PRIVACY TRADEOFFS IN VERTICAL FEDERATED LEARNING

Linh Tran [1] **Timothy Castiglia** [1] **Stacy Patterson** [1] **Ana Milanova** [1]

ABSTRACT

We present VFL-PBM, a communication-efficient Vertical Federated Learning algorithm with Differential Privacy guarantees. VFL-PBM combines Secure Multi-Party Computation with the recently introduced Poisson Binomial Mechanism to protect parties' private datasets during model training. We analyze the end-to-end privacy and convergence behavior of our algorithm, and we provide the first theoretical characterization of the relationship between privacy, convergence error, and communication cost in differentially-private VFL. Our experiments show the VFL model performs well, with negligible decline in accuracy as we increase the privacy parameters.

1 INTRODUCTION

Federated Learning (FL) (McMahan et al., 2017) is a machine learning technique where training data is distributed across multiple parties, and the goal is to collaboratively train a global model. The parties execute the training algorithm, facilitated by a central server, without directly sharing the private data with each other or the server. FL has been used in various applications such as drug discovery, mobile keyboard prediction, and ranking browser history suggestion (Aledhari et al., 2020).

The majority of FL algorithms support Horizontal Federated Learning (HFL) (e.g., (Yang et al., 2019)), where the datasets of the parties are distributed horizontally, i.e, all parties share the same features, but each holds a different set of data samples. In contrast, Vertical Federated Learning (VFL) targets the case where all parties share the same set of data sample IDs but each has a different set of features (e.g., (Hu et al., 2019; Gu et al., 2021)). An example of VFL includes a bank, a hospital, and an insurance company that wish to train a model predicting customer credit scores. The three institutions have a common set of customers, but the bank holds information about customers' transactions, while the hospital knows about medical records, and the insurance company has data about customers' accident reports. Such a scenario must employ VFL to train models on private vertically distributed data.

*Equal contribution [1]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA. Correspondence to: Linh Tran <tranl3@rpi.edu>.

Federated Learning Systems (FLSys) workshop, in Conjunction with the 6^{th} MLSys Conference, Miami, Florida, USA, 2023.

While FL is designed to address privacy concerns by keeping data decentralized and on the premises where it is generated, there is possible information leakage from the messages exchanged during training (Geiping et al., 2020; Mahendran & Vedaldi, 2015). Thus, it is important to develop methods for FL that have provable privacy guarantees. A common approach for privacy-preservation is *Differential Privacy* (DP) (Dwork & Roth, 2014), in which the private data is protected by adding noise at various stages in the training algorithm. Through careful application of DP, one can protect the data, not only in a single computation, but throughout the execution of the training algorithm. A number of works have studied the end-to-end privacy of DP-based HFL algorithms (e.g., (Truex et al., 2019; Wei et al., 2020; Truex et al., 2020)). However, there is limited prior work on privacy analysis in VFL, and none that addresses the interplay between the convergence of the training algorithm, the end-to-end privacy, and the communication cost.

We propose PBM-VFL, a new VFL algorithm that combines the Poisson Binomial Mechanism (PBM) (Chen et al., 2022) with Secure Multi-Party Computation (MPC) to provide DP over the private training datasets. In PBM-VFL, as in other VFL algorithms, each party trains a local network that transforms their raw features into embeddings. The server trains a fusion model that produces the predicted label from these embeddings. To protect data privacy, the parties quantize their embeddings into differentially private integer vectors using PBM. The parties then apply MPC over the integer values so that the server aggregates the quantized sum without learning anything else. The server then estimates the embedding sum based on the quantized sum and uses the estimated value to calculate the loss and the gradients needed for training.

We analyze the end-to-end privacy and the convergence

behavior of PBM-VFL and relate this analysis to the communication cost. Through these analyses, we provide the first theoretical characterization of the tradeoffs between privacy, convergence error, and communication cost in VFL. We note that VFL presents a different privacy problem than HFL. In HFL, parties share an aggregated gradient over a minibatch with the server, whereas in VFL, the server learns the sum of the party embeddings *for each sample* in a minibatch individually. This difference requires a different privacy analysis. Further, the impact of DP noise enters via the gradient computation in HFL, whereas it enters via an argument to the loss function in VFL. This change necessitates different convergence analysis.

We summarize our main contributions in this work.

1. We introduce PBM-VFL, a communication efficient and private VFL algorithm.
2. We provide analysis of the overall privacy budget.
3. We prove the convergence bounds of PBM-VFL and give the relationship between the privacy budget and communication cost.
4. We evaluate our algorithm by training on ModelNet-10 and CIFAR-10 datasets. Our results show that the VFL model performs well with negligible decline in accuracy as we increase the privacy parameters.

Related work. Multiple works apply DP in HFL tasks to protect the data or the model (Truex et al., 2019; Wei et al., 2020; Truex et al., 2020; Agarwal et al., 2018; Chen et al., 2022). As discussed above, there are significant differences in applying DP to VFL. We note that, (Agarwal et al., 2018) and (Chen et al., 2022) both utilize quantization and MPC to protect the gradient computations. While PBM-VFL utilizes the same mechanisms, there are significant differences in their application and analysis in VFL.

There are previous papers that provide different private methods for VFL. (Lu & Ding, 2020) provide an MPC protocol for VFL, but they do not use Differential Privacy, and thus information may leak from the aggregated embeddings. (Ranbaduge & Ding, 2022) applies a Gaussian DP mechanism on a general VFL algorithm, but their privacy analysis is incomplete. (Li et al., 2023) uses a Laplace DP mechanism for a VFL k-means algorithm, but their algorithm does not extend to neural networks. (Xu et al., 2021) proposes a secure and communication-efficient framework for VFL using Functional Encryption for secure gradient computation, but they do not consider privacy for end-to-end training.

Outline. The rest of the paper is organized as follows. In Section 2, we briefly overview the basic principle of Differential Privacy, introduce the scalar Poisson Binomial Mechanism, and recall the definition of Multi-Party Computation. We provide the system model, problem formulation with our assumptions, and threat model in Section 3. Section 4 presents the detail of our algorithm, and Section 5 presents the analysis. Our experimental results are given in Section 6, and we conclude in Section 7.

2 BACKGROUND

In this section, we present background on Differential Privacy and the related building blocks we use in our privacy-preserving VFL algorithm.

2.1 Differential Privacy

Differential privacy provides a strong privacy guarantee that ensures that an individual's sensitive information, e.g., the training data, remains private even if the adversary has access to auxiliary information. A standard notion of Differential Privacy is (ϵ, δ)-DP, which is defined as follows (Dwork & Roth, 2014).

Definition 2.1. A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP if for any two adjacent inputs $D, D' \in \mathcal{D}$ and for any subset of outputs $\mathcal{S} \in \mathcal{R}$ it holds that

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta$$

In this work, we employ a related version of DP called Rényi Differential Privacy (RDP) (Mironov, 2017), which is based on the concept of the Rényi divergence. The Rényi divergence and RDP are defined as follows.

Definition 2.2. For two probability distributions P and Q defined over $\mathcal{R}$, the Rényi divergence of order $\alpha > 1$ is

$$D_\alpha(P, Q) \coloneqq \frac{1}{\alpha - 1} \log \left(\mathbb{E}_{x \sim Q} \left(\frac{P(x)}{Q(x)} \right)^\alpha \right).$$

Definition 2.3. A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (α, ϵ)-RDP if for any two adjacent inputs $D, D' \in \mathcal{D}$, it holds that

$$D_\alpha(P_{\mathcal{M}(D)}, P_{\mathcal{M}(D')}) \leq \epsilon$$

We utilize RDP rather than (ϵ, δ)-DP because it facilitates calculation of the cumulative privacy loss over the sequence training iterations in the algorithm execution. The notions of DP and RDP are closely related as shown in the following lemma.

Lemma 2.4. *(Lemma B.2 (Chen et al., 2022)) If $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP for all $\alpha > 1$, then for any $\delta > 0$, $\mathcal{M}$ satisfies $(\epsilon_{DP}(\delta), \delta)$-DP where*

$$\epsilon_{DP} = \Theta \left(\sqrt{\sup_\alpha \frac{\epsilon(\alpha))}{\alpha} \log(1/\delta)} \right).$$

Algorithm 1 Scalar Poisson Binomial Mechanism

Input: $x_i \in [-C, C], \beta \in [0, \frac{1}{4}], b \in \mathbb{N}$
$p_i \leftarrow \frac{1}{2} + \frac{\beta}{C} x_i$
$q_i \leftarrow \text{Binom}(b, p_i)$
Output: Quantized value q_i

2.2 Poisson Binomial Mechanism

A key component of our algorithm (see Section 4) is to protect the privacy of inputs to a distributed sum computation. To achieve this, we rely on the combination of RDP and MPC. We use the Poisson Binomial Mechanism (PBM) developed in (Chen et al., 2022) to provide RDP. Unlike other HFL private methods, PBM utilizes RDP guarantee, and uses the Binomial distribution to generate scalar quantized values which is suitable for integer-based MPC.

We sketch the process for computing a sum of scalar values. Suppose each participant m, $m = 1, \ldots, M$ has an input $x_m \in [-C, C]$, and the goal is to estimate the sum $s = \sum_{m=1}^{M} x_m$ while protecting the privacy of the inputs. Each participant first quantizes its value according to Algorithm 1. The values of $\beta \in [0, \frac{1}{4}]$ and $b \in \mathbb{N}$ are chosen to achieve a desired RDP and accuracy tradeoff. The participants use MPC to find the sum $\hat{q} = \sum_{m=1}^{M} q_m$. An estimation value of s is then computed from $\hat{q}$ as $\tilde{s} = \frac{C}{\beta b}(\hat{q} - \frac{bM}{2})$. We provide the theorem for this sum computation, which is a slight modification from the result in (Chen et al., 2022).

Theorem 2.5. *((Chen et al., 2022)) Let $x_m \in [-C, C]$, $m = 1, \ldots, M$, $\beta \in [0, \frac{1}{4}]$, and $b \in \mathbb{N}$. Then, the sum computation:*

1. *satisfies $(\alpha, \epsilon(\alpha))$-RDP for $\alpha > 1$ and $\epsilon(\alpha) = \Omega(b\beta^2\alpha/M)$*
2. *yields an unbiased estimate of s with variance $\frac{C^2 M}{4\beta^2 b}$.*

2.3 Multi-Party Computation

While the PBM can be used to provide RDP for a sum, we must also ensure that the inputs to the sum are not leaked during the computation. This is achieved via MPC. MPC is a cryptographic mechanism that allows a set of parties to jointly compute a function over their secret inputs, so that only the function output is revealed at the end of the protocol.

There are a variety of MPC methods that can be used for sum computation. For the purposes of our communication cost analysis, we fix an MPC protocol, specifically Protocol 0 for Secure Aggregation from (Bonawitz et al., 2016). It considers M parties, each holding an integer secret value $q_m \in [0, b)$ and an honest-but-curious server. The goal is to compute $\sum q_m$ collaboratively so that the server learns the sum and nothing else, while the parties learn nothing. [1]

In Protocol 0, each pair of parties m_1, m_2 samples two random integers in $[0, b)$, u_{m_1,m_2} and u_{m_2,m_1} using a pseudo-random number generator with a seed known to only parties m_1 and m_2. Each party then computes $M - 1$ perturbations $p_{m,m'} = u_{m,m'} - u_{m',m}$ and masks its secret value by computing $y_m = q_m + \sum_{m'=1}^{M} p_{m,m'}$ ($p_{m,m} = 0$). It then sends y_m to the server. The server sums all y_m: $S = \sum_{m=1}^{M} y_m + \sum_{m=1}^{M} \sum_{m'=1}^{M} p_{m,m'}$. One can see that S computes $\sum_{m=1}^{M} q_m$ because $p_{m_1,m_2} = u_{m_1,m_2} - u_{m_2,m_1}$ and $p_{m_2,m_1} = u_{m_2,m_1} - u_{m_1,m_2}$ cancel each other. At the same time, the protocol is perfectly secure, revealing nothing about the individual q_m's to the server.

To analyze communication cost, observe that each party incurs cost by sending y_m. Since $-(M-1)b < y_m < Mb$, $O(\log\ Mb) = O(\log\ M + \log\ b)$ bits suffice to represent the range of negative and positive values of y_m. Thus, we need $O(\log\ M + \log\ b)$ per-party bits to send the masked value.

3 Problem Formulation

3.1 System Model

We consider a system consisting of M parties and a server. There is a dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$ partitioned across the M parties, where N is the number of data samples and D is the number of features. Let $\mathbf{x}^i$ denote the ith sample of $\mathbf{X}$. For each sample $\mathbf{x}^i$, each party m holds a disjoint subset, i.e., a vertical partition, of the features. We denote this subset by $\mathbf{x}^i_m$, and note that $\mathbf{x}^i = [\mathbf{x}^i_1, \ldots, \mathbf{x}^i_M]$. The entire vertical partition of $\mathbf{X}$ that is held by party m is denoted by $\mathbf{X}_m$, with $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_M]$.

Let y_i be the label for sample $\mathbf{x}_i$, and let $\mathbf{y} \in \mathbb{R}^{N \times 1}$ denote the set of all labels. We assume that the labels are stored at the server. As it is standard, the dataset is aligned for all parties in a privacy-preserving manner as a pre-processing step. This can be done using Private Entity Resolution (Xu et al., 2021).

3.2 Training Problem

The goal is to train a global model using the data from all parties and the labels from the server. Each party m trains a local network h_m with parameters $\boldsymbol{\theta}_m$ that takes the vertical partition of a sample $\mathbf{x}$ as input and produces an embedding

[1] We note that (Bonawitz et al., 2016) and (Bonawitz et al., 2017) present more robust and more secure versions of Protocol 0. They add an $O(M)$ term to the communication of each party to maintain additional robustness and security. We fix Protocol 0 for simplicity and because its cost is precisely the cost of data transfer influenced by our parameter b; Bonawitz et al. (Bonawitz et al., 2017) show that data transfer dominates communication.

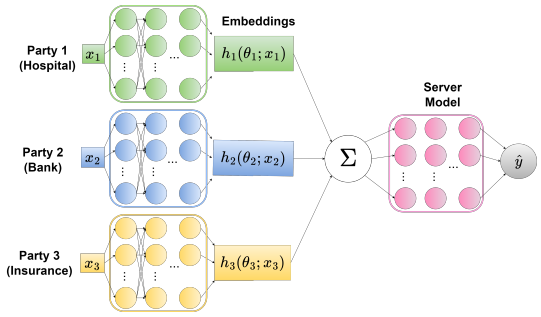


Figure 1. Example global model with neural networks.

of dimension P. The server trains a fusion model h_0 with parameters $\boldsymbol{\theta}_0$ that takes a sum of the embeddings for a sample as input and produces a predicted label $\hat{y}$. The global model $f(\cdot)$ has the form

$$f(\mathbf{x}; \Theta) := h_0 \left(\sum_{m=1}^{M} h_m(\boldsymbol{\theta}_m; \mathbf{x}_m);\ \boldsymbol{\theta}_0 \right) \tag{1}$$

where Θ denotes the set of all model parameters. An example of the global model architecture is shown in Figure 1.

To train this model, the parties and the server collaborate to minimize a loss function of the form

$$\mathcal{L}(\Theta; \mathbf{X}, \mathbf{y}) := \frac{1}{N} \sum_{i=1}^{N} \ell \left(\boldsymbol{\theta}_0, \hat{h}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M; \mathbf{x}^i); y^i \right) \tag{2}$$

where $\ell(\cdot)$ is the loss for a single sample $(\mathbf{x}^i, y^i)$, and

$$\hat{h}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M; \mathbf{x}^i) = \sum_{m=1}^{M} h_m(\boldsymbol{\theta}_m; \mathbf{x}_m^i). \tag{3}$$

We sometimes omit $\mathbf{X}$ and $\mathbf{y}$ from the notation when the context is clear.

Let $\mathcal{B} := (\mathbf{X}^{\mathcal{B}}, \mathbf{y}^{\mathcal{B}})$ be a randomly sampled mini-batch of B samples. We denote the stochastic partial derivative of $\mathcal{L}$ over $\mathcal{B}$ with respect to $\boldsymbol{\theta}_m$, $m = 0, \ldots, M$, by

$$\nabla_m \mathcal{L}_{\mathcal{B}}(\Theta) := \frac{1}{B} \sum_{(\mathbf{x}^i, y^i) \in \mathcal{B}} \nabla_{\boldsymbol{\theta}_m} \ell(\boldsymbol{\theta}_0, \hat{h}(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M; \mathbf{x}^i); y^i).$$

The partial derivatives of $\mathcal{L}$ and $\mathcal{L}_{\mathcal{B}}$ with respect to $\hat{h}$ are denoted by $\nabla_{\hat{h}} \mathcal{L}$ and $\nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}}$, respectively.

We make the following assumptions.

Assumption 3.1. Smoothness:

1. There exists positive constant $L < \infty$ such that for all Θ_1, Θ_2:

$$\|\nabla \mathcal{L}(\Theta_1) - \nabla \mathcal{L}(\Theta_2)\| \leq L \|\Theta_1 - \Theta_2\| \tag{4}$$

2. There exist positive constants $L_0 < \infty$ and $L_{\hat{h}} < \infty$ such that for all server parameters $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_0'$ and all embedding sums $\boldsymbol{h}$ and $\boldsymbol{h}'$:

$$\|\nabla_{\boldsymbol{\theta}_0} \ell(\boldsymbol{\theta}_0, \boldsymbol{h}) - \nabla_{\boldsymbol{\theta}_0} \ell(\boldsymbol{\theta}_0', \boldsymbol{h}')\| \leq L_0 \|[\boldsymbol{\theta}_0^T, \boldsymbol{h}^T]^T - [\boldsymbol{\theta}_0'^T, \boldsymbol{h}'^T]^T\| \tag{5}$$

$$\|\nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0, \boldsymbol{h}) - \nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0', \boldsymbol{h}')\| \leq L_{\hat{h}} \|[\boldsymbol{\theta}_0^T, \boldsymbol{h}^T]^T - [\boldsymbol{\theta}_0'^T, \boldsymbol{h}'^T]^T\|. \tag{6}$$

Assumption 3.2. Unbiased gradients: For every mini-batch $\mathcal{B}$, the stochastic gradient is unbiased:

$$\mathbb{E}_{\mathcal{B}} \nabla \mathcal{L}_{\mathcal{B}}(\Theta) = \nabla \mathcal{L}(\Theta).$$

Assumption 3.3. Bounded variance: There exists positive constant $\sigma < \infty$ such that for every mini-batch $\mathcal{B}$ (with $|\mathcal{B}| = B$)

$$\mathbb{E}_{\mathcal{B}} \|\nabla \mathcal{L}(\Theta) - \nabla \mathcal{L}_{\mathcal{B}}(\Theta)\|^2 \leq \frac{\sigma^2}{B}. \tag{7}$$

Assumption 3.4. Bounded embeddings: There exists positive constant $C < \infty$ such that for $m = 1, \ldots, M$, for all $\boldsymbol{\theta}_m$ and $\mathbf{x}_m$, $\|h_m(\boldsymbol{\theta}_m; \mathbf{x}_m)\|_\infty \leq C$.

Assumption 3.5. Bounded embedding gradients: There exists positive constants $H_m < \infty$ for $m = 1, \ldots, M$ such that for all $\boldsymbol{\theta}_m$ and all samples i, the embedding gradients are bounded as

$$\|\nabla_m h_m(\boldsymbol{\theta}_m; \mathbf{x}_m^i)\|_{\mathcal{F}} \leq H_m \tag{8}$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm.

Part 1 of Assumption 3.1, Assumption 3.2, and Assumption 3.3 are standard in the analysis of gradient-based algorithms (e.g., Nguyen et al. (2018); Bottou et al. (2018)). Part 2 of Assumption 3.1 bounds the rate of change of the partial derivative of ℓ with respect to each of its two arguments. This assumption is needed to ensure convergence over the noisy embedding sums. Assumption 3.4 bounds the individual components of the embeddings. This can be achieved via a standard activation function such as sigmoid or $tanh$. Assumption 3.5 bounds the partial derivatives of the embeddings with respect to a single sample. This bound is also necessary to analyze the impact of the DP noise on the algorithm convergence.

3.3 Threat Model

Our privacy goal is to protect the training data $\mathbf{X}_m$ of each party m from the other parties and the server. We assume that all parties and the server are honest-but-curious. They correctly follow the training algorithm, but they can try to infer the data of other parties from information exchanged in the algorithm. We assume that the parties do not collude and that communication occurs through robust and

Algorithm 2 Privacy-Preserving VFL

1: **Initialize:** $\Theta^0 = [\boldsymbol{\theta}_0^0, \boldsymbol{\theta}_1^0, \ldots, \boldsymbol{\theta}_M^0]$
2: **for** $t \leftarrow 0, \ldots, T-1$ **do**
3: Randomly sample $\mathcal{B}^t$ from $(\mathbf{X}, \mathbf{y})$
4: **for** party $m \leftarrow 1, \ldots, M$ in parallel **do**
5: /* Generate quantized embeddings for $\mathcal{B}^t$ */
6: $q_m^t \leftarrow \textbf{PBM}(h_m(\boldsymbol{\theta}_m^t; \mathbf{X}_m^{\mathcal{B}^t}), b, \beta)$
7: **end for**
8: /* At server */
9: $\hat{q}^t \leftarrow \sum_{m=1}^{M} q_m^t$ via **MPC**
10: $\tilde{h}^t \leftarrow \frac{C}{\beta b}(\hat{q}^t - \frac{bM}{2})$
11: Server sends $\nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$ to all parties
12: /* server updates its parameters */
13: $\boldsymbol{\theta}_0^{t+1} \leftarrow \boldsymbol{\theta}_0^t - \eta \nabla_0 \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$
14: **for** $m \leftarrow 1, \ldots, M$ in parallel **do**
15: /* party m updates its parameters */
16: $\nabla_m \mathcal{L}_{\mathcal{B}}(\Theta^t) \leftarrow$
17: $\nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{X}_m^t) \nabla_{h_m} \tilde{h}^t \nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$
18: $\boldsymbol{\theta}_m^{t+1} \leftarrow \boldsymbol{\theta}_m^t - \eta^t \nabla_m \mathcal{L}_{\mathcal{B}}(\Theta^t)$
19: **end for**
20: **end for**

secure channels that ensure no party drops out and no "man-in-the-middle" attacks occur. (One can easily extend our system with the more secure and robust MPC protocols from (Bonawitz et al., 2017)).

4 ALGORITHM

We now present PBM-VFL. Pseudocode is given in Algorithm 2.

Each party m and the server initialize their local parameters $\boldsymbol{\theta}_m$, $m = 0, \ldots, M$ (line 1). The algorithm runs for T iterations. In each iteration, a minibatch $\mathcal{B}^t$ is chosen at random from $\mathbf{X}$. Each party m generates an embedding $h_m(\boldsymbol{\theta}_m^t; \mathbf{x}^i)$ for each sample i in the minibatch. We denote the set of party m's embeddings for the minibatch by $h_m(\boldsymbol{\theta}_m^t; \mathbf{X}_m^{\mathcal{B}^t})$. Each party m computes the set of noisy quantized embeddings q_m^t using PBM component-wise on each embedding (lines 3-7). The benefit of PBM here is that it quantizes the embedding components into $\log_2 b$-bit integer vectors that are suitable for the MPC protocol. The quantization also reduces the communication cost for the sum computation.

To complete forward propagation, the server needs an estimate of the sum of the embeddings from each party for each sample in $\mathcal{B}^t$. The parties and the server execute MPC (e.g., Protocol 0), which reveals $\hat{q}^i$ for each sample $i \in B^t$ to the server. For each $i \in \mathcal{B}^t$ the server estimates the embedding sum as $\tilde{h}^i = \frac{C}{\beta b}(\hat{q}^i - \frac{bM}{2})$ (lines 9-10). We let $\tilde{h}^t$ denote the set of noisy embedding sums.

The server calculates the gradient of $\mathcal{L}_{\mathcal{B}}$ with respect to $\hat{h}$ for the minibatch using $\tilde{h}^t$, denoted $\nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$ and sends this information to the parties so they can complete their parameter updates (line 11). Then the server calculates the stochastic gradient of $\mathcal{L}$ with respect to its own parameters, denoted $\nabla_0 \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$, and uses this gradient to update its own model parameters with learning rate η (line 13). Finally, after each party receives the partial derivative from the server, it computes the partial derivative of $\mathcal{L}_{\mathcal{B}^t}$ with respect to its local model parameters using the chain rule as:

$$\nabla_m \mathcal{L}_{\mathcal{B}^t}(\Theta^t) = \nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{X}_m^{\mathcal{B}^t}) \nabla_{h_m} \tilde{h}^t \nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \tilde{h}^t).$$

Note that $\nabla_{h_m} \tilde{h}^t$ is the identity operator. The party then updates its local parameters (lines 16-18) using this partial derivative, with learning rate η.

Information Sharing There are two places in Algorithm 2 where information about $\mathbf{X}$ is shared. The first is when the server learns the sum of the embeddings for each sample in a minibatch (line 9). We protect the inputs to this computation via PBM and MPC. The second is when the server sends $\nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$ to each party. By the post-processing property of DP, this gradient retains the same privacy protection as the sum computation. We give a formal analysis of the algorithm privacy in Section 5.

Communication Cost We now discuss the communication cost of Algorithm 2. Each party sends its masked quantized embedding at a cost of $O(P(log\ M + log\ b))$ bits, as detailed in 2 and the cumulative cost for M parties and mini-batch of size B becomes $O(BMP(log\ M + log\ b))$. In the back propagation, the server sends the partial derivatives without quantization to each party, which is the most costly message exchanging step. Nevertheless, we save a significant number of bits when the parties send their masked quantized embedding to the server. Let F be the number of bits to represent a floating point number. Then the cost of sending these partial derivatives to M parties is $O(BMPF)$. The total communication cost for Algorithm 2 is $O(TBMP(log\ M + \log\ b + F))$.

5 ANALYSIS

We now present our theoretical results with respect to the privacy and convergence of PBM-VFL, and we provide a discussion of the tradeoffs between them.

5.1 Privacy

Our method makes use of DP that aims to provide a measure of indistinguishability between adjacent datasets consisting of multiple data samples. We say that two datasets are adjacent if they differ by a single party's feature set for a

single sample. We give an accounting of the privacy budget across T iterations of Algorithm 2.

Theorem 5.1. *Let Assumption 3.4 hold, and assume $\beta \in [0, \frac{1}{4}]$ and $b \in \mathbb{N}$. Algorithm 2 after T iterations satisfies $(\alpha, \epsilon_{final}(\alpha))$-RDP where $\alpha > 1$ and $\epsilon_{final}(\alpha) = O(\frac{TBb\beta^2\alpha}{MN})$.*

Proof. The proof is an adaptation of the steps in (Chen et al., 2022). To account for the privacy budget $\epsilon_{final}(\alpha)$, we begin by applying Theorem 2.5. It states that, for a sample i, the computation of the sum $\tilde{h}_i^t$ is $(\alpha, \epsilon(\alpha))$-RDP for any $\alpha > 1$ and $\epsilon(\alpha) = \Omega(b\beta^2\alpha/M)$. To compute $\nabla_{\hat{h}} \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \tilde{h}^t)$, the server applies a deterministic function on $\tilde{h}$. By standard post-processing arguments, it follows that this computation provides $(\alpha, \epsilon(\alpha))$-RDP with the same $\epsilon(\alpha)$.

We now consider privacy loss across all T iterations. At each iteration, the algorithm processes a sample at a rate B/N, leading to an expected TB/N number of times that each sample is used over T iterations. Accounting for privacy loss across all T iterations leads to $\epsilon_{final}(\alpha) = O(\frac{TBb\beta^2\alpha}{MN})$. □

5.2 Convergence

We next present our theoretical result on the convergence of Algorithm 2. The proof is provided in Appendix A.

Theorem 5.2. ***Convergence:*** *Under Assumptions 3.1-3.5, if $\eta < \frac{1}{2L}$, then the average squared gradient over T iterations of Algorithm 2 satisfies:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left(\|\nabla\mathcal{L}(\Theta^t)\|^2\right) \leq \frac{2(\mathcal{L}(\Theta^0) - \mathbb{E}(\mathcal{L}(\Theta^T))}{\eta T} + 2L\eta\frac{\sigma^2}{B} + (1+2L\eta)\left(\frac{C^2MP(L_0^2 + L_{\hat{h}}^2\sum_{m=1}^{M}H_m^2)}{4\beta^2 b}\right). \quad (9)$$

The first term in the bound in (9) is determined by the difference between the loss of the initial model and the model after T training iterations. This term vanishes as T goes to infinity. The second term is the convergence error associated with variance of the stochastic gradients and the Lipschitz constant L. The third term is the convergence error arises from the DP noise in the sums of the embeddings. This error depends on the inverse of b, the number of embedding quantization bins, and the inverse square of β, which controls the degree of privacy. As b or β increases, this error decreases.

Remark 5.3. **Asymptotic convergence** If $\eta = \frac{1}{\sqrt{T}}$ and B is independent of T then

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla\mathcal{L}(\Theta^t)\|^2 = O(\sqrt{T} + \mathcal{E}) \quad (10)$$

where $\mathcal{E} = O(\frac{1}{\beta^2 b})$ is the error due to the PBM. We note that if the embedding sums are computed exactly, giving up privacy, the algorithm reduces to standard SGD; the third term in (9) becomes 0, giving a convergence rate of $O(\frac{1}{\sqrt{T}})$.

5.3 Tradeoffs

We observe that there is a connection between the algorithm privacy, communication cost, and convergence behavior. Let us consider a fixed value for the privacy parameter β. We can reduce the convergence error in Theorem 5.2 by increasing b, but this results in less privacy guarantee. Higher b also enlarges the algorithm communication cost, but so long as $\log b < F$, this increase is negligible.

Similarly, if we fix the communication cost of the algorithm over T iterations, we can increase the privacy by decreasing β. This, in turn, leads to an increase in the convergence error on the order of $\frac{1}{\beta^2}$.

The number of parties M also affects the privacy budget and convergence error. With larger M, each party gets more protection for their data but with larger convergence error.

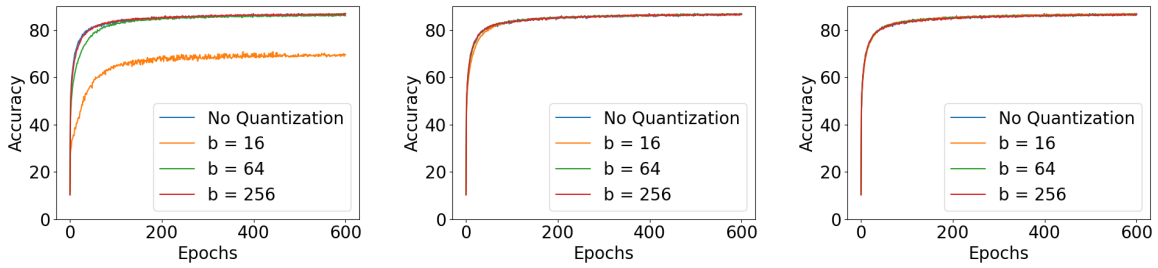
6 Experiments

We present experiments to examine the privacy and accuracy tradeoff of Algorithm 2. We study how different values of the PBM parameters b and β affect the performance of the model as well as the RDP privacy budget. We conducted our experiments on two datasets ModelNet-10 and CIFAR-10.

ModelNet-10: ModelNet-10 is a set of images generated from CAD models, where each model produces images for 12 different angles, with 10 different object classes for image classification. We performed experiments with 6 and 12 parties, where each party holds 2 or 1 view(s) of the model, respectively. Each party's local model is a neural network with two convolutional layers and a fully-connected layer, and the server model consists of a fully-connected layer that uses a cross-entropy loss. We use a fixed batch size of 64 with learning rate of 0.01 and train for 250 epochs.

CIFAR-10: CIFAR-10 is another image dataset with 10 object classes for classification task. We train the model with 4 parties, each holds a different quadrant (16×16) of the images. Each party uses a ResNet18 neural network model, and the server model consists of a fully-connected layer with a cross-entropy loss. We use a fixed batch size of 100 with learning rate of 0.01 and train for 600 epochs.

We consider different PBM parameters for the two datasets



(a) $\beta = 0.05$ (b) $\beta = 0.15$ (c) $\beta = 0.25$

Figure 2. The accuracy of CIFAR-10 by epochs for various values of b and β.

to show the privacy-accuracy tradeoff. We choose the same set of $\beta \in \{0.05, 0.15, 0.25\}$ for both datasets, and we pick $b \in \{4, 16, 64\}$ for ModelNet-10 and $b \in \{16, 64, 256\}$ for CIFAR-10. The nature of the CIFAR-10 dataset permits less noise in the raw data and typically requires a smaller learning rate for convergence. To bound the embedding values into the range $[-C, C]$ as required for Algorithm 1, we use the $tanh$ activation function to scale the embedding values into the range $[-1, 1]$ for each party model of both datasets.

Figure 2 shows the results of experiments for the CIFAR-10 dataset with three different values of $\beta = \{0.05, 0.15, 0.25\}$. Each figure shows how b affects the test accuracy. We get a moderate decrease in accuracy with $b = 16$ and $\beta = 0.05$ in Figure 2a. Nevertheless, with the same β value, $b = 64$ and $b = 256$ yield almost the same accuracy as the case without any quantization. There is also an insignificant loss in the test accuracy for $\beta = 0.15$ and $\beta = 0.25$ with different b values, as illustrated in Figures 2b and 2c.

As discussed in the previous section, the privacy bound of PBM-VFL is $O(\frac{TBb\beta^2\alpha}{MN})$, which indicates that we get more privacy by reducing the value of b and β. There is tradeoff between privacy and accuracy, and smaller b and β lead to less accuracy in the model performance.

The result of the experiments for the ModelNet-10 dataset with 6 parties and 12 parties are shown in Figures 3 and 4, respectively. In these figures, we fix the value for $b = \{4, 16, 64\}$ to illustrate the influence of $\beta = \{0.05, 0.15, 0.25\}$ on the model performance. The privacy-accuracy tradeoff is more prevalent in the ModelNet-10 experiments. With 6 parties, the model performs relatively well even when $b = 4$ (Figure 3a). There is no significant difference between different β values for $b = 4$, however we do see a clear increase in accuracy for higher β values in Figures 3b and 3c.

The results for the ModelNet-10 dataset with 12 parties in Figure 4 show a similar trend as the graphs for 6 parties. The accuracy for different values of b are very close to the case without any quantization and DP noise. In addition, there is a growth in the model performance when β increases, specially in Figure 4b and 4c. We also see a slight decline in the test accuracy when the number of parties increases from 6 to 12 in the ModelNet-10 tests (Figure 3a vs. Figure 4a, etc). However, this drop is negligible, and the model still performs well with larger number of parties.

In conclusion, our results support the privacy-accuracy tradeoff summarized in Section 5. Notably, even higher-privacy values of b, such as $b = 16$ achieve high accuracy, which entails fewer bits of communication per round.

7 Conclusion

We presented PBM-VFL, a privacy-preserving and communication-efficient algorithm for training models with vertically partitioned data. We analyzed privacy and convergence behavior and proved an end-to-end privacy bound as well as a convergence bound. In future work, we seek to develop new analytical techniques to achieve a tighter bound on the privacy budget for specific model architectures and loss functions, and to design new methods to improve the privacy/accuracy tradeoff in VFL.

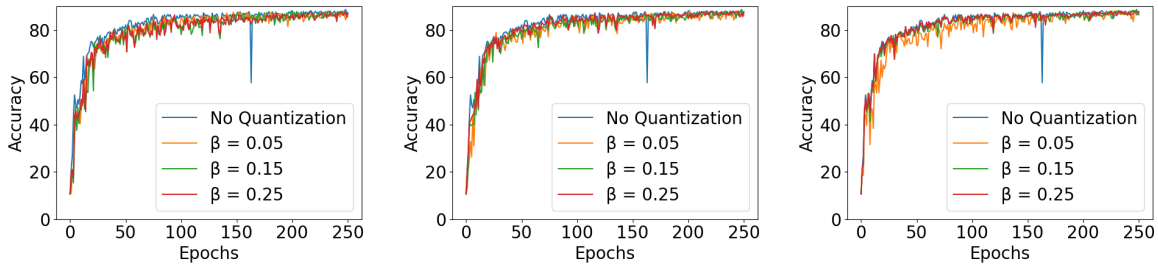
Acknowledgements

This work was supported by NSF grants CNS-1814898 and CNS-1553340, and by the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network.

References

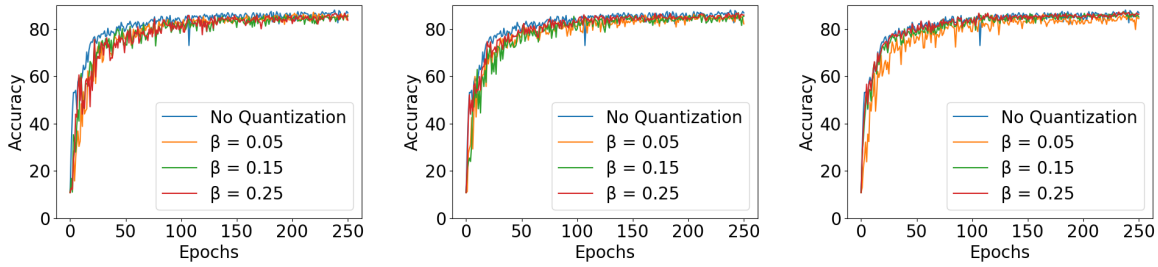
Agarwal, N., Suresh, A. T., Yu, F., Kumar, S., and Mcmahan, H. B. cpSGD: Communication-efficient and differentially-private distributed SGD, 2018.

Aledhari, M., Razzak, R., Parizi, R. M., and Saeed, F. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725,



(a) $b = 4$ (b) $b = 16$ (c) $b = 64$

Figure 3. The accuracy of ModelNet-10 with 6 parties by epochs for various values of b and β.



(a) $b = 4$ (b) $b = 16$ (c) $b = 64$

Figure 4. The accuracy of ModelNet-10 with 12 parties by epochs for various values of b and β.

2020.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for federated learning on user-held data. *arXiv prepring arXiv:1611.04482*, 2016.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *Proc. ACM SIGSAC Conf. Computer and Communications Security*, pp. 1175–1191, 2017.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

Chen, W., Özgür, A., and Kairouz, P. The poisson binomial mechanism for unbiased federated learning with secure aggregation. In *Proc. Int. Conf. Machine Learning*, pp. 3490–3506, 2022.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, Aug 2014.

Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. Inverting gradients - how easy is it to break privacy in federated learning? *Adv. Neural Inf. Process. Syst.*, 2020.

Gu, B., Xu, A., Huo, Z., Deng, C., and Huang, H. Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. *IEEE Trans. on Neural Netw. Learn. Syst.*, pp. 1–13, 2021.

Hu, Y., Niu, D., Yang, J., and Zhou, S. FDML: A collaborative machine learning framework for distributed features. *Proc. ACM Int. Conf. Knowl. Discov. Data Min.*, pp. 2232–2240, 2019.

Li, Z., Wang, T., and Li, N. Differentially private vertical federated clustering. *Proc. VLDB Endow.*, 16(6): 1277–1290, Apr 2023.

Lu, L. and Ding, N. Multi-party private set intersection in vertical federated learning. In *Proc. IEEE 19th Int. Conf. Trust, Security and Privacy in Computing and Communications*, pp. 707–714, 2020.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 5188–5196, 2015.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep

networks from decentralized data. *Proc. 20th Int. Conf. on Artif. Intell.*, pp. 1273–1282, 2017.

Mironov, I. Rényi differential privacy. In *Proc. IEEE 30th Computer Security Foundations Symp.*, 2017.

Nguyen, L. M., Nguyen, P. H., van Dijk, M., Richtárik, P., Scheinberg, K., and Takác, M. SGD and Hogwild! convergence without the bounded gradients assumption. *Proc. Int. Conf. on Machine Learn.*, 80:3747–3755, 2018.

Ranbaduge, T. and Ding, M. Differentially private vertical federated learning. *arXiv preprint arXiv:2211.06782*, 2022.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In *Proc. 12th ACM Workshop Artificial Intelligence and Security*, pp. 1–11, 2019.

Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., and Wei, W. Ldp-fed: Federated learning with local differential privacy. In *Proc. Third ACM Int. Workshop Edge Systems, Analytics and Networking*, pp. 61—66, 2020.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Vincent Poor, H. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.*, 15: 3454–3469, 2020.

Xu, R., Baracaldo, N., Zhou, Y., Anwar, A., Joshi, J., and Ludwig, H. Fedv: Privacy-preserving federated learning over vertically partitioned data. In *Proc. 14th ACM Workshop Artificial Intelligence and Security*, pp. 181–192, 2021.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.

A PROOF OF THEOREM 5.2

Let Θ^t be the set of model parameters in iteration t. For brevity, we let $\hat{h}_i^t$ denote $\hat{h}(\boldsymbol{\theta}_1^t, \ldots, \boldsymbol{\theta}_M^t; \mathbf{x}^i)$. We can write the update rule for Θ as

$$\Theta^{t+1} = \Theta^t - \eta G^t \tag{11}$$

with

$$G^t := \frac{1}{B} \sum_{i \in \mathcal{B}^t} \nabla \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t + \varepsilon_i^t) \tag{12}$$

where ε_i^t is the P-vector of noise resulting from the PBM for the embedding sum of sample i in iteration t.

With some abuse of notation, we let $\hat{h}^t$ denote concatenation of the embedding sums for the minibatch $\mathcal{B}^t$ (without noise) and let $\nabla \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \hat{h}^t)$ denote the average stochastic gradient of $\mathcal{L}$ over minibatch $\mathcal{B}^t$.

We first bound the difference between G^t and $\nabla \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \hat{h}^t)$ in the following lemma.

Lemma A.1. *It holds that*

$$\mathbb{E}_{\mathcal{B}^t} \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \hat{h}^t)\|^2 \leq \frac{C^2 M P (L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^M H_m^2)}{4\beta^2 b}. \tag{13}$$

Proof. We first note that

$$\mathbb{E}_{\mathcal{B}^t} \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \hat{h}^t)\|^2 = \sum_{m=0}^{M} \mathbb{E}_{\mathcal{B}^t} \|G_m^t - \nabla_m \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0^t, \hat{h}^t)\|^2 \tag{14}$$

where G_m^t is the block of G^t corresponding to party m.

For $m = 0$, using Assumption 3.1, we can bound the first term in the summation in (14) as

$$\mathbb{E}_{\mathcal{B}^t} \|G_0^t - \nabla_0 \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0, \hat{h})\|^2$$
$$\leq \frac{1}{B} \sum_{i \in \mathcal{B}^t} \mathbb{E}_{\mathcal{B}^t} \|\nabla \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t + \varepsilon_i^t) - \nabla \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t)\|^2 \tag{15}$$
$$\leq \frac{L_0^2}{B} \sum_{i \in \mathcal{B}^t} \mathbb{E}_{\mathcal{B}^t} \|\varepsilon_i^t\|^2. \tag{16}$$

For $m = 1, \ldots M$, by the chain rule, we have

$$G_m^t = \frac{1}{B} \sum_{i \in \mathcal{B}} \nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{x}_i) \nabla_{h_m} \hat{h}_i^t \nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t + \varepsilon_i^t) \tag{17}$$

$$\nabla_m \mathcal{L}_{\mathcal{B}^t}(\theta_0^t, \hat{h}^t) = \frac{1}{B} \sum_{i \in \mathcal{B}} \nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{x}_i) \nabla_{h_m} \hat{h}_i^t \nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t). \tag{18}$$

It follows that

$$\mathbb{E}_{\mathcal{B}^t} \|G_m^t - \nabla_m \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \hat{h}^t)\|^2 = \mathbb{E}_{\mathcal{B}^t} \left(\frac{1}{B^2} \sum_{i \in \mathcal{B}^t} \| \nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{x}_i) \nabla_{h_m} \hat{h}_i^t \left(\nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t + \varepsilon_i^t) - \nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t) \right) \|^2 \right). \tag{19}$$

Noting that $\nabla_{\hat{h}} \hat{h}_i^t = \boldsymbol{I}$, we have

$$\begin{aligned}
&\mathbb{E}_{\mathcal{B}^t} \|G_m^t - \nabla_m \mathcal{L}_{\mathcal{B}}(\boldsymbol{\theta}_0^t, \hat{h}^t)\|^2 \\
&\leq \frac{1}{B} \sum_{i \in \mathcal{B}^t} \mathbb{E}_{\mathcal{B}^t} \|\nabla_m h_m(\boldsymbol{\theta}_m^t; \mathbf{x}_i)\|_{\mathcal{F}}^2 \|\nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t + \varepsilon_i^t) \\
&\qquad\qquad - \nabla_{\hat{h}} \ell(\boldsymbol{\theta}_0^t, \hat{h}_i^t)\|^2 \qquad (20) \\
&\leq \frac{H_m^2 L_{\hat{h}}^2}{B} \sum_{i \in \mathcal{B}^t} \mathbb{E}_{\mathcal{B}^t} \|\varepsilon_i^t\|^2 \qquad (21)
\end{aligned}$$

where (21) follows from (20) by Assumptions 3.1 and 3.5.

Combining (16) and (21), we obtain

$$\begin{aligned}
&\mathbb{E}_{\mathcal{B}^t} \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\boldsymbol{\theta}_0, \hat{h}^t)\|^2 \leq \qquad (22) \\
&\qquad \left(L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^{M} H_m^2\right) \frac{1}{B} \sum_{i \in \mathcal{B}^t} \mathbb{E}_{\mathcal{B}^t} \|\varepsilon_i^t\|^2 \qquad (23) \\
&\leq \frac{C^2 M P (L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^{M} H_m^2)}{4\beta^2 b} \qquad (24)
\end{aligned}$$

where (24) follows from (23) by Theorem 2.5. □

We now prove Theorem 5.2.

Proof. By Assumption 3.1, we have

$$\begin{aligned}
&\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t) \\
&\leq -\langle \nabla \mathcal{L}(\Theta^t), \Theta^{t+1} - \Theta^t \rangle + \frac{L}{2} \| \Theta^{t+1} - \Theta^t \|^2 \qquad (25) \\
&= -\eta \langle \nabla \mathcal{L}(\Theta^t), G^t \rangle + \frac{L\eta^2}{2} \|G^t\|^2 \qquad (26) \\
&= -\eta \langle \nabla \mathcal{L}(\Theta^t), G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) \rangle \\
&\quad - \eta \langle \nabla \mathcal{L}(\Theta^t), \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) \rangle \\
&\quad + \frac{L\eta^2}{2} \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) + \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \qquad (27) \\
&\leq -\eta \langle \nabla \mathcal{L}(\Theta^t), G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) \rangle \\
&\quad - \eta \langle \nabla \mathcal{L}(\Theta^t), \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) \rangle \\
&\quad + L\eta^2 \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 + L\eta^2 \|\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2. \qquad (28)
\end{aligned}$$

Taking expectation with respect to t, conditioned on Θ^t:

$$\begin{aligned}
&\mathbb{E}_t \left(\mathcal{L}(\Theta^{t+1})\right) - \mathcal{L}(\Theta^t) \\
&\leq \frac{\eta}{2} \|\nabla \mathcal{L}(\Theta^t)\|^2 + \frac{\eta}{2} \mathbb{E}_t \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \\
&\quad - \eta \langle \nabla \mathcal{L}(\Theta^t), \mathbb{E}_t \left(\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\right) \rangle \\
&\quad + L\eta^2 \mathbb{E}_t \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \\
&\quad + L\eta^2 \mathbb{E}_t \|\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \qquad (29) \\
&= -\frac{\eta}{2} \|\nabla \mathcal{L}(\Theta^t)\|^2 \\
&\quad \frac{\eta}{2} (1 + 2L\eta) \mathbb{E}_t \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \\
&\quad + L\eta^2 \mathbb{E}_t \|\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \qquad (30) \\
&= -\frac{\eta}{2} \|\nabla \mathcal{L}(\Theta^t)\|^2 \\
&\quad \frac{\eta}{2} (1 + 2L\eta) \mathbb{E}_t \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \\
&\quad + L\eta^2 \mathbb{E}_t \|\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) - \nabla \mathcal{L}(\Theta^t)\|^2 \\
&\quad + L\eta^2 \mathbb{E}_t \|\nabla \mathcal{L}(\Theta^t)\|^2 \qquad (31) \\
&\leq -\frac{\eta}{2} (1 - 2L\eta) \|\nabla \mathcal{L}(\Theta^t)\|^2 \\
&\quad + \frac{\eta}{2} (1 + 2L\eta) \mathbb{E}_t \|G^t - \nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t)\|^2 \\
&\quad + L\eta^2 \mathbb{E}_t \|\nabla \mathcal{L}_{\mathcal{B}^t}(\Theta^t) - \nabla \mathcal{L}(\Theta^t)\|^2 \qquad (32)
\end{aligned}$$

where (29) follows from (28) by the fact that $A \cdot B = \frac{1}{2}A^2 + \frac{1}{2}B^2 - \frac{1}{2}(A - B)^2$, (30) follows from (29) by Assumption 3.2, and (31) also follows from (30) by Assumption 3.2.

We apply Assumption 3.3 and Lemma A.1 to (32):

$$\begin{aligned}
\mathbb{E}_t \left(\mathcal{L}(\Theta^{t+1})\right) - \mathcal{L}(\Theta^t) &\leq -\frac{\eta}{2} (1 - 2L\eta) \|\nabla \mathcal{L}(\Theta^t)\|^2 \\
&\quad + \frac{\eta}{2}(1 + 2L\eta) \left(\frac{C^2 M P (L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^{M} H_m^2)}{4\beta^2 b}\right) \\
&\quad + \frac{L\eta^2 \sigma^2}{B}. \qquad (33)
\end{aligned}$$

We next apply the assumption that $\eta < \frac{1}{2L}$ and rearrange (33) to obtain

$$\begin{aligned}
\|\nabla \mathcal{L}(\Theta^t)\|^2 &\leq \frac{2(\mathcal{L}(\Theta^t) - \mathbb{E}_t(\mathcal{L}(\Theta^{t+1}))}{\eta} \\
&\quad + (1 + 2L\eta) \left(\frac{C^2 M P (L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^{M} H_m^2)}{4\beta^2 b}\right) \\
&\quad + 2L\eta \frac{\sigma^2}{B}. \qquad (34)
\end{aligned}$$

Averaging over T iterations and taking total expectation give

$$\begin{aligned}
\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left(\|\nabla \mathcal{L}(\Theta^t)\|^2\right) &\leq \frac{2(\mathcal{L}(\Theta^0) - \mathbb{E}(\mathcal{L}(\Theta^T))}{\eta T} \\
&\quad + (1 + 2L\eta) \left(\frac{C^2 M P (L_0^2 + L_{\hat{h}}^2 \sum_{m=1}^{M} H_m^2)}{4\beta^2 b}\right) \\
&\quad + 2L\eta \frac{\sigma^2}{B}. \qquad (35)
\end{aligned}$$

□