problems are proposed. However, less work studies bilevel problems under the federated setting. [55, 36, 35] studies a type of federated bilevel problems where the lower level problems are federated. In particular, [36] solves the noisy label problem in FL. Note that our FedSep is formulated as a type of federated bilevel problems, however, our problem is different from that in [55, 36], as our problem has a unique lower problem for each client.

C Proof for the convergence of Algorithm 1

In this section, we present the proofs for Algorithm 1. First, we denote some notation for clarity. Recall that we denote $\Delta\hat{y}^{(m)} = \hat{y}_I^{(m)} - \hat{y}_0^{(m)} = \sum_{i=0}^{I-1} -\eta\nabla f^{(m)}(\hat{y}_i^{(m)}; \mathcal{B}_{\hat{y}})$ to be the update of the learning problem on client m. Then we denote $\mu_i^{(m)}$ as:

$$\mu_i^{(m)} = -\nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big(\nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)})$$

and denote $\mu^{(m)}$ as:

$$\mu^{(m)} = -\nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big(\nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)})\big)^{-1}\Delta\hat{y}^{(m)}$$

while in Algorithm 1, we use the stochastic encoder $\widetilde{Enc}\{\cdot\}$ to compute $\Delta\hat{x}_t^{(m)}$,

$$\Delta\hat{x}_t^{(m)} = \sum_{q=0}^{Q} \tau\nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x)(I - \tau\nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x))^q\Delta\hat{y}^{(m)}$$

Since $x^{(m)}$ is not updated during local steps, we omit the superscript when clear from the context. We further denote $\Delta\hat{x}_{t,i}^{(m)}$ as

$$\Delta\hat{x}_{t,i}^{(m)} = -\sum_{q=0}^{Q} \tau\nabla_{xy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x)(I - \tau\nabla_{yy} g^{(m)}(x, y_{I_{dec}}^{(m)}; \mathcal{B}_x))^q\nabla f^{(m)}(\hat{y}_i^{(m)}; \mathcal{B}_{\hat{y}})$$

Lastly, recall the hyper-gradient of $h^{(m)}(x)$ has the form of:

$$\nabla h^{(m)}(x) = -\nabla_{xy} g^{(m)}(x, y_x^{(m)})\big(\nabla_{yy} g^{(m)}(x, y_x^{(m)})\big)^{-1}\nabla f^{(m)}(y_x^{(m)})$$

Assumption C.1 (Assumption 4)**.** We have unbiased stochastic first order and second order derivative oracle with bounded variance, more specifically, denote $z = (x, y)$, we have:

a) we have $\nabla f^{(m)}(y;\xi)$, such that: $E[\nabla f^{(m)}(y;\xi)] = \nabla f^{(m)}(y)$ and $var(\nabla f^{(m)}(y;\xi)) \leq \sigma^2$;

b) we have $\nabla g^{(m)}(z;\xi)$, such that: $E[\nabla g^{(m)}(z;\xi)] = \nabla g^{(m)}(z)$ and $var(\nabla g^{(m)}(z;\xi)) \leq \sigma^2$;

c) we have $\nabla_{y^2} g^{(m)}(z,\xi)$, such that: $E[\nabla_{y^2} g^{(m)}(z;\xi)] = \nabla_{y^2} g^{(m)}(z)$ and $var(\nabla_{y^2} g^{(m)}(z;\xi)) \leq \sigma^2$;

d) we have $\nabla_{xy} g^{(m)}(z;\xi)$, such that: $E[\nabla_{xy} g^{(m)}(z;\xi)] = \nabla_{xy} g^{(m)}(z)$ and $var(\nabla_{xy} g^{(m)}(z;\xi)) \leq \sigma^2$;

Proposition C.2. *Suppose Assumptions 3.2 and 3.3 hold, the following statements hold:*

a) $\|\mu_i^{(m)} - \nabla h^{(m)}(x)\|^2 \leq 2\hat{L}^2\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + 2\kappa^2 L^2\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2$*, where* $\hat{L} = O(\kappa^2)$.

b) $h^{(m)}(x)$ is Lipschitz continuous in x with constant $\bar{L}$ i.e., for any given $x_1, x_2 \in \mathbb{R}^p$, we have $\|\nabla h^{(m)}(x_2) - \nabla h^{(m)}(x_1)\|^2 \leq \bar{L}^2\|x_2 - x_1\|$ where $\bar{L} = O(\kappa^3)$.

where we denote the condition number as $\kappa = L/\mu$.

Proof. We prove the Part a) here. Proof of b) and can be referred in Lemma 2.2 of [18].

$$
\begin{aligned}
&\|\mu_i^{(m)} - \nabla h^{(m)}(x)\| \\
&= \left\|\nabla_{xy} g(x, y_{I_{dec}}^{(m)})\Big(\nabla_{yy} g(x, y_{I_{dec}}^{(m)})\Big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)}) - \nabla_{xy} g(x, y_x^{(m)})\Big(\nabla_{yy} g(x, y_x^{(m)})\Big)^{-1}\nabla f^{(m)}(y_x^{(m)})\right\| \\
&\leq \left\|\nabla_{xy} g(x, y_{I_{dec}}^{(m)}) - \nabla_{xy} g(x, y_x^{(m)})\right\|\left\|\Big(\nabla_{yy} g(x, y_x^{(m)})\Big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)})\right\| \\
&\quad + \left\|\nabla_{xy} g(x, y_x^{(m)})\right\|\left\|\Big(\nabla_{yy} g(x, y_{I_{dec}}^{(m)})\Big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)}) - \Big(\nabla_{yy} g(x, y_x^{(m)})\Big)^{-1}\nabla f^{(m)}(y_x^{(m)})\right\| \\
&\leq \frac{C_f L_{xy}}{\mu}\left\|y_{I_{dec}}^{(m)} - y_x^{(m)}\right\| + L\left\|\Big(\nabla_{yy} g(x, y_{I_{dec}}^{(m)})\Big)^{-1}\nabla f^{(m)}(\hat{y}_i^{(m)}) - \Big(\nabla_{yy} g(x, y_x^{(m)})\Big)^{-1}\nabla f^{(m)}(y_x^{(m)})\right\| \\
&\leq \frac{C_f L_{xy}}{\mu}\left\|y_{I_{dec}}^{(m)} - y_x^{(m)}\right\| + C_f L\left\|\Big(\nabla_{yy} g(x, y_{I_{dec}}^{(m)})\Big)^{-1} - \Big(\nabla_{yy} g(x, y_x^{(m)})\Big)^{-1}\right\| + \kappa\left\|\nabla f^{(m)}(\hat{y}_i^{(m)}) - \nabla f^{(m)}(y_x^{(m)})\right\| \\
&\leq \left(\frac{C_f L_{xy}}{\mu} + \frac{\kappa C_f L_{yy}}{\mu}\right)\left\|y_{I_{dec}}^{(m)} - y_x^{(m)}\right\| + \kappa\left\|\nabla f^{(m)}(\hat{y}_i^{(m)}) - \nabla f^{(m)}(y_x^{(m)})\right\|
\end{aligned}
$$

Suppose we denote $\hat{L} = \left(\frac{C_f L_{xy}}{\mu} + \frac{\kappa C_f L_{yy}}{\mu}\right)$, then we have:

$$
\left\|\mu_i^{(m)} - \nabla h^{(m)}(x)\right\|^2 \leq 2\hat{L}^2\left\|y_{I_{dec}}^{(m)} - y_x^{(m)}\right\|^2 + 2\kappa^2\left\|\Delta\nabla f^{(m)}(\hat{y}_i^{(m)}) - \nabla_y f(y_x^{(m)})\right\|^2
$$

which completes the proof. □

Proposition C.3. *(Lemma 4 and 7 in [62]) Suppose Assumptions 3.2, 3.3 and 3.4 hold and $\tau < \frac{1}{L}$, we have*

a) $\|\mathbb{E}_\xi[\Delta\hat{x}_{t,i}^{(m)}] - \mu_i^{(m)}\| \leq G_1$, where $G_1 = \kappa(1 - \tau\mu)^{Q+1}C_f$

b) $\mathbb{E}\|\Delta\hat{x}_{t,i}^{(m)} - \mathbb{E}_\xi[\Delta\hat{x}_{t,i}^{(m)}]\|^2 \leq G_2^2$, where $G_2^2 = (2C_f^2 + 12C_f^2L^2\tau^2(Q+1)^2 + 4C_f^2L^2(Q+2)(Q+1)^2\tau^4\sigma^2)/b_x$

where we assume the mini-batch has $|\mathcal{B}_x| = b_x$

C.1 Lower Problem Solution Error

Lemma C.4. *When $\gamma < \frac{1}{L}$, we have:*

$$
\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left\|y_{I_{dec}}^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 \leq \frac{(1-\mu\gamma)^{I_{dec}}}{M}\sum_{m=1}^{M}\mathbb{E}\left\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 + \frac{2\gamma\sigma^2}{\mu b_y}
$$

Proof. First, follow the property of the strong convex function, we have:

$$
\mathbb{E}\left\|y_i^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 \leq (1-\mu\gamma)\mathbb{E}\left\|y_{i-1}^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 + \frac{2\gamma^2\sigma^2}{b_y}
$$

where we choose $\gamma < 1/L$. Next, we telescope from $i = 1 \to I_{dec}$ to have

$$
\begin{aligned}
\mathbb{E}\left\|y_{I_{dec}}^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 &\leq (1-\mu\gamma)^{I_{dec}}\mathbb{E}\left\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 + \frac{2\gamma^2\sigma^2}{b_y}\sum_{i=0}^{I}(1-\mu\gamma)^i \\
&\leq (1-\mu\gamma)^{I_{dec}}\mathbb{E}\left\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\right\|^2 + \frac{2\gamma\sigma^2}{\mu b_y}
\end{aligned}
$$

Average over all clients, we get the claim in the lemma. □

Lemma C.5. *For $I \geq 1$, than we have:*

$$\frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2 \leq \frac{3I}{M}\sum_{m=1}^{M}\mathbb{E}\|\hat{y}_0^{(m)} - y_x^{(m)}\|^2 + 6I^2\eta^2\frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\|\nabla f^{(m)}(\hat{y}_i^{(m)})\|^2 + \frac{2I^3\eta^2\sigma^2}{b_y}$$

Proof. By the update rule in Algorithm 1, we have:

$$\begin{aligned}
\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2 &= \mathbb{E}\|\hat{y}_{i-1}^{(m)} - \eta\nabla f^{(m)}(\hat{y}_{i-1}^{(m)}; \mathcal{B}_{\hat{y}}) - y_x^{(m)}\|^2 \\
&\leq (1+\frac{1}{I})\mathbb{E}\|\hat{y}_{i-1}^{(m)} - y_x^{(m)}\|^2 + (1+I)\eta^2\|\nabla f^{(m)}(\hat{y}_{i-1}^{(m)}; \mathcal{B}_{\hat{y}})\|^2 \\
&\leq (1+\frac{1}{I})\mathbb{E}\|\hat{y}_{i-1}^{(m)} - y_x^{(m)}\|^2 + 2I\eta^2\|\nabla f^{(m)}(\hat{y}_{i-1}^{(m)}; \mathcal{B}_{\hat{y}})\|^2 \\
&\leq (1+\frac{1}{I})\mathbb{E}\|\hat{y}_{i-1}^{(m)} - y_x^{(m)}\|^2 + 2I\eta^2\|\nabla f^{(m)}(\hat{y}_{i-1}^{(m)})\|^2 + \frac{2I\eta^2\sigma^2}{b_y}
\end{aligned}$$

where the first inequality is by the generalized inequality. Next we telescope over i, to obtain:

$$\begin{aligned}
\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2 &\leq \sum_{j=1}^{i}(1+\frac{1}{I})^{i-j}\big(2I\eta^2\|\nabla f^{(m)}(\hat{y}_j^{(m)})\|^2 + \frac{2I\eta^2\sigma^2}{b_y}\big) + (1+\frac{1}{I})^i\mathbb{E}\|\hat{y}_0^{(m)} - y_x^{(m)}\|^2 \\
&\leq (1+\frac{1}{I})^I\sum_{i=1}^{I}\big(2I\eta^2\|\nabla f^{(m)}(\hat{y}_i^{(m)})\|^2 + \frac{2I\eta^2\sigma^2}{b_y}\big) + (1+\frac{1}{I})^I\mathbb{E}\|\hat{y}_0^{(m)} - y_x^{(m)}\|^2 \\
&\leq 3\mathbb{E}\|\hat{y}_0^{(m)} - y_x^{(m)}\|^2 + 6I\eta^2\sum_{i=1}^{I}\|\nabla f^{(m)}(\hat{y}_i^{(m)})\|^2 + \frac{2I^2\eta^2\sigma^2}{b_y}
\end{aligned}$$

The third inequality uses the inequality $log(1+a/x) \leq a/x$ for $x > -a$, so we have $(1+a/x)^x \leq e^a$, Then we choose $a = 1$ and $x = I$. Finally, we use the fact that $e \leq 3$. It completes the proof by summing over all i. □

C.2 Descent Lemma

Lemma C.6. *For all $t \in [T]$, the iterates generated satisfy:*

$$\mathbb{E}\Big\|\nabla h(x_t) - \mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\Big\|^2 \leq \frac{1}{M}\sum_{m=1}^{M}\Big(2\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + 2\kappa^2L^2\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2\Big) + 2G_1^2$$

Proof. By definition of $\bar{\Delta}\hat{x}_{t,i}$, $\mu_{t,i}^{(m)}$ and $\nabla h(x_t)$, we have:

$$\begin{aligned}
&\mathbb{E}\Big\|\nabla h(x_t) - \mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\Big\|^2 \\
&\overset{(a)}{\leq} \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\Big\|\mathbb{E}_\xi[\Delta\hat{x}_t^{(m)}] - \nabla h^{(m)}(x_t)\Big\|^2 \\
&\leq \frac{2}{M}\sum_{m=1}^{M}\mathbb{E}\Big[\Big\|\mathbb{E}_\xi[\Delta\hat{x}_{t,i}^{(m)}] - \mu_{t,i}^{(m)}\Big\|^2 + \Big\|\mu_{t,i}^{(m)} - \nabla h^{(m)}(x_t)\Big\|^2\Big] \\
&\overset{(b)}{\leq} \frac{1}{M}\sum_{m=1}^{M}\Big(2\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + 2\kappa^2\mathbb{E}\|\nabla f^{(m)}(\hat{y}_i^{(m)}) - \nabla f^{(m)}(y_x^{(m)})\|^2\Big) + 2G_1^2 \\
&\leq \frac{1}{M}\sum_{m=1}^{M}\Big(2\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + 2\kappa^2L^2\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2\Big) + 2G_1^2
\end{aligned}$$

where inequality (a) follows the generalized triangle inequality; inequality (b) follows the Proposition C.2 and Proposition C.3.

□

Lemma C.7. *Suppose $\eta\eta_g \leq \frac{1}{2I\bar{L}}$ For $t \in [T]$, the iterates generated satisfy:*

$$\mathbb{E}[h(x_{t+1})] \leq \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 - \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\left\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\right\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2$$
$$+ \frac{\eta\eta_g}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\left(\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + \kappa^2L^2\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2\right)$$

where the expectation is w.r.t the stochasticity of the algorithm.

Proof. Using the smoothness of f we have:

$$\begin{aligned}
\mathbb{E}[h(x_{t+1})] &\leq \mathbb{E}[h(x_t)] + \mathbb{E}\langle\nabla h(x_t), x_{t+1} - x_t\rangle + \frac{\bar{L}}{2}\mathbb{E}\|x_{t+1} - x_t\|^2 \\
&\overset{(a)}{=} \mathbb{E}[h(x_t)] - \eta_g\mathbb{E}\langle\nabla h(x_t), \mathbb{E}_\xi[\bar{\Delta}\hat{x}_t]\rangle + \frac{\eta_g^2\bar{L}}{2}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_t]\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} \\
&= \mathbb{E}[h(x_t)] - \eta_g\eta\sum_{i=1}^{I}\mathbb{E}\langle\nabla h(x_t), \mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\rangle + \frac{I\eta^2\eta_g^2\bar{L}}{2}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} \\
&\overset{(b)}{=} \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{\eta\eta_g}{2}\sum_{i=1}^{I}\mathbb{E}\|\nabla h(x_t) - \mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2 \\
&\quad - \left(\frac{\eta\eta_g}{2} - \frac{I\eta^2\eta_g^2\bar{L}}{2}\right)\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} \\
&\overset{(c)}{\leq} \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 - \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\left\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\right\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2 \\
&\quad + \frac{\eta\eta_g}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\left(\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + \kappa^2L^2\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2\right)
\end{aligned}$$

where equality (a) follows from the iterate update given in Step 21 of Algorithm 1; (b) uses $\langle a, b\rangle = \frac{1}{2}[\|a\|^2 + \|b\|^2 - \|a - b\|^2]$; (c) follows the condition that $I\eta\eta_g \leq 1/2\bar{L}$ and Lemma C.6. This completes the proof. ☐

C.3 Proof of Convergence Theorem

We prove the convergence of Algorithm 1 in this section.

Theorem C.8. *Suppose we the learning rates*

$$\eta = \min\left(1, \left(\frac{8Ib_xM\bar{L}h(x_1)}{TG_2^2}\right)^{1/2}, \left(\frac{4\bar{L}h(x_1)}{C_\eta I^2T}\right)^{1/3}\right),\ \gamma = \min\left(\frac{1}{2L}, \left(\frac{1}{C_\gamma T}\right)^{1/2}\right)$$

and $\eta_g = \frac{1}{2I\bar{L}}$, then we have:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\right) \\
&= O\left(\frac{\kappa^3}{T} + \left(\frac{\kappa^5}{T}\right)^{1/2} + \left(\frac{\kappa^6}{T^2}\right)^{1/3} + \kappa^2(1-\tau\mu)^{2(Q+1)} + \kappa^4(1-\mu\gamma)^{I_{dec}}\right)
\end{aligned}$$

To reach an ϵ stationary point, we choose $Q = O(\kappa\log(\frac{\kappa}{\epsilon}))$, $I_{dec} = O(\kappa\log(\frac{\kappa}{\epsilon}))$ and $T = O(\kappa^3\epsilon^{-1.5})$ number of iterations.

Proof. By Lemma C.7, and combine with Lemma C.4 and Lemma C.5 to have:

$$\mathbb{E}[h(x_{t+1})] \le \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 - \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\Big\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\Big\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2$$
$$+ \frac{\eta\eta_g}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\Big(\hat{L}^2\mathbb{E}\|y_{I_{dec}}^{(m)} - y_x^{(m)}\|^2 + \kappa^2L^2\mathbb{E}\|\hat{y}_i^{(m)} - y_x^{(m)}\|^2\Big)$$
$$\le \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 - \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\Big\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\Big\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2$$
$$+ \frac{I\hat{L}^2\eta\eta_g(1-\mu\gamma)^{I_{dec}}}{M}\sum_{m=1}^{M}\mathbb{E}\Big\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\Big\|^2 + \frac{2I\hat{L}^2\eta\eta_g\gamma\sigma^2}{\mu b_y} + \frac{2I^3\kappa^2L^2\eta^3\eta_g\sigma^2}{b_y}$$
$$+ \frac{3I\kappa^2L^2\eta\eta_g}{M}\sum_{m=1}^{M}\mathbb{E}\|\hat{y}_0^{(m)} - y_x^{(m)}\|^2 + \frac{6I^2\kappa^2L^2\eta^3\eta_g}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\|\nabla f^{(m)}(\hat{y}_i^{(m)})\|^2$$

By Algorithm 1, we have $\hat{y}_0^{(m)} = y_{I_{dec}}^{(m)}$, then we have:

$$\mathbb{E}[h(x_{t+1})] \le \mathbb{E}[h(x_t)] - \frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 - \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\Big\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\Big\|^2 + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2$$
$$+ \frac{I(3\kappa^2L^2+\hat{L}^2)\eta\eta_g(1-\mu\gamma)^{I_{dec}}}{M}\sum_{m=1}^{M}\mathbb{E}\Big\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\Big\|^2 + \frac{2I\hat{L}^2\eta\eta_g\gamma\sigma^2}{\mu b_y}$$
$$+ \frac{6I\kappa^2L^2\eta\eta_g\gamma\sigma^2}{\mu b_y} + \frac{6I^2\kappa^2L^2\eta^3\eta_g}{M}\sum_{m=1}^{M}\sum_{i=1}^{I}\|\nabla f^{(m)}(\hat{y}_i^{(m)})\|^2 + \frac{2I^3\kappa^2L^2\eta^3\eta_g\sigma^2}{b_y}$$

Assume that $\|y_0^{(m)} - y_{x^{(m)}}^{(m)}\|^2 \le C_0$ for some constant C_0, and use Assumption 3.2. We sum over $t \in [T]$ to obtain:

$$\frac{1}{T}\sum_{t=1}^{T}\big(\frac{I\eta\eta_g}{2}\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{\eta\eta_g}{4}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\big)$$
$$\le \frac{h(x_1)}{T} + \frac{I\eta^2\eta_g^2\bar{L}G_2^2}{2b_xM} + I\eta\eta_g G_1^2 + I(3\kappa^2L^2+\hat{L}^2)\eta\eta_g(1-\mu\gamma)^{I_{dec}}C_0 + \frac{2I\hat{L}^2\eta\eta_g\gamma\sigma^2}{\mu b_y}$$
$$+ \frac{6I\kappa^2L^2\eta\eta_g\gamma\sigma^2}{\mu b_y} + 6I^3\kappa^2L^2\eta^3\eta_g C_f + \frac{2I^3\kappa^2L^2\eta^3\eta_g\sigma^2}{b_y}$$

Then we divide by $(I\eta\eta_g)/2$ on both sides and have:

$$\frac{1}{T}\sum_{t=1}^{T}\big(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\big)$$
$$\le \frac{2h(x_1)}{TI\eta\eta_g} + \frac{\eta\eta_g\bar{L}G_2^2}{b_xM} + 12I^2\kappa^2L^2\eta^2C_f + \frac{4I^2\kappa^2L^2\eta^2\sigma^2}{b_y}$$
$$+ \frac{4(3\kappa^2L^2+\hat{L}^2)\gamma\sigma^2}{\mu b_y} + 2G_1^2 + 2(3\kappa^2L^2+\hat{L}^2)(1-\mu\gamma)^{I_{dec}}C_0$$

Next, we denote constant $C_\eta = \big(12\kappa^2L^2C_f + \frac{4\kappa^2L^2\sigma^2}{b_y}\big)$ and $C_\gamma = \frac{4(3\kappa^2L^2+\hat{L}^2)\sigma^2}{\mu b_y}$, and set $\eta_g = \frac{1}{2I\bar{L}}$, then we have:

$$\frac{1}{T}\sum_{t=1}^{T}\big(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\big)$$
$$\le \frac{4\bar{L}h(x_1)}{T\eta} + \frac{\eta G_2^2}{2Ib_xM} + C_\eta I^2\eta^2 + C_\gamma\gamma + 2G_1^2 + 2(3\kappa^2L^2+\hat{L}^2)(1-\mu\gamma)^{I_{dec}}C_0$$

Then we choose

$$\eta = \min\left(1, \left(\frac{8Ib_x M\bar{L}h(x_1)}{TG_2^2}\right)^{1/2}, \left(\frac{4\bar{L}h(x_1)}{C_\eta I^2 T}\right)^{1/3}\right),\ \gamma = \min\left(\frac{1}{2L}, \left(\frac{1}{C_\gamma T}\right)^{1/2}\right)$$

Then we obtain:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\right) \\
&\leq \frac{4\bar{L}h(x_1)}{T} + \left(\frac{2G_2^2\bar{L}h(x_1)}{Ib_x MT}\right)^{1/2} + \left(\frac{16I^2 C_\eta \bar{L}^2 h(x_1)^2}{T^2}\right)^{1/3} \\
&\quad + \left(\frac{C_\gamma}{T}\right)^{1/2} + 2G_1^2 + 2(3\kappa^2 L^2 + \hat{L}^2)(1-\mu\gamma)^{I_{dec}} C_0
\end{aligned}$$

Finally, since $\hat{L} = O(\kappa^2)$, $\bar{L} = O(\kappa^3)$ and $\mu\gamma = O(\kappa^{-1})$. Suppose we choose $I = O(1)$, then $C_\eta = O(\kappa^2)$, $C_\gamma = O(\kappa^5)$, and use $G_1 = \kappa(1-\tau\mu)^{Q+1}C_f$ in Proposition C.3, we have:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\|\nabla h(x_t)\|^2 + \frac{1}{2I}\sum_{i=1}^{I}\mathbb{E}\|\mathbb{E}_\xi[\bar{\Delta}\hat{x}_{t,i}]\|^2\right) \\
&= O\left(\frac{\kappa^3}{T} + \left(\frac{\kappa^5}{T}\right)^{1/2} + \left(\frac{\kappa^6}{T^2}\right)^{1/3} + \kappa^2(1-\tau\mu)^{2(Q+1)} + \kappa^4(1-\mu\gamma)^{I_{dec}}\right)
\end{aligned}$$

and to reach an ϵ stationary point, we choose $Q = O(\kappa\log(\frac{\kappa}{\epsilon}))$, $I_{dec} = O(\kappa\log(\frac{\kappa}{\epsilon}))$ and $T = O(\kappa^5\epsilon^{-2})$ number of iterations. □