Compressed-VFL: Communication-Efficient Learning with Vertically Partitioned Data

Timothy Castiglia [1] **Anirban Das** [1] **Shiqiang Wang** [2] **Stacy Patterson** [1]

Abstract

We propose Compressed Vertical Federated Learning (C-VFL) for communication-efficient training on vertically partitioned data. In C-VFL, a server and multiple parties collaboratively train a model on their respective features utilizing several local iterations and sharing compressed intermediate results periodically. Our work provides the first theoretical analysis of the effect message compression has on distributed training over vertically partitioned data. We prove convergence of non-convex objectives at a rate of $O(\frac{1}{\sqrt{T}})$ when the compression error is bounded over the course of training. We provide specific requirements for convergence with common compression techniques, such as quantization and top-k sparsification. Finally, we experimentally show compression can reduce communication by over 90% without a significant decrease in accuracy over VFL without compression.

1. Introduction

Federated Learning (McMahan et al., 2017) is a distributed machine learning approach that has become of much interest in both theory (Li et al., 2020; Wang et al., 2019; Liu et al., 2020) and practice (Bonawitz et al., 2019; Rieke et al., 2020; Lim et al., 2020) in recent years. Naive distributed learning algorithms may require frequent exchanges of large amounts of data, which can lead to slow training performance (Lin et al., 2020). Further, participants may be globally distributed, with high latency network connections. To mitigate these factors, Federated Learning algorithms aim to be communication-efficient by design. Methods such as *local updates* (Moritz et al., 2016; Liu et al., 2019), where parties train local parameters for multiple iterations without communication, and message compression (Stich et al., 2018; Wen et al., 2017; Karimireddy et al., 2019) reduce message frequency and size, respectively, with little impact on training performance.

Federated Learning methods often target the case where the data among parties is distributed horizontally: each party's data shares the same features but parties hold data corresponding to different sample IDs. This is known as Horizontal Federated Learning (HFL) (Yang et al., 2019). However, there are several application areas where data is partitioned in a *vertical* manner: the parties store data on the same sample IDs but different feature spaces.

An example of a vertically partitioned setting includes a hospital, bank, and insurance company seeking to train a model to predict something of mutual interest, such as customer credit score. Each of these institutions may have data on the same individuals but store medical history, financial transactions, and vehicle accident reports, respectively. These features must remain local to the institutions due to privacy concerns, rules and regulations (e.g., GDPR, HIPAA), and/or communication network limitations. In such a scenario, Vertical Federated Learning (VFL) methods must be employed. Although VFL is less well-studied than HFL, there has been a growing interest in VFL algorithms recently (Hu et al., 2019; Gu et al., 2021; Cha et al., 2021), and VFL algorithms have important applications including risk prediction, smart manufacturing, and discovery of pharmaceuticals (Kairouz et al., 2021).

Typically in VFL, each party trains a local embedding function that maps raw data features to a meaningful vector representation, or *embedding*, for prediction tasks. For example, a neural network can be an embedding function for mapping the text of an online article to a vector space for classification (Koehrsen, 2018). Referring to Figure 1, suppose Party 1 is a hospital with medical data features x_1. The hospital computes its embedding $h_1(\theta_1; x_1)$ for the features by feeding x_1 through a neural network. The other parties (the bank and insurance company), compute embeddings for their features, then all parties share the embeddings in a private manner (e.g., homomorphic encryption, secure multi-party computation, or secure aggregation). The embeddings

[1]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA [2]IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Correspondence to: Timothy Castiglia <castit@rpi.edu>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

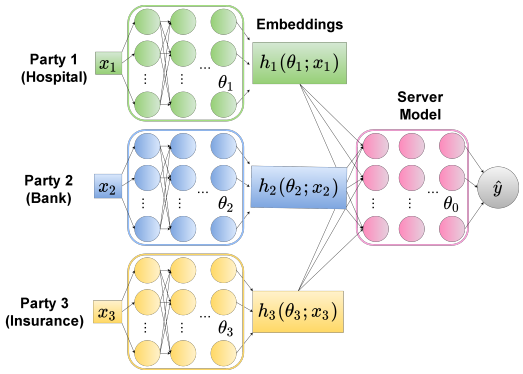


Figure 1. Example global model with neural networks. To obtain a $\hat{y}$ prediction for a data sample x, each party m feeds the local features of x, x_m, into a neural network. The output of this neural network is the embedding $h_m(\theta_m; x_m)$. All embeddings are then fed into the server model neural network with parameters θ_0.

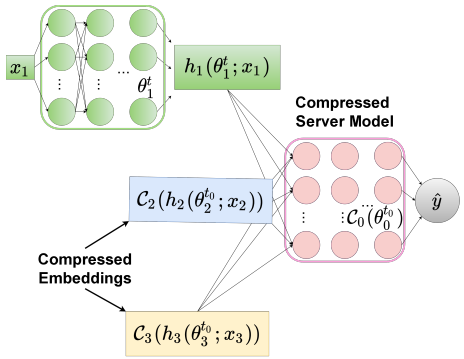


Figure 2. Example local view of a global model with neural networks. When running C-VFL, Party 1 (in green) only has a compressed snapshot of the other parties embeddings and the server model. To calculate $\hat{y}$, Party 1 uses its own embedding calculated at iteration t, and the embeddings and server model calculated at time t_0, the latest communication iteration, and compressed with $\mathcal{C}_m$.

are then combined in a *server model* θ_0 to determine the final loss of the global model. A server model (or fusion network) captures the complicated interactions of embeddings and is often a complex, non-linear model (Gu et al., 2019; Nie et al., 2021; Han et al., 2021b). Embeddings can be very large, in practice, sometimes requiring terabytes of communication over the course of training.

Motivated by this, we propose Compressed Vertical Federated Learning (C-VFL), a general framework for communication-efficient Federated Learning over vertically partitioned data. In our algorithm, parties communicate compressed embeddings periodically, and the parties and the server each run block-coordinate descent for multiple local iterations, in parallel, using stochastic gradients to update their local parameters.

C-VFL is the first theoretically verified VFL algorithm that applies embedding compression. Unlike in HFL algorithms, C-VFL compresses embeddings rather than gradients. Previous work has proven convergence for HFL algorithms with gradient compression (Stich et al., 2018; Wen et al., 2017; Karimireddy et al., 2019). However, no previous work analyzes the convergence requirements for VFL algorithms that use embedding compression. Embeddings are parameters in the partial derivatives calculated at each party. The effect of compression error on the resulting partial derivatives may be complex; therefore, the analysis in previous work on gradient compression in HFL does not apply to compression in VFL. In our work, we prove that, under a diminishing compression error, C-VFL converges at a rate of $O(\frac{1}{\sqrt{T}})$, which is comparable to previous VFL algorithms that do not employ compression. We also analyze common compressors, such as quantization and sparsification, in C-VFL and provide bounds on their compression parameters to ensure convergence.

C-VFL also generalizes previous work by supporting an arbitrary server model. Previous work in VFL has either only analyzed an arbitrary server model without local updates (Chen et al., 2020), or analyzed local updates with a linear server model (Liu et al., 2019; Zhang et al., 2020; Das & Patterson, 2021). C-VFL is designed with an arbitrary server model, allowing support for more complex prediction tasks than those supported by previous VFL algorithms.

We summarize our main contributions in this work.

1. We introduce C-VFL with an arbitrary compression scheme. Our algorithm generalizes previous work in VFL by including both an arbitrary server model and multiple local iterations.

2. We prove convergence of C-VFL to a neighborhood of a fixed point on non-convex objectives at a rate of $O(\frac{1}{\sqrt{T}})$ for a fixed step size when the compression error is bounded over the course of training. We also prove that the algorithm convergence error goes to zero for a diminishing step size if the compression error diminishes as well. Our work provides novel analysis for the effect of compressing embeddings on convergence in a VFL algorithm. Our analysis also applies to Split Learning when uploads to the server are compressed.

3. We provide convergence bounds on parameters in common compressors that can be used in C-VFL. In particular, we examine scalar quantization (Bennett, 1948), lattice vector quantization (Zamir & Feder, 1996), and top-k sparsification (Lin et al., 2018).

4. We evaluate our algorithm by training on MIMIC-III, CIFAR-10, and ModelNet10 datasets. We empirically show

how C-VFL can reduce the number of bits sent by over 90% compared to VFL with no compression without a significant loss in accuracy of the final model.

Related Work. (Richtárik & Takác, 2016; Hardy et al., 2017) were the first works to propose Federated Learning algorithms for vertically partitioned data. (Chen et al., 2020; Romanini et al., 2021) propose the inclusion of an arbitrary server model in a VFL algorithm. However, these works do not consider multiple local iterations, and thus communicate at every iteration. (Liu et al., 2019), (Feng & Yu, 2020), and (Das & Patterson, 2021) all propose different VFL algorithms with local iterations for vertically partitioned data but do not consider an arbitrary server model. Split Learning is a related concept to VFL (Gupta & Raskar, 2018). Split Learning can be thought of a special case of VFL when there is only one party. Recent works (He et al., 2020; Han et al., 2021a) have extended Split Learning to a Federated Learning setting. However, these works focus on the HFL setting and do not apply message compression. In contrast to previous works, our work addresses a vertical scenario, an arbitrary server model, local iterations, and message compression.

Message compression is a common topic in HFL scenarios, where participants exchange gradients determined by their local datasets. Methods of gradient compression in HFL include scalar quantization (Bernstein et al., 2018), vector quantization (Shlezinger et al., 2021), and top-k sparsification (Shi et al., 2019). In C-VFL, compressed embeddings are shared, rather than compressed gradients. Analysis in previous work on gradient compression in HFL does not apply to compression in VFL, as the effect of embedding compression error on each party's partial derivatives may be complex. No prior work has analyzed the impact of compression on convergence in VFL.

Outline. In Section 2, we provide the problem formulation and our assumptions. Section 3 presents the details of C-VFL. In Section 4, we present our main theoretical results. Our experimental results are given in Section 5. Finally, we conclude in Section 6.

2. Problem Formulation

We present our problem formulation and notation to be used in the rest of the paper. We let $\|a\|$ be the 2-norm of a vector a, and let $\|\mathbf{A}\|_{\mathcal{F}}$ be the Frobenius norm of a matrix $\mathbf{A}$.

We consider a set of M parties $\{1, \dots, M\}$ and a server. The dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$ is vertically partitioned a priori across the M parties, where N is the number of data samples and D is the number of features. The i-th row of $\mathbf{X}$ corresponds to a data sample x^i. For each sample x^i, a party m holds a disjoint subset of the features, denoted x_m^i, so that $x^i = [x_1^i, \dots, x_M^i]$. For each x^i, there is a corresponding label y^i. Let $\mathbf{y} \in \mathbb{R}^{N \times 1}$ be the vector of all sample labels. We let $\mathbf{X}_m \in \mathbb{R}^{N \times D_m}$ be the local dataset of a party m, where the i-th row correspond to data features x_m^i. We assume that the server and all parties have a copy of the labels $\mathbf{y}$. For scenarios where the labels are private and only present at a single party, the label holder can provide enough information for the parties to compute gradients for some classes of model architectures (Liu et al., 2019).

Each party m holds a set of model parameters θ_m as well as a local *embedding* function $h_m(\cdot)$. The server holds a set of parameters θ_0 called the *server model* and a loss function $l(\cdot)$ that combines the *embeddings* $h_m(\theta_m; x_m^i)$ from all parties. Our objective is to minimize the following:

$$F(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{N} \sum_{i=1}^{N} l(\theta_0, h_1(\theta_1; x_1^i), \dots, h_M(\theta_M; x_M^i); y^i) \quad (1)$$

where $\Theta = [\theta_0^T, \dots, \theta_M^T]^T$ is the *global model*. An example of a global model Θ is in Figure 1.

For simplicity, we let $m = 0$ refer to the server, and define $h_0(\theta_0; x^i) := \theta_0$ for all x^i, where $h_0(\cdot)$ is equivalent to the identity function. Let $h_m(\theta_m; x_m^i) \in \mathbb{R}^{P_m}$ for $m = 0, \dots, M$, where P_m is the size of the m-th embedding. Let $\nabla_m F(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta_m} l(\theta_0, h_1(\theta_1; x_1^i), \dots, h_M(\theta_M; x_M^i); y^i)$ be the partial derivatives for parameters θ_m.

Let $\mathbf{X}^{\mathbf{B}}$ and $\mathbf{y}^{\mathbf{B}}$ be the set of samples and labels corresponding to a randomly sampled mini-batch $\mathbf{B}$ of size B. We let the stochastic partial derivatives for parameters θ_m be $\nabla_m F_{\mathbf{B}}(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{B} \sum_{x^i, y^i \in \mathbf{X}^{\mathbf{B}}, \mathbf{y}^{\mathbf{B}}} \nabla_{\theta_m} l(\theta_0, h_1(\theta_1; x_1^i), \dots, h_M(\theta_M; x_M^i); y)$. We may drop $\mathbf{X}$ and $\mathbf{y}$ from $F(\cdot)$ and $F_{\mathbf{B}}(\cdot)$. With a minor abuse of notation, we let $h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}}) := \{h_m(\theta_m; x_m^{\mathbf{B}^1}), \dots, h_m(\theta_m; x_m^{\mathbf{B}^B})\}$ be the set of all party m embeddings associated with mini-batch $\mathbf{B}$, where $\mathbf{B}^i$ is the i-th sample in the mini-batch $\mathbf{B}$. We let $\nabla_m F_{\mathbf{B}}(\Theta)$ and $\nabla_m F_{\mathbf{B}}(\theta_0, h_1(\theta_1; \mathbf{X}_1^{\mathbf{B}}), \dots, h_M(\theta_M; \mathbf{X}_M^{\mathbf{B}}))$ be equivalent, and use them interchangeably.

Assumption 1. **Smoothness**: There exists positive constants $L < \infty$ and $L_m < \infty$, for $m = 0, \dots, M$, such that for all Θ_1, Θ_2, the objective function satisfies:

$$\|\nabla F(\Theta_1) - \nabla F(\Theta_2)\| \leq L \|\Theta_1 - \Theta_2\|$$
$$\|\nabla_m F_{\mathbf{B}}(\Theta_1) - \nabla_m F_{\mathbf{B}}(\Theta_2)\| \leq L_m \|\Theta_1 - \Theta_2\|.$$

Assumption 2. **Unbiased gradients**: For $m = 0, \dots, M$, for every mini-batch $\mathbf{B}$, the stochastic partial derivatives are unbiased, i.e., $\mathbb{E}_{\mathbf{B}} \nabla_m F_{\mathbf{B}}(\Theta) = \nabla_m F(\Theta)$.

Assumption 3. **Bounded variance**: For $m = 0, \dots, M$, there exists constants $\sigma_m < \infty$ such that the variance

of the stochastic partial derivatives are bounded as: $\mathbb{E}_{\mathbf{B}}\|\nabla_m F(\Theta) - \nabla_m F_{\mathbf{B}}(\Theta)\|^2 \leq \frac{\sigma_m^2}{B}$ for a mini-batch $\mathbf{B}$ of size B.

Assumption 1 bounds how fast the gradient and stochastic partial derivatives can change. Assumptions 2 and 3 require that the stochastic partial derivatives are unbiased estimators of the true partial derivatives with bounded variance. Assumptions 1–3 are common assumptions in convergence analysis of gradient-based algorithms (Tsitsiklis et al., 1986; Nguyen et al., 2018; Bottou et al., 2018). We note Assumptions 2–3 are similar to the IID assumptions in HFL convergence analysis. However, in VFL settings, all parties store identical sample IDs but different subsets of features. Hence, there is no equivalent notion of a non-IID distribution in VFL.

Assumption 4. **Bounded Hessian**: There exists positive constants H_m for $m = 0, \ldots, M$ such that for all Θ, the second partial derivatives of $F_{\mathbf{B}}$ with respect to $h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}})$ satisfy:

$$\|\nabla^2_{h_m(\theta_m;\mathbf{X}_m^{\mathbf{B}})} F_{\mathbf{B}}(\Theta)\|_{\mathcal{F}} \leq H_m \tag{2}$$

for any mini-batch $\mathbf{B}$.

Assumption 5. **Bounded Embedding Gradients**: There exists positive constants G_m for $m = 0, \ldots, M$ such that for all θ_m, the stochastic embedding gradients are bounded:

$$\|\nabla_{\theta_m} h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}})\|_{\mathcal{F}} \leq G_m \tag{3}$$

for any mini-batch $\mathbf{B}$.

Since we are assuming a Lipschitz-continuous loss function (Assumption 1), we know the Hessian of F is bounded. Assumption 4 strengthens this assumption slightly to also bound the Hessian over any mini-batch. Assumption 5 bounds the magnitude of the partial derivatives with respect to embeddings. This embedding gradient bound is necessary to ensure convergence in the presence of embedding compression error (see Appendix A.2 for details).

3. Algorithm

We now present C-VFL, a communication-efficient method for training a global model with vertically partitioned data. In each *global round*, a mini-batch $\mathbf{B}$ is chosen randomly from all samples and parties share necessary information for local training on this mini-batch. Each party m, in parallel, runs block-coordinate stochastic gradient descent on its local model parameters θ_m for Q local iterations. C-VFL runs for a total of R global rounds, and thus runs for $T = RQ$ total local iterations.

For party m to compute the stochastic gradient with respect to its features, it requires the embeddings computed by all other parties $j \neq m$. In C-VFL, these embeddings are shared with the server then distributed to the parties. We reduce communication cost by only sharing embeddings every global round. Further, each party compresses their embeddings before sharing. We define a set of general compressors for compressing party embeddings and the server model: $\mathcal{C}_m(\cdot) : \mathbb{R}^{P_m} \to \mathbb{R}^{P_m}$ for $m = 0, \ldots, M$. To calculate the gradient for data sample x^i, party m receives $\mathcal{C}_j(h_j(\theta_j; x_j^i))$ from all parties $j \neq m$. With this information, a party m can compute $\nabla_m F_{\mathbf{B}}$ and update its parameters θ_m for multiple local iterations. Note that each party uses a stale view of the global model to compute its gradient during these local iterations, as it is reusing the embeddings it receives at the start of the round. In Section 4, we show that C-VFL converges even though parties use stale information. An example view a party has of the global model during training is in Figure 2. Here, t is the current iteration and t_0 is the start of the most recent global round, when embeddings were shared.

Algorithm 1 Compressed Vertical Federated Learning

1: **Initialize:** θ_m^0 for all parties m and server model θ_0^0
2: **for** $t \leftarrow 0, \ldots, T-1$ **do**
3: **if** t mod $Q = 0$ **then**
4: Randomly sample $\mathbf{B}^t \in \{\mathbf{X}, \mathbf{y}\}$
5: **for** $m \leftarrow 1, \ldots, M$ in parallel **do**
6: Send $\mathcal{C}_m(h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t}))$ to server
7: **end for**
8: $\hat{\Phi}^{t_0} \leftarrow \{\mathcal{C}_0(\theta_0^t), \mathcal{C}_1(h_1(\theta_1^t)), \ldots, \mathcal{C}_M(h_M(\theta_M^t))\}$
9: Server sends $\hat{\Phi}^{t_0}$ to all parties
10: **end if**
11: **for** $m \leftarrow 0, \ldots, M$ in parallel **do**
12: $\hat{\Phi}_m^t \leftarrow \{\hat{\Phi}_{-m}^{t_0}; h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}})\}$
13: $\theta_m^{t+1} \leftarrow \theta_m^t - \eta^{t_0} \nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t; \mathbf{y}^{\mathbf{B}^{t_0}})$
14: **end for**
15: **end for**

Algorithm 1 details the procedure of C-VFL. In each global round, when t mod $Q = 0$, a mini-batch $\mathbf{B}$ is randomly sampled from $\mathbf{X}$ and the parties exchange the associated embeddings, compressed using $\mathcal{C}_m(\cdot)$, via the server (lines 3-9). Each party m completes Q local iterations, using the compressed embeddings it received in iteration t_0 and its own m-th uncompressed embeddings $h_m(\theta_m^t, \mathbf{X}_m^{\mathbf{B}^{t_0}})$. We denote the set of embeddings that each party receives in iteration t_0 as:

$$\hat{\Phi}^{t_0} := \{\mathcal{C}_0(\theta_0^{t_0}), \mathcal{C}_1(h_1(\theta_1^{t_0})), \ldots, \mathcal{C}_M(h_M(\theta_M^{t_0}))\}. \tag{4}$$

We let $\hat{\Phi}_{-m}^{t_0}$ be the set of compressed embeddings from parties $j \neq m$, and let $\hat{\Phi}_m^t := \{\hat{\Phi}_{-m}^{t_0}; h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}})\}$. For each local iteration, each party m updates θ_m by computing the stochastic partial derivatives $\nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t; \mathbf{y}^{\mathbf{B}^{t_0}})$ and applying a gradient step with step size η^{t_0} (lines 11-14).

A key difference of C-VFL from previous VFL algorithms

is the support of a server model with trainable parameters, allowing for arbitrary fusion networks. To support such a model with multiple local iterations, the server model parameters are shared with the parties. Also note that the same mini-batch is used for all Q local iterations, thus communication is only required every Q iterations. Therefore, without any compression, the total communication cost is $O(R \cdot M \cdot (B \cdot \sum_m P_m + |\theta_0|))$ for R global rounds. Our compression technique replaces P_m and $|\theta_0|$ with smaller values based on the compression factor. For cases where embeddings, the batch size, and the server model are large, this reduction can greatly decrease the communication cost.

Privacy. We now discuss privacy-preserving mechanisms for C-VFL. In HFL settings, model update or gradient information is shared in messages. It has been shown that gradients can leak information about the raw data (Phong et al., 2018; Geiping et al., 2020). However in C-VFL, parties only share embeddings and can only calculate the partial derivatives associated with the server model and their local models. Commonly proposed HFL gradient attacks cannot be performed on C-VFL. Embeddings may be vulnerable to model inversion attacks (Mahendran & Vedaldi, 2015), which are methods by which an attacker can recover raw input to a model using the embedding output and black-box access to the model. One can protect against such an attack using homomorphic encryption (Cheng et al., 2021; Hardy et al., 2017) or secure multi-party computation (Gu et al., 2021). Alternatively, if the input to the server model is the sum of party embeddings, then secure aggregation methods (Bonawitz et al., 2016) can be applied. Several works have proposed privacy-perserving methods for VFL settings (Cheng et al., 2021; Çatak, 2015; Zheng et al., 2022) that are compatible with the C-VFL algorithm.

Note that C-VFL assumes all parties have access to the labels. For low-risk scenarios, such as predicting credit score, labels may not need to be private among the parties. In cases where labels are private, one can augment C-VFL to apply the method in (Liu et al., 2019) for gradient calculation without the need for sharing labels. Our analysis in Section 4 would still hold in this case, and the additional communication is reduced by the use of message compression.

4. Analysis

In this section, we discuss our analytical approach and present our theoretical results. We first define the compression error associated with $\mathcal{C}_m(\cdot)$:

Definition 4.1. Compression Error: Let vectors $\epsilon_m^{x^i}$ for $m = 0, \ldots, M$, be the compression errors of $\mathcal{C}_m(\cdot)$ on a data sample x^i: $\epsilon_m^{x^i} := \mathcal{C}_m(h_m(\theta_m; x^i)) - h_m(\theta_m; x^i)$. Let $\epsilon_m^{t_0}$ be the $P_m \times B$ matrix with $\epsilon_m^{x^i}$ for all data samples x^i in mini-batch $\mathbf{B}^{t_0}$ as the columns. We denote the expected squared message compression error from party m at round t_0 as $\mathcal{E}_m^{t_0} := \mathbb{E}\,\|\epsilon_m^{t_0}\|_{\mathcal{F}}^2$.

Let $\hat{\mathbf{G}}^t$ be the stacked partial derivatives at iteration t:

$$\hat{\mathbf{G}}^t := [(\nabla_0 F_{\mathbf{B}}(\hat{\Phi}_0^t; \mathbf{y}^{\mathbf{B}^{t_0}}))^T, \ldots, (\nabla_M F_{\mathbf{B}}(\hat{\Phi}_M^t; \mathbf{y}^{\mathbf{B}^{t_0}}))^T]^T.$$

The model Θ evolves as:

$$\Theta^{t+1} = \Theta^t - \eta^{t_0} \hat{\mathbf{G}}^t. \tag{5}$$

We note the reuse of the mini-batch of $\mathbf{B}^{t_0}$ for Q iterations in this recursion. This indicates that the stochastic gradients are not unbiased during local iterations $t_0 + 1 \leq t \leq t_0 + Q - 1$. Using conditional expectation, we can apply Assumption 2 to the gradient calculated at iteration t_0 when there is no compression error. We define Φ^{t_0} to be the set of embeddings that would be received by each party at iteration t_0 if no compression error were applied:

$$\Phi^{t_0} := \{\theta_0^{t_0}, h_1(\theta_1^{t_0}), \ldots, h_M(\theta_M^{t_0})\}. \tag{6}$$

We let $\Phi_{-m}^{t_0}$ be the set of embeddings from parties $j \neq m$, and let $\Phi_m^t := \{\Phi_{-m}^{t_0}; h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}})\}$. Then, if we take expectation over $\mathbf{B}^{t_0}$ conditioned on previous global models Θ^t up to t_0:

$$\mathbb{E}_{\mathbf{B}^{t_0}}[\nabla_m F_{\mathbf{B}}(\Phi_m^{t_0}) \mid \{\Theta^\tau\}_{\tau=0}^{t_0}] = \nabla_m F(\Phi_m^{t_0}). \tag{7}$$

With the help of (7), we can prove convergence by bounding the difference between the gradient at the start of each global round and those calculated during local iterations (see the proof of Lemma 2 in Appendix A.2 for details).

To account for compression error, using the chain rule and Taylor series expansion, we obtain:

Lemma 1. Under Assumptions 4-5, the norm of the difference between the objective function value with compressed and uncompressed embeddings is bounded as:

$$\mathbb{E}\|\nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t) - \nabla_m F_{\mathbf{B}}(\Phi_m^t)\|^2 \leq H_m^2 G_m^2 \sum_{j=0, j\neq m}^{M} \mathcal{E}_j^{t_0}.$$

The proof of Lemma 1 is given in Appendix A.2. Using Lemma 1, we can bound the effect of compression error on convergence.

We present our main theoretical results. All proofs are provided in Appendix A.

Theorem 4.2. ***Convergence with fixed step size****: Under Assumptions 1-5, if $\eta^{t_0} = \eta$ for all iterations and satisfies $\eta^{t_0} \leq \frac{1}{16Q \max\{L, \max_m L_m\}}$, then the average squared*

Table 1. Choice of common compressor parameters to achieve a convergence rate of $O(1/\sqrt{T})$. P_m is the size of the m-th embedding. In scalar quantization, we let there be 2^q quantization levels, and let $h_{\max}$ and $h_{\min}$ be respectively the maximum and minimum components in $h_m(\theta_m^t; x_m^i)$ for all iterations t, parties m, and x_m^i. We let V be the size of the lattice cell in vector quantization. We let k be the number of parameters sent in an embedding after top-k sparsification, and $(\|h\|^2)_{\max}$ be the maximum value of $\|h_m(\theta_m^t; x_m^i)\|^2$ for all iterations t, parties m, and x_m^i.

	Scalar Quantization	**Vector Quantization**	**Top-k Sparsification**
Parameter Choice	$q = \Omega\left(\log_2\left(BP_m(h_{\max}-h_{\min})^2\sqrt{T}\right)\right)$	$V = O\left(\frac{1}{BP_m\sqrt{T}}\right)$	$k = \Omega\left(P_m - \frac{P_m}{B(\|h\|^2)_{\max}\sqrt{T}}\right)$
Compression Error	$\mathcal{E}_m^{t_0} \leq BP_m\frac{(h_{\max}-h_{\min})^2}{12}2^{-2q}$	$\mathcal{E}_m^{t_0} \leq \frac{VBP_m}{24}$	$\mathcal{E}_m^{t_0} \leq B(1-\frac{k}{P_m})(\|h\|^2)_{\max}$

gradient over R global rounds of Algorithm 1 is bounded:

$$\frac{1}{R}\sum_{t_0=0}^{R-1}\mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \leq \frac{4\left[F(\Theta^0)-\mathbb{E}\left[F(\Theta^T)\right]\right]}{\eta T} + 6\eta QL\sum_{m=0}^{M}\frac{\sigma_m^2}{B} + \frac{92Q^2}{R}\sum_{m=0}^{M}H_m^2G_m^2\sum_{t_0=0}^{R-1}\sum_{j=0,j\neq m}^{M}\mathcal{E}_j^{t_0}. \quad (8)$$

The first term in (8) is based on the difference between the initial model and final model of the algorithm. The second term is the error associated with the variance of the stochastic gradients and the Lipschitz constants L and L_m's. The third term relates to the average compression error over all iterations. The larger the error introduced by a compressor, the larger the convergence error is. We note that setting $\mathcal{E}_j^{t_0} = 0$ for all parties and iterations provides an error bound on VFL without compression and is an improvement over the bound in (Liu et al., 2019) in terms of Q, M, and B. The second and third terms include a coefficient relating to local iterations. As the number of local iterations Q increases, the convergence error increases. However, increasing Q also has the effect of reducing the number of global rounds. Thus, it may be beneficial to have $Q > 1$ in practice. We explore this more in experiments in Section 5. The second and third terms scale with M, the number of parties. However, VFL scenarios typically have a small number of parties (Kairouz et al., 2021), and thus M plays a small role in convergence error. We note that when $M = 1$ and $Q = 1$, Theorem 4.2 applies to Split Learning (Gupta & Raskar, 2018) when only uploads to the server are compressed.

Remark 4.3. Let $\mathcal{E} = \frac{1}{R}\sum_{t_0=0}^{R-1}\sum_{m=0}^{M}\mathcal{E}_m^{t_0}$. If $\eta^{t_0} = \frac{1}{\sqrt{T}}$ for all global rounds t_0, for Q and B independent of T, then

$$\frac{1}{R}\sum_{t_0=0}^{R-1}\mathbb{E}\left[\|\nabla F(\Theta^{t_0})\|^2\right] = O\left(\frac{1}{\sqrt{T}}+\mathcal{E}\right).$$

This indicates that if $\mathcal{E} = O(\frac{1}{\sqrt{T}})$ then we can achieve a convergence rate of $O(\frac{1}{\sqrt{T}})$. Informally, this means that C-VFL can afford compression error and not worsen asymptotic convergence when this condition is satisfied. We discuss how this affects commonly used compressors in practice later in the section.

We consider a diminishing step size in the following:

Theorem 4.4. ***Convergence with diminishing step size**: Under Assumptions 1-5, if $0 < \eta^{t_0} < 1$ satisfies $\eta^{t_0} \leq \frac{1}{16Q\max\{L,\max_m L_m\}}$, then the minimum squared gradient over R global rounds of Algorithm 1 is bounded:*

$$\min_{t_0=0,\ldots,R-1}\mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] = O\left(\frac{1}{\sum_{t_0=0}^{R-1}\eta^{t_0}} + \frac{\sum_{t_0=0}^{R-1}(\eta^{t_0})^2}{\sum_{t=0}^{T-1}\eta^{t_0}} + \frac{\sum_{t_0=0}^{R-1}\sum_{m=0}^{M}\eta^{t_0}\mathcal{E}_m^{t_0}}{\sum_{t_0=0}^{R-1}\eta^{t_0}}\right).$$

If η^{t_0} and $\mathcal{E}_m^{t_0}$ satisfy $\sum_{t_0=0}^{\infty}\eta^{t_0} = \infty$, $\sum_{t_0=0}^{\infty}(\eta^{t_0})^2 < \infty$, and $\sum_{t_0=0}^{\infty}\sum_{m=0}^{M}\eta^{t_0}\mathcal{E}_m^{t_0} < \infty$, then $\min_{t_0=0,\ldots,R-1}\mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \to 0$ as $R \to \infty$.

According to Theorem 4.4, the product of the step size and the compression error must be summable over all iterations. In the next subsection, we discuss how to choose common compressor parameters to ensure this property is satisified. We also see in Section 5 that good results can be achieved empirically without diminishing the step size or compression error.

Common Compressors. In this section, we show how to choose common compressor parameters to achieve a convergence rate of $O(\frac{1}{\sqrt{T}})$ in the context of Theorem 4.2, and guarantee convergence in the context of Theorem 4.4. We analyze three common compressors: a uniform scalar quantizer (Bennett, 1948), a 2-dimensional hexagonal lattice quantizer (Zamir & Feder, 1996), and top-k sparsification (Lin et al., 2018). For uniform scalar quantizer, we let there be 2^q quantization levels. For the lattice vector quantizer, we let V be the volume of each lattice cell. For top-k sparsification, we let k be the number of embedding components sent in a message. In Table 1, we present the choice of compressor parameters in order to achieve a convergence

rate of $O(\frac{1}{\sqrt{T}})$ in the context of Theorem 4.2. We show how we calculate these bounds in Appendix B and provide some implementation details for their use. We can also use Table 1 to choose compressor parameters to ensure convergence in the context of Theorem 4.4. Let $\eta^{t_0} = O(\frac{1}{t_0})$, where t_0 is the current round. Then setting $T = t_0$ in Table 1 provides a choice of compression parameters at each iteration to ensure the compression error diminishes at a rate of $O(\frac{1}{\sqrt{t_0}})$, guaranteeing convergence. Diminishing compression error can be achieved by increasing the number of quantization levels, decreasing the volume of lattice cells, or increasing the number of components sent in a message.

5. Experiments

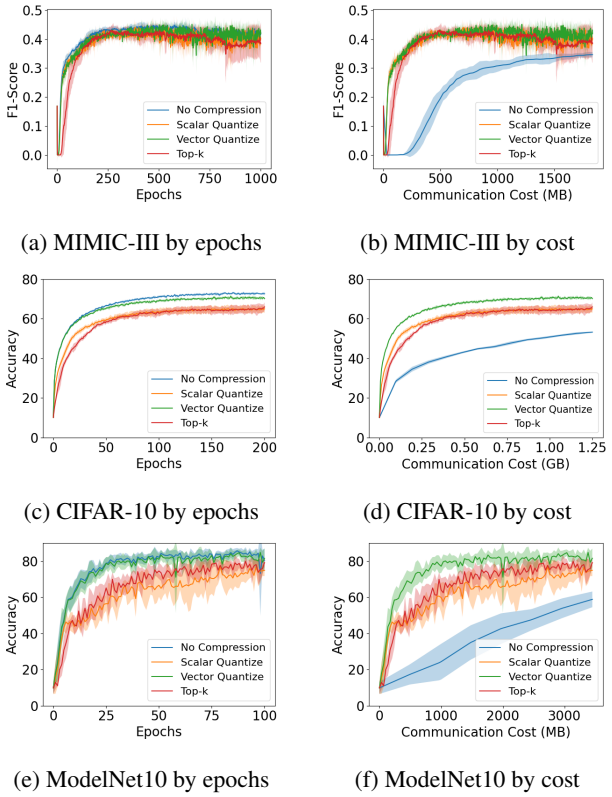
We present experiments to examine the performance of C-VFL in practice. The goal of our experiments is to examine the effects that different compression techniques have on training, and investigate the accuracy/communication trade-off empirically. We run experiments on three datasets: the MIMIC-III dataset (Johnson et al., 2016), the ModelNet10 dataset (Wu et al., 2015), and the CIFAR-10 dataset (Krizhevsky et al., 2009). We provide more details on the datasets and training procedure in Appendix C, as well as additional plots and experiments in Appendix D.

MIMIC-III: MIMIC-III is an anonymized hospital patient time series dataset. In MIMIC-III, the task is binary classification to predict in-hospital mortality. We train with a set of 4 parties, each storing 19 of the 76 features. Each party trains an LSTM and the server trains two fully-connected layers. We use a fixed step size of 0.01, a batch size of 1000, and train for 1000 epochs.

CIFAR-10: CIFAR-10 is an image dataset for object classification. We train with a set of 4 parties, each storing a different quadrant of every image. Each party trains ResNet18, and the server trains a fully-connected layer. We use a fixed step size of 0.0001 and a batch size of 100, and train for 200 epochs.

ModelNet10: ModelNet10 is a set of CAD models, each with images of 12 different camera views. The task of ModelNet10 is classification of images into 10 object classes. We run experiments with both a set of 4 and 12 parties, where parties receive 3 or 1 view(s) of each CAD model, respectively. Each party's network consists of two convolutional layers and a fully-connected layer, and the server model consists of a fully-connected layer. We use a fixed step size of 0.001 and a batch size of 64, and train for 100 epochs.

We consider the three compressors discussed in Section 4: a uniform scalar quantizer, a 2-dimensional hexagonal lattice (vector) quantizer, and top-k sparsification. For both quantizers, the embedding values need to be bounded. In the case of the models used for MIMIC-III and CIFAR-10, the embedding values are already bounded, but the CNN used for ModelNet10 may have unbounded embedding values. We scale embedding values for ModelNet10 to the range $[0, 1]$. We apply subtractive dithering to both the scalar quantizer (Wannamaker, 1997) and vector quantizer (Shlezinger et al., 2021).



(a) MIMIC-III by epochs (b) MIMIC-III by cost

(c) CIFAR-10 by epochs (d) CIFAR-10 by cost

(e) ModelNet10 by epochs (f) ModelNet10 by cost

Figure 3. C-VFL when compressing to 2 bits per component. The solid lines are the mean of 5 runs, while the shaded region represents the standard deviation. We show test F_1-Score on MIMIC-III dataset and test accuracy on CIFAR-10 and ModelNet10 dataset, plotted by epochs and communication cost.

In our experiments, each embedding component is a 32-bit float. Let b be the bits per component we compress to. For the scalar quantizer, this means there are 2^b quantization levels. For the 2-D vector quantizer, this means there are 2^{2b} vectors in the codebook. The volume V of the vector quantizer is a function of the number of codebook vectors. For top-k sparsification, $k = P_m \cdot \frac{b}{32}$ as we use 32-bit components. We train using C-VFL and consider cases where $b = 2$, 3, and 4. We compare with a case where $b = 32$. This corresponds to a standard VFL algorithm without embedding compression (Liu et al., 2019), acting as a baseline for accuracy.

In Figure 3, we plot the test F_1-Score and test accuracy for

Table 2. MIMIC-III maximum F_1-Score reached during training, and communication cost to reach a target test F_1-Score of 0.4. Value shown is the mean of 5 runs, $\pm$ the standard deviation. In these experiments, $Q = 10$ and $M = 4$.

Compressor	Max F_1-Score Reached	Cost (MB) Target $= 0.4$
None $b = 32$	0.448 ± 0.010	3830.0 ± 558.2
Scalar $b = 2$	0.441 ± 0.018	233.1 ± 28.7
Vector $b = 2$	0.451 ± 0.021	236.1 ± 17.9
Top-k $b = 2$	0.431 ± 0.016	309.8 ± 93.6
Scalar $b = 3$	0.446 ± 0.011	343.1 ± 18.8
Vector $b = 3$	0.455 ± 0.020	330.5 ± 10.6
Top-k $b = 3$	0.435 ± 0.030	470.7 ± 116.8
Scalar $b = 4$	0.451 ± 0.020	456.0 ± 87.8
Vector $b = 4$	0.446 ± 0.017	446.5 ± 21.3
Top-k $b = 4$	0.453 ± 0.014	519.1 ± 150.4

Table 3. CIFAR-10 maximum test accuracy reached during training, and communication cost to reach a target accuracy of 70%. Value shown is the mean of 5 runs, $\pm$ the standard deviation. A "–" indicates that the target was not reached during training. We let $Q = 10$ and $M = 4$.

Compressor	Max Accuracy Reached	Cost (GB) Target $= 70\%$
None $b = 32$	$73.18\% \pm 0.44\%$	7.69 ± 0.35
Scalar $b = 2$	$65.16\% \pm 1.85\%$	–
Vector $b = 2$	$71.43\% \pm 0.47\%$	0.68 ± 0.06
Top-k $b = 2$	$66.02\% \pm 2.24\%$	–
Scalar $b = 3$	$71.49\% \pm 1.05\%$	1.22 ± 0.17
Vector $b = 3$	$72.50\% \pm 0.40\%$	0.81 ± 0.05
Top-k $b = 3$	$71.56\% \pm 0.81\%$	1.24 ± 0.22
Scalar $b = 4$	$71.80\% \pm 1.18\%$	1.72 ± 0.26
Vector $b = 4$	$73.17\% \pm 0.39\%$	0.98 ± 0.08
Top-k $b = 4$	$72.03\% \pm 1.77\%$	1.43 ± 0.26

MIMIC-III, CIFAR-10, and ModelNet10 when training with $b = 2$. We use F_1-Score for MIMIC-III as the in-hospital mortality prediction task has high class imbalance; most people in the dataset did not die in the hospital. For these experiments, we let $M = 4$ for ModelNet10. The solid line in each plot represents the average accuracy over five runs, while the shaded regions represent one standard deviation. In Figures 3a, 3c, and 3e, we plot by the number of training epochs. We can see in all cases, although convergence can be a bit slower, training with compressed embeddings still reaches similar accuracy to no compression. In Figures 3b, 3d, and 3f, we plot by the communication cost in bytes. The cost of communication includes both the upload of (compressed) embeddings to the server and download of embeddings and server model to all parties. We can see that by compressing embeddings, we can reach higher accuracy with significantly less communication cost. In all datasets, the compressors reach similar accuracy to each other, though top-k sparsification performs slightly worse than the others on MIMIC-III, while vector quantization performs the best in both on CIFAR-10 and ModelNet10.

In Tables 2, 3 and 4, we show the maximum test accuracy reached during training and the communication cost to reach a target accuracy for MIMIC-III, CIFAR-10, and ModelNet10. We show results for all three compressors with $b = 2$, 3, and 4 bits per component, as well as the baseline of $b = 32$. For the MIMIC-III dataset, we show the maximum test F_1-Score reached and the total communication cost of reaching an F_1-Score of 0.4. The maximum F_1-Score for each case is within a standard deviation of each other. However, the cost to reach target score is much smaller as the value of b decreases for all compressors. We can see that when $b = 2$, we can achieve over 90% communication cost reduction over no compression to reach a target F_1-Score.

Table 4. ModelNet10 maximum test accuracy reached during training, and communication cost to reach a target accuracy of 75%. Value shown is the mean of 5 runs, $\pm$ the standard deviation. We let $Q = 10$ and $M = 4$.

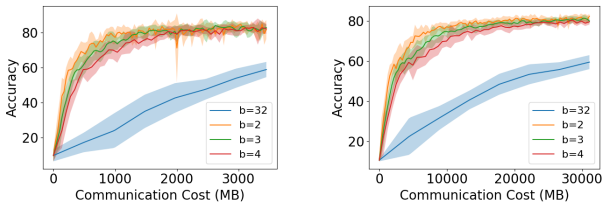
Compressor	Max Accuracy Reached	Cost (MB) Target $= 75\%$
None $b = 32$	$85.68\% \pm 1.57\%$	9604.80 ± 2933.40
Scalar $b = 2$	$76.94\% \pm 5.87\%$	1932.00 ± 674.30
Vector $b = 2$	$84.80\% \pm 2.58\%$	593.40 ± 170.98
Top-k $b = 2$	$79.91\% \pm 2.86\%$	1317.90 ± 222.95
Scalar $b = 3$	$81.32\% \pm 1.61\%$	1738.80 ± 254.79
Vector $b = 3$	$85.66\% \pm 1.36\%$	900.45 ± 275.01
Top-k $b = 3$	$81.63\% \pm 1.24\%$	1593.90 ± 225.34
Scalar $b = 4$	$81.19\% \pm 1.88\%$	2194.20 ± 266.88
Vector $b = 4$	$85.77\% \pm 1.69\%$	1200.60 ± 366.68
Top-k $b = 4$	$83.50\% \pm 1.21\%$	1821.60 ± 241.40

For the CIFAR-10 and ModelNet10 datasets, Tables 3 and 4 show the maximum test accuracy reached and the total communication cost of reaching a target accuracy. We can see that, for both datasets, vector quantization tends to outperform both scalar quantization and top-k quantization. Vector quantization benefits from considering components jointly, and thus can have better reconstruction quality than scalar quantization and top-k sparsification (Woods, 2006).

In Table 5, we consider the communication/computation tradeoff of local iterations. We show how the number of local iterations affects the time to reach a target F_1-Score in the MIMIC-III dataset. We train C-VFL with vector quantization $b = 3$ and set the local iterations Q to 1, 10, and 25. Note that the $Q = 1$ case corresponds to adding embedding compression to previously proposed VFL algorithms that do not have multiple local iterations (Hu et al., 2019; Romanini

Table 5. MIMIC-III time in seconds to reach a target F_1-Score for different local iterations Q and communication latency t_c with vector quantization and $b = 3$. Value shown is the mean of 5 runs, $\pm$ one standard deviation.

t_c	**Time to Reach Target F_1-Score** 0.45		
	$Q = 1$	$Q = 10$	$Q = 25$
1	694.53 $\pm$ 150.75	470.86 $\pm$ 235.35	445.21 $\pm$ 51.44
10	1262.78 $\pm$ 274.10	512.82 $\pm$ 256.32	461.17 $\pm$ 53.29
50	3788.32 $\pm$ 822.30	699.30 $\pm$ 349.53	532.12 $\pm$ 61.49
200	13259.14 $\pm$ 2878.04	1398.60 $\pm$ 699.05	798.19 $\pm$ 92.23



(a) Parties $M = 4$ (b) Parties $M = 12$

Figure 4. Communication cost of training on ModelNet10 with vector quantization. The solid lines are the mean of 5 runs, and the shaded region represents one standard deviation.

et al., 2021). We simulate a scenario where computation time for training a mini-batch of data at each party takes 10 ms, and communication of embeddings takes a total of 1, 10, 50, and 200 ms roundtrip. These different communication latencies correspond to the distance between the parties and the server: within the same cluster, on the same local network, within the same region, and across the globe. According to Theorem 4.2, increasing the number of local iterations Q increases convergence error. However, the target test accuracy is reached within less time when Q increases. The improvement over $Q = 1$ local iterations increases as the communication latency increases. In systems where communication latency is high, it may be beneficial to increase the number of local iterations. The choice of Q will depend on the accuracy requirements of the given prediction task and the time constraints on the prediction problem.

Finally, in Figure 4, we plot the test accuracy of ModelNet10 against the communication cost when using vector quantization with $b = 2$, 3, 4, and 32. We include plots for 4 and 12 parties. We note that changing the number of parties changes the global model structure Θ as well. We can see in both cases that smaller values of b reach higher test accuracies at lower communication cost. The total communication cost is larger with 12 parties, but the impact of increasing compression is similar for both $M = 4$ and $M = 12$.

6. Conclusion

We proposed C-VFL, a distributed communication-efficient algorithm for training a model over vertically partitioned data. We proved convergence of the algorithm at a rate of $O(\frac{1}{\sqrt{T}})$, and we showed experimentally that communication cost could be reduced by over 90% without a significant decrease in accuracy. For future work, we seek to relax our bounded gradient assumption and explore the effect of adaptive compressors.

Acknowledgements

This work was supported by the Rensselaer-IBM AI Research Collaboration (http://airc.rpi.edu), part of the IBM AI Horizons Network (http://ibm.biz/AIHorizons), and the National Science Foundation under grant CNS-1553340.

References

Bennett, W. R. Spectra of quantized signals. *Bell Syst. Tech. J.*, 27(3):446–472, 1948.

Bernstein, J., Wang, Y., Azizzadenesheli, K., and Anandkumar, A. SIGNSGD: compressed optimisation for non-convex problems. *Proc. Int. Conf. on Machine Learn.*, 2018.

Bonawitz, K. A., 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:1611.04482*, 2016.

Bonawitz, K. A., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., Overveldt, T. V., Petrou, D., Ramage, D., and Roselander, J. Towards federated learning at scale: System design. *Proc. of Machine Learn. Sys.*, 2019.

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

Çatak, F. Ö. Secure multi-party computation based privacy preserving extreme learning machine algorithm over vertically distributed data. *Proc. Adv. Neural Inf. Process. Syst.*, 9490:337–345, 2015.

Ceballos, I., Sharma, V., Mugica, E., Singh, A., Roman, A., Vepakomma, P., and Raskar, R. Splitnn-driven vertical partitioning. *arXiv:2008.04137*, 2020.

Cha, D., Sung, M., and Park, Y.-R. Implementing vertical federated learning using autoencoders: Practical application, generalizability, and utility study. *JMIR Medical Informatics*, 9(6):e26598, 2021.

Chen, T., Jin, X., Sun, Y., and Yin, W. VAFL: a method of vertical asynchronous federated learning. *arXiv:2007.06081*, 2020.

Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., and Yang, Q. Secureboost: A lossless federated learning framework. *IEEE Intell. Syst.*, 36(6):87–98, 2021.

Das, A. and Patterson, S. Multi-tier federated learning for vertically partitioned data. *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Process.*, pp. 3100–3104, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009.

Feng, S. and Yu, H. Multi-participant multi-class vertical federated learning. *arXiv:2001.11154*, 2020.

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. doi: 10.1109/TNNLS.2021.3072238.

Gu, Y., Lyu, X., Sun, W., Li, W., Chen, S., Li, X., and Marsic, I. Mutual correlation attentive factors in dyadic fusion networks for speech emotion recognition. *Proc. ACM Int. Conf. on Multimedia*, 2019.

Gupta, O. and Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.*, 116:1–8, 2018.

Han, D.-J., Bhatti, H. I., Lee, J., and Moon, J. Accelerating federated learning with split learning on locally generated losses. In *ICML 2021 Workshop on Federated Learning for User Privacy and Data Confidentiality*, 2021a.

Han, W., Chen, H., and Poria, S. Improving multimodal fusion with hierarchical mutual information maximization for multimodal sentiment analysis. *Proc. 2020 Conf. Empir. Methods in Nat. Lang. Process.*, pp. 9180–9192, 2021b.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., and Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv:1711.10677*, 2017.

He, C., Annavaram, M., and Avestimehr, S. Group knowledge transfer: Federated learning of large cnns at the edge. *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

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.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. MIMIC-III, a freely accessible critical care database. *Nature*, 2016.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K. A., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021. doi: 10.1561/2200000083.

Karimireddy, S. P., Rebjock, Q., Stich, S. U., and Jaggi, M. Error feedback fixes signSGD and other gradient compression schemes. *Proc. Int. Conf. on Machine Learn.*, 2019.

Koehrsen, W. Book recommendation system. https://github.com/WillKoehrsen/wikipedia-data-science/blob/master/notebooks/Book2018.

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

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proc. of Machine Learn. Sys.*, 2020.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surveys Tuts.*, 2020.

Lin, T., Stich, S. U., Patel, K. K., and Jaggi, M. Don't use large mini-batches, use local SGD. *Proc. Int. Conf. on Learn. Representations*, 2020.

Lin, Y., Han, S., Mao, H., Wang, Y., and Dally, B. Deep gradient compression: Reducing the communication bandwidth for distributed training. *Proc. Int. Conf. on Learn. Representations*, 2018.

Liu, L., Zhang, J., Song, S., and Letaief, K. B. Client-edge-cloud hierarchical federated learning. *Proc. IEEE Int. Conf. on Comm.*, 2020.

Liu, Y., Kang, Y., Zhang, X., Li, L., Cheng, Y., Chen, T., Hong, M., and Yang, Q. A communication efficient vertical federated learning framework. *Adv. Neural Inf. Process. Syst., Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

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.

Moritz, P., Nishihara, R., Stoica, I., and Jordan, M. I. Sparknet: Training deep networks in spark. *Proc. Int. Conf. on Learn. Representations*, 2016.

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.

Nie, W., Liang, Q., Wang, Y., Wei, X., and Su, Y. MMFN: multimodal information fusion networks for 3d model classification and retrieval. *ACM Trans. on Multimedia Computing, Communications, and Applications*, 2021.

Phong, L. T., Aono, Y., Hayashi, T., Wang, L., and Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Trans. Inf. Forensics Security*, 13(5):1333–1345, 2018.

Richtárik, P. and Takác, M. Parallel coordinate descent methods for big data optimization. *Math. Program.*, 156 (1-2):433–484, 2016.

Rieke, N., Hancox, J., Li, W., Milletarì, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., Ourselin, S., Sheller, M., Summers, R. M., Trask, A., Xu, D., Baust, M., and Cardoso, M. J. *Digital Medicine*, 2020.

Romanini, D., Hall, A. J., Papadopoulos, P., Titcombe, T., Ismail, A., Cebere, T., Sandmann, R., Roehm, R., and Hoeh, M. A. PyVertical: A vertical federated learning framework for multi-headed SplitNN. *Int. Conf. Learn. Representations, Workshop on Distributed and Private Machine Learn.*, 2021.

Shi, S., Zhao, K., Wang, Q., Tang, Z., and Chu, X. A convergence analysis of distributed SGD with communication-efficient gradient sparsification. *Proc. Int. Joint Conf. on Artif. Intell.*, 2019.

Shlezinger, N., Chen, M., Eldar, Y. C., Poor, H. V., and Cui, S. Uveqfed: Universal vector quantization for federated learning. *IEEE Trans. Signal Process.*, 69:500–514, 2021.

Stich, S. U., Cordonnier, J., and Jaggi, M. Sparsified SGD with memory. *Adv. Neural Inf. Process. Syst.*, 2018.

Tsitsiklis, J., Bertsekas, D., and Athans, M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. Autom. Control*, 31(9): 803–812, 1986.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.*, 37(6):1205–1221, 2019.

Wannamaker, R. A. *The Theory of Dithered Quantization*. PhD thesis, 1997.

Wen, W., Xu, C., Yan, F., Wu, C., Wang, Y., Chen, Y., and Li, H. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Adv. Neural Inf. Process. Syst.*, 2017.

Woods, J. W. *Multidimensional signal, image, and video processing and coding*. Elsevier, 2006.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3D shapenets: A deep representation for volumetric shapes. *Proc. IEEE Int. Conf. Comput. Vis.*, pp. 1912–1920, 2015.

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.

Zamir, R. and Feder, M. On lattice quantization noise. *IEEE Trans. Inf. Theory*, 42(4):1152–1159, 1996.

Zhang, X., Yin, W., Hong, M., and Chen, T. Hybrid federated learning: Algorithms and implementation. *arXiv:2012.12420*, 2020.

Zheng, F., Chen, C., Zheng, X., and Zhu, M. Towards secure and practical machine learning via secret sharing and random permutation. *Knowl. Based Syst.*, 245:108609, 2022.

A. Proofs of Theorems 4.2 and 4.4

In this section, we provide the proofs for Theorems 4.2 and 4.4.

A.1. Additional Notation

Before starting the proofs, we define some additional notation to be used throughout. At each iteration t, each party m trains with the embeddings $\hat{\Phi}_m^t$. This is equivalent to the party training directly with the models θ_m^t and $\theta_j^{t_0}$ for all $j \neq m$, where t_0 is the last communication iteration when party m received the embeddings. We define:

$$\gamma_{m,j}^t = \begin{cases} \theta_j^t & m = j \\ \theta_j^{t_0} & \text{otherwise} \end{cases} \tag{A.1}$$

to represent party m's view of party j's model at iteration t. We define the column vector $\Gamma_m^t = [(\gamma_{m,0}^t)^T; \ldots; (\gamma_{m,M}^t)^T]^T$ to be party m's view of the global model at iteration t.

We introduce some notation to help with bounding the error introduced by compression. We define $\hat{F}_{\mathbf{B}}(\Gamma_m^t)$ to be the stochastic loss with compression error for a randomly selected mini-batch $\mathbf{B}$ calculated by party m at iteration t:

$$\hat{F}_{\mathbf{B}}(\Gamma_m^t) \coloneqq F_{\mathbf{B}}\left(\theta_0^{t_0} + \epsilon_0^{t_0}, h_1(\theta_1^{t_0}; \mathbf{X}_1^{\mathbf{B}^{t_0}}) + \epsilon_1^{t_0}, \ldots, h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}}), \ldots, h_M(\theta_M^{t_0}; \mathbf{X}_M^{\mathbf{B}^{t_0}}) + \epsilon_M^{t_0}\right). \tag{A.2}$$

Recall the recursion over the global model Θ:

$$\Theta^{t+1} = \Theta^t - \eta^{t_0} \hat{\mathbf{G}}^t. \tag{A.3}$$

We can equivalently define $\hat{\mathbf{G}}^t$ as follows:

$$\hat{\mathbf{G}}^t = \left[(\nabla_0 \hat{F}_{\mathbf{B}}(\Gamma_0^t))^T, \ldots, (\nabla_M \hat{F}_{\mathbf{B}}(\Gamma_M^t))^T\right]^T. \tag{A.4}$$

Note that the compression error in $\hat{F}(\cdot)$ is applied to the embeddings, and not the model parameters. Thus, $F(\cdot)$ and $\hat{F}(\cdot)$ are different functions. In several parts of the proof, we need to bound the compression error in $\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t)$.

For our analysis, we redefine the set of embeddings for a mini-batch $\mathbf{B}$ of size B from party m as a matrix:

$$h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}}) \coloneqq \left[h_m(\theta_m; x_m^{\mathbf{B}^1}), \ldots, h_m(\theta_m; x_m^{\mathbf{B}^B})\right]. \tag{A.5}$$

$h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}})$ is a matrix with dimensions $P_m \times B$ where each column is the embedding from party m for a single sample in the mini-batch.

Let $P = \sum_{m=0}^M P_m$ be the sum of the sizes of all embeddings. We redefine the set of embeddings used by a party m to calculate its gradient without compression error as a matrix:

$$\hat{\Phi}_m^t = \left[(\theta_0^{t_0})^T, (h_1(\theta_1^{t_0}; \mathbf{X}_1^{\mathbf{B}^{t_0}}))^T, \ldots, (h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}}))^T, \ldots, (h_M(\theta_M^{t_0}; \mathbf{X}_M^{\mathbf{B}^{t_0}}))^T\right]^T. \tag{A.6}$$

$\hat{\Phi}_m^t$ is a matrix with dimensions $P \times B$ where each column is the concatenation of embeddings for all parties for a single sample in the mini-batch.

Recall the set of compression error vectors for a mini-batch $\mathbf{B}$ of size B from party m is the matrix:

$$\epsilon_m^{t_0} \coloneqq \left[\epsilon_m^{\mathbf{B}^1}, \ldots, \epsilon_m^{\mathbf{B}^B}\right]. \tag{A.7}$$

$\epsilon_m^{t_0}$ is a matrix of dimensions $P_m \times B$ where each column is the compression error from party m for a single sample in the mini-batch.

We define the compression error on each embedding used in party m's gradient calculation at iteration t:

$$E_m^{t_0} = \left[(\epsilon_0^{t_0})^T, \ldots, (\epsilon_{m-1}^{t_0})^T, \mathbf{0}^T, (\epsilon_{m-1}^{t_0})^T, \ldots, (\epsilon_M^{t_0})^T\right]^T. \tag{A.8}$$

$E_m^{t_0}$ is a matrix with dimensions $P \times B$ where each column is the concatenation of compression error on embeddings for all parties for a single sample in the mini-batch.

With some abuse of notation, we define:

$$\nabla_m F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0}) \coloneqq \nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t). \tag{A.9}$$

Note that we can apply the chain rule to $\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t)$:

$$\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t) = \nabla_{\theta_m} h_m(\theta_m^t) \nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0}). \tag{A.10}$$

With this expansion, we can now apply Taylor series expansion to $\nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0})$ around the point Φ_m^t:

$$\nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0}) = \nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t) + \nabla^2_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t)^t E_m^{t_0} + \ldots \tag{A.11}$$

We let the infinite sum of all terms in this Taylor series from the second partial derivatives and up be denoted as R_0^m:

$$R_0^m(\Phi_m^t + E_m^{t_0}) \coloneqq \nabla^2_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t)^T E_m^{t_0} + \ldots \tag{A.12}$$

Note that all compression error is in $R_0^m(\Phi_m^t + E_m^{t_0})$. Presented in Section A.2, the proof of Lemma 1' shows how we can bound $R_0^m(\Phi_m^t + E_m^{t_0})$, bounding the compression error in $\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t)$.

Let $\mathbb{E}^{t_0} = \mathbb{E}_{\mathbf{B}^{t_0}}[\ \cdot\ |\ \{\Theta^\tau\}_{\tau=0}^{t_0}]$. Note that by Assumption 2, $\mathbb{E}^{t_0}\left[\mathbf{G}^{t_0}\right] = \nabla F(\Theta^{t_0})$ as when there is no compression error in the gradients $\mathbf{G}$, they are equal to the full-batch gradient in expectation when conditioned on the model parameters up to the iteration t_0. However, this is not true for iterations $t_0 + 1 \leq t \leq t_0 + Q - 1$, as we reuse the mini-batch $\mathbf{B}^{t_0}$ in these local iterations. We upper bound the error introduced by stochastic gradients calculated during local iterations in Lemma 2.

A.2. Supporting Lemmas

Next, we provide supporting lemmas and their proofs.

We restate Lemma 1 here:

Lemma 1. Under Assumptions 4-5, the norm of the difference between the objective function value with and without error is bounded:

$$\mathbb{E}\left\|\nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t) - \nabla_m F_{\mathbf{B}}(\Phi_m^t)\right\|^2 \leq H_m^2 G_m^2 \sum_{j=0, j\neq m}^{M} \mathcal{E}_j^{t_0}. \tag{A.13}$$

To prove Lemma 1, we first prove the following lemma:

Lemma 1'. Under Assumptions 4-5, the squared norm of the partial derivatives for party m's embedding multiplied by the Taylor series terms $R_0^m(\Phi_m^t + E_m^{t_0})$ is bounded:

$$\left\|\nabla_{\theta_m} h_m(\theta_m^t) R_0^m(\Phi_m^t + E_m^{t_0})\right\|^2 \leq H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}. \tag{A.14}$$

Proof.

$$\left\|\nabla_{\theta_m} h_m(\theta_m^t) R_0^m(\Phi_m^t + E_m^{t_0})\right\|^2 \leq \left\|\nabla_{\theta_m} h_m(\theta_m^t)\right\|_{\mathcal{F}}^2 \left\|R_0^m(\Phi_m^t + E_m^{t_0})\right\|_{\mathcal{F}}^2 \tag{A.15}$$

$$\leq H_m^2 \left\|\nabla_{\theta_m} h_m(\theta_m^t)\right\|_{\mathcal{F}}^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.16}$$

where (A.16) follows from Assumption 4 and the following property of the Taylor series approximation error:

$$\left\|R_0^m(\Phi_m^t + E_m^{t_0})\right\|_{\mathcal{F}} \leq H_m \left\|E_m^{t_0}\right\|_{\mathcal{F}}. \tag{A.17}$$

Applying Assumption 5, we have:

$$\left\|\nabla_{\theta_m} h_m(\theta_m^t) R_0^m(\Phi_m^t + E_m^{t_0})\right\|^2 \leq H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2. \tag{A.18}$$

□

We now prove Lemma 1.

Proof. Recall that:

$$\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t) = \nabla_m F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0}) \tag{A.19}$$
$$= \nabla_{\theta_m} h_m(\theta_m^t) \nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t + E_m^{t_0}). \tag{A.20}$$

Next we apply Taylor series expansion as in (A.11):

$$\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t) = \nabla_{\theta_m} h_m(\theta_m^t) \left(\nabla_{h_m(\theta_m)} F_{\mathbf{B}}(\Phi_m^t) + R_0^m(\Phi_m^t + E_m^{t_0})\right) \tag{A.21}$$
$$= \nabla_m F_{\mathbf{B}}(\Gamma_m^t) + \nabla_{\theta_m} h_m(\theta_m^t) R_0^m(\Phi_m^t + E_m^{t_0}) \tag{A.22}$$

Rearranging and applying expectation and the squared 2-norm, we can bound further:

$$\mathbb{E}\left\|\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^t) - \nabla_m F_{\mathbf{B}}(\Gamma_m^t)\right\|^2 = \mathbb{E}\left\|\nabla_{\theta_m} h_m(\theta_m^t) R_0^m(\Phi_m^t + E_m^{t_0})\right\|^2 \tag{A.23}$$
$$\leq H_m^2 G_m^2 \mathbb{E}\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.24}$$
$$= H_m^2 G_m^2 \sum_{j \neq m} \mathbb{E}\left\|\epsilon_j^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.25}$$
$$= H_m^2 G_m^2 \sum_{j \neq m} \mathcal{E}_j^{t_0} \tag{A.26}$$

where (A.24) follows from Lemma 1', (A.25) follows from the definition of $E_m^{t_0}$, and (A.26) follows from Definition 4.1. □

Lemma 2. If $\eta^{t_0} \leq \frac{1}{4Q \max_m L_m}$, then under Assumptions 1-5 we can bound the conditional expected squared norm difference of gradients $\mathbf{G}^{t_0}$ and $\hat{\mathbf{G}}^t$ for iterations t_0 to $t_0 + Q - 1$ as follows:

$$\sum_{t=t_0}^{t_0+Q-1} \mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t - \mathbf{G}^{t_0}\right\|^2\right] \leq 16Q^3(\eta^{t_0})^2 \sum_{m=0}^{M} L_m^2 \left\|\nabla_m F(\Theta^{t_0})\right\|^2$$
$$+ 16Q^3(\eta^{t_0})^2 \sum_{m=0}^{M} L_m^2 \frac{\sigma_m^2}{B}$$
$$+ 64Q^3 \sum_{m=0}^{M} H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2. \tag{A.27}$$

Proof.

$$\mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t-\mathbf{G}^{t_0}\right\|^2\right]=\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^t)-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.28}$$

$$=\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^t)-\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})+\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.29}$$

$$\leq(1+n)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^t)-\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})\right\|^2\right]$$
$$+\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.30}$$

$$\leq 2(1+n)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^t)-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t-1})\right\|^2\right]$$
$$+2(1+n)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_{\theta_m}h_m(\theta_m^t)R_0^m(\Phi_m^t+E_m^{t_0})-\nabla_{\theta_m}h_m(\theta_m^{t-1})R_0^m(\Phi_m^{t-1}+E_m^{t-1})\right\|^2\right]$$
$$+\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.31}$$

$$\leq 2(1+n)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^t)-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t-1})\right\|^2\right]+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2$$
$$+\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.32}$$

where (A.30) follows from the fact that $(X+Y)^2\leq(1+n)X^2+(1+\frac{1}{n})Y^2$ for some positive n and (A.32) follows from Lemma 1'.

Applying Assumption 1 to the first term in (A.30) we have:

$$\mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t-\mathbf{G}^{t_0}\right\|^2\right]\leq 2(1+n)\sum_{m=0}^{M}L_m^2\mathbb{E}^{t_0}\left[\left\|\Gamma_m^t-\Gamma_m^{t-1}\right\|^2\right]$$
$$+2\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2 \tag{A.33}$$

$$=2(\eta^{t_0})^2(1+n)\sum_{m=0}^{M}L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})\right\|^2\right]$$
$$+2\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2 \tag{A.34}$$

where (A.34) follows from the update rule $\Gamma_m^t=\Gamma_m^{t-1}-\eta^{t_0}\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})$.

Bounding further:

$$\mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t-\mathbf{G}^{t_0}\right\|^2\right] \leq 2(\eta^{t_0})^2(1+n)\sum_{m=0}^{M} L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})+\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2 \tag{A.35}$$
$$\leq 4(\eta^{t_0})^2(1+n)\sum_{m=0}^{M} L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+4(\eta^{t_0})^2(1+n)\sum_{m=0}^{M} L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+\left(1+\frac{1}{n}\right)\sum_{m=0}^{M}\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2 \tag{A.36}$$
$$=\sum_{m=0}^{M}\left(4(\eta^{t_0})^2(1+n)L_m^2+\left(1+\frac{1}{n}\right)\right)\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+4(\eta^{t_0})^2(1+n)\sum_{m=0}^{M} L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+n)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|^2. \tag{A.37}$$

Let $n=Q$. We simplify (A.37) further:

$$\mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t-\mathbf{G}^{t_0}\right\|^2\right]$$
$$\leq\sum_{m=0}^{M}\left(4(\eta^{t_0})^2(1+Q)L_m^2+\left(1+\frac{1}{Q}\right)\right)\mathbb{E}^{t_0}\left[\left\|\nabla_m\hat{F}_{\mathbf{B}}(\Gamma_m^{t-1})-\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+4(\eta^{t_0})^2(1+Q)\sum_{m=0}^{M} L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+8(1+Q)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2. \tag{A.38}$$

Let $\eta^{t_0} \leq \frac{1}{4Q \max_m L_m}$. We bound (A.38) as follows:

$$\mathbb{E}^{t_0}\left[\left\|\hat{\mathbf{G}}^t - \mathbf{G}^{t_0}\right\|^2\right] \leq \left(\frac{(1+Q)}{4Q^2} + \left(1+\frac{1}{Q}\right)\right) \sum_{m=0}^{M} \mathbb{E}^{t_0}\left[\left\|\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^{t-1}) - \nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 4(\eta^{t_0})^2 (1+Q) \sum_{m=0}^{M} L_m^2 \mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 8(1+Q) \sum_{m=0}^{M} H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.39}$$
$$\leq \left(\frac{1}{2Q} + \left(1+\frac{1}{Q}\right)\right) \sum_{m=0}^{M} \mathbb{E}^{t_0}\left[\left\|\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^{t-1}) - \nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 4(\eta^{t_0})^2 (1+Q) \sum_{m=0}^{M} L_m^2 \mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 8(1+Q) \sum_{m=0}^{M} H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.40}$$
$$\leq \left(1+\frac{2}{Q}\right) \sum_{m=0}^{M} \mathbb{E}^{t_0}\left[\left\|\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^{t-1}) - \nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 4(\eta^{t_0})^2 (1+Q) \sum_{m=0}^{M} L_m^2 \mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right]$$
$$+ 8(1+Q) \sum_{m=0}^{M} H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 . \tag{A.41}$$

We define the following notation for simplicity:

$$A^t \coloneqq \sum_{m=0}^{M} \mathbb{E}^{t_0}\left[\left\|\nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^{t}) - \nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.42}$$
$$B_0 \coloneqq 4(\eta^{t_0})^2 (1+Q) \sum_{m=0}^{M} L_m^2 \mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] \tag{A.43}$$
$$B_1 \coloneqq 8(1+Q) \sum_{m=0}^{M} H_m^2 G_m^2 \left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.44}$$
$$C \coloneqq \left(1+\frac{2}{Q}\right). \tag{A.45}$$

Note that we have shown that $A^t \leq CA^{t-1} + B_0 + B_1$. Therefore:

$$A^{t_0+1} \leq CA^{t_0} + (B_0 + B_1) \tag{A.46}$$
$$A^{t_0+2} \leq C^2 A^{t_0} + C(B_0 + B_1) + (B_0 + B_1) \tag{A.47}$$
$$A^{t_0+3} \leq C^3 A^{t_0} + C^2(B_0 + B_1) + C(B_0 + B_1) + (B_0 + B_1) \tag{A.48}$$
$$\vdots \tag{A.49}$$
$$A^t \leq C^{t-t_0-1} A^{t_0} + (B_0 + B_1) \sum_{k=0}^{t-t_0-2} C^k \tag{A.50}$$
$$= C^{t-t_0-1} A^{t_0} + (B_0 + B_1) \frac{C^{t-t_0-1} - 1}{C - 1}. \tag{A.51}$$

We bound the first term in (A.51) by applying Lemma 1:

$$A^{t_0} = \sum_{m=0}^{M} \mathbb{E}^{t_0} \left[\left\| \nabla_m \hat{F}_{\mathbf{B}}(\Gamma_m^{t_0}) - \nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0}) \right\|^2 \right] \tag{A.52}$$

$$\leq \sum_{m=0}^{M} H_m^2 G_m^2 \left\| E_m^{t_0} \right\|_{\mathcal{F}}^2 . \tag{A.53}$$

Summing over the set of local iterations $t_0, \ldots, t_0^+$, where $t_0^+ \coloneqq t_0 + Q - 1$:

$$\sum_{t=t_0}^{t_0^+} C^{t-t_0-1} A^{t_0} = A^{t_0} \sum_{t=t_0}^{t_0^+} C^{t-t_0-1} \tag{A.54}$$

$$= A^{t_0} \frac{C^Q - 1}{C - 1} \tag{A.55}$$

$$= A^{t_0} \frac{\left(1 + \frac{2}{Q}\right)^Q - 1}{\left(1 + \frac{2}{Q}\right) - 1} \tag{A.56}$$

$$\leq Q A^{t_0} \frac{e^2 - 1}{2} \tag{A.57}$$

$$\leq 4Q A^{t_0} \tag{A.58}$$

$$\leq 4Q \sum_{m=0}^{M} H_m^2 G_m^2 \left\| E_m^{t_0} \right\|_{\mathcal{F}}^2 . \tag{A.59}$$

It is left to bound the second term in (A.51) over the set of local iterations $t_0, \ldots, t_0 + Q - 1$.

$$\sum_{t=t_0}^{t_0^+} (B_0 + B_1) \frac{C^{t-t_0-1} - 1}{C - 1} \leq \sum_{t=t_0}^{t_0^+} (B_0 + B_1) \frac{C^{t-t_0-1} - 1}{C - 1} \tag{A.60}$$

$$= \frac{(B_0 + B_1)}{C - 1} \left(\sum_{t=t_0}^{t_0^+} C^{t-t_0-1} - Q \right) \tag{A.61}$$

$$= \frac{(B_0 + B_1)}{C - 1} \left(\frac{C^Q - 1}{C - 1} - Q \right) \tag{A.62}$$

$$= \frac{(B_0 + B_1)}{\left(1 + \frac{2}{Q}\right) - 1} \left(\frac{\left(1 + \frac{2}{Q}\right)^Q - 1}{\left(1 + \frac{2}{Q}\right) - 1} - Q \right) \tag{A.63}$$

$$= \frac{Q(B_0 + B_1)}{2} \left(\frac{Q \left[\left(1 + \frac{2}{Q}\right)^Q - 1 \right]}{2} - Q \right) \tag{A.64}$$

$$= \frac{Q^2(B_0 + B_1)}{2} \left(\frac{\left(1 + \frac{2}{Q}\right)^Q - 1}{2} - 1 \right) \tag{A.65}$$

$$\leq \frac{Q^2(B_0 + B_1)}{2} \left(\frac{e^2 - 1}{2} - 1 \right) \tag{A.66}$$

$$\leq 2Q^2(B_0 + B_1) \tag{A.67}$$

Plugging the values for B_0 and B_1:

$$\sum_{t=t_0}^{t_0^+}(B_0+B_1)\frac{C^{t-t_0-1}-1}{C-1} \leq 8Q^2(\eta^{t_0})^2(1+Q)\sum_{m=0}^{M}L_m^2\mathbb{E}^{t_0}\left[\left\|\nabla_m F_{\mathbf{B}}(\Gamma_m^{t_0})\right\|^2\right] + 16Q^2(1+Q)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.68}$$

Applying Assumption 3 and adding in the first term in (A.51):

$$\begin{aligned}\sum_{t=t_0}^{t_0^+}A^t &\leq 8Q^2(\eta^{t_0})^2(1+Q)\sum_{m=0}^{M}L_m^2\left\|\nabla_m F(\Theta^{t_0})\right\|^2 \\ &\quad + 8Q^2(\eta^{t_0})^2(1+Q)\sum_{m=0}^{M}L_m^2\frac{\sigma_m^2}{B} \\ &\quad + 4(4Q^2(1+Q)+Q)\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \qquad \text{(A.69)} \\ &\leq 16Q^3(\eta^{t_0})^2\sum_{m=0}^{M}L_m^2\left\|\nabla_m F(\Theta^{t_0})\right\|^2 \\ &\quad + 16Q^3(\eta^{t_0})^2\sum_{m=0}^{M}L_m^2\frac{\sigma_m^2}{B} \\ &\quad + 64Q^3\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2. \qquad \text{(A.70)}\end{aligned}$$

□

A.3. Proof of Theorems 4.2 and 4.4

Let $t_0^+ \coloneqq t_0+Q-1$. By Assumption 1:

$$F(\Theta^{t_0^+})-F(\Theta^{t_0}) \leq \left\langle\nabla F(\Theta^{t_0}),\Theta^{t_0^+}-\Theta^{t_0}\right\rangle+\frac{L}{2}\left\|\Theta^{t_0^+}-\Theta^{t_0}\right\|^2 \tag{A.71}$$

$$= -\left\langle\nabla F(\Theta^{t_0}),\sum_{t=t_0}^{t_0^+}\eta^{t_0}\hat{\mathbf{G}}^t\right\rangle+\frac{L}{2}\left\|\sum_{t=t_0}^{t_0^+}\eta^{t_0}\hat{\mathbf{G}}^t\right\|^2 \tag{A.72}$$

$$\leq -\sum_{t=t_0}^{t_0^+}\eta^{t_0}\left\langle\nabla F(\Theta^{t_0}),\hat{\mathbf{G}}^t\right\rangle+\frac{LQ}{2}\sum_{t=t_0}^{t_0^+}(\eta^{t_0})^2\left\|\hat{\mathbf{G}}^t\right\|^2 \tag{A.73}$$

where (A.73) follows from fact that $(\sum_{n=1}^{N}x_n)^2 \leq N\sum_{n=1}^{N}x_n^2$.

We bound further:

$$F(\Theta^{t_0^+}) - F(\Theta^{t_0}) \leq -\sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\rangle - \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \mathbf{G}^{t_0} \right\rangle + \frac{LQ}{2} \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} + \mathbf{G}^{t_0} \right\|^2 \tag{A.74}$$

$$\leq -\sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\rangle - \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \mathbf{G}^{t_0} \right\rangle + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \mathbf{G}^{t_0} \right\|^2 \tag{A.75}$$

$$= \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle -\nabla F(\Theta^{t_0}), \mathbf{G}^{t_0} - \hat{\mathbf{G}}^t \right\rangle - \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \mathbf{G}^{t_0} \right\rangle + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \mathbf{G}^{t_0} \right\|^2. \tag{A.76}$$

$$\leq \frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\| \nabla F(\Theta^{t_0}) \right\|^2 + \frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 - \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\langle \nabla F(\Theta^{t_0}), \mathbf{G}^{t_0} \right\rangle + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \left\| \mathbf{G}^{t_0} \right\|^2 \tag{A.77}$$

where (A.77) follows from the fact that $A \cdot B = \frac{1}{2}A^2 + \frac{1}{2}B^2 - \frac{1}{2}(A-B)^2$.

We apply the expectation $\mathbb{E}^{t_0}$ to both sides of (A.77):

$$\mathbb{E}^{t_0}\left[F(\Theta^{t_0^+})\right] - F(\Theta^{t_0}) \leq -\frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} \left\| \nabla F(\Theta^{t_0}) \right\|^2 + \frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} (1 + 2LQ\eta^{t_0}) \mathbb{E}^{t_0}\left[\left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 \right] + LQ \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \mathbb{E}^{t_0}\left[\left\| \mathbf{G}^{t_0} \right\|^2 \right] \tag{A.78}$$

$$\leq -\frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} (1 - 2LQ\eta^{t_0}) \left\| \nabla F(\Theta^{t_0}) \right\|^2 + \frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} (1 + 2LQ\eta^{t_0}) \mathbb{E}^{t_0}\left[\left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 \right] + LQ \sum_{m=0}^{M} \frac{\sigma_m^2}{B} \sum_{t=t_0}^{t_0^+} (\eta^{t_0})^2 \tag{A.79}$$

$$= -\frac{Q}{2} \eta^{t_0} (1 - 2LQ\eta^{t_0}) \left\| \nabla F(\Theta^{t_0}) \right\|^2 + \frac{1}{2} \sum_{t=t_0}^{t_0^+} \eta^{t_0} (1 + 2LQ\eta^{t_0}) \mathbb{E}^{t_0}\left[\left\| \hat{\mathbf{G}}^t - \mathbf{G}^{t_0} \right\|^2 \right] + LQ^2 (\eta^{t_0})^2 \sum_{m=0}^{M} \frac{\sigma_m^2}{B} \tag{A.80}$$

where (A.78) follows from applying Assumption 2 and noting that $\mathbb{E}^{t_0}\left[\mathbf{G}^{t_0}\right] = \nabla F(\Theta^{t_0})$, and (A.80) follows from Assumption 3.

Applying Lemma 2 to (A.80):

$$\begin{aligned}\mathbb{E}^{t_0}\left[F(\Theta^{t_0^+})\right] - F(\Theta^{t_0}) \leq & -\frac{Q}{2}\eta^{t_0}(1-2LQ\eta^{t_0})\left\|\nabla F(\Theta^{t_0})\right\|^2 \\ & + 8Q^3(\eta^{t_0})^3(1+2LQ\eta^{t_0})\sum_{m=0}^{M} L_m^2\left\|\nabla_m F(\Theta_m^{t_0})\right\|^2 \\ & + 8Q^3(\eta^{t_0})^3(1+2LQ\eta^{t_0})\sum_{m=0}^{M} L_m^2\frac{\sigma_m^2}{B} \\ & + 32Q^3\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M} H_m^2 G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \\ & + LQ^2(\eta^{t_0})^2\sum_{m=0}^{M}\frac{\sigma_m^2}{B} \end{aligned} \tag{A.81}$$

$$\begin{aligned}\leq & -\frac{Q}{2}\sum_{m=0}^{M}\eta^{t_0}(1-2LQ\eta^{t_0}-16Q^2L_m^2(\eta^{t_0})^2-16Q^3L_m^2L(\eta^{t_0})^3))\left\|\nabla_m F(\Theta^{t_0})\right\|^2 \\ & + (LQ^2(\eta^{t_0})^2+8Q^3L_m^2(\eta^{t_0})^3+8Q^4LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} \\ & + 32Q^3\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M} H_m^2 G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2. \end{aligned} \tag{A.82}$$

Let $\eta^{t_0} \leq \frac{1}{16Q\max\{L,\max_m L_m\}}$. Then we bound (A.82) further:

$$\begin{aligned}\mathbb{E}^{t_0}\left[F(\Theta^{t_0^+})\right] - F(\Theta^{t_0}) \leq & -\frac{Q}{2}\sum_{m=0}^{M}\eta^{t_0}\left(1-\frac{1}{8}-\frac{1}{16}-\frac{1}{16^2}\right)\left\|\nabla_m F(\Theta^{t_0})\right\|^2 \\ & + (LQ^2(\eta^{t_0})^2+8Q^3L_m^2(\eta^{t_0})^3+8Q^4LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} \\ & + 16Q^3\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M} H_m^2 G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \end{aligned} \tag{A.83}$$

$$\begin{aligned}\leq & -\frac{3Q}{8}\eta^{t_0}\left\|\nabla F(\Theta^{t_0})\right\|^2 \\ & + (LQ^2(\eta^{t_0})^2+8Q^3L_m^2(\eta^{t_0})^3+8Q^4LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} \\ & + 32Q^3\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M} H_m^2 G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \end{aligned} \tag{A.84}$$

After some rearranging of terms:

$$\begin{aligned}\eta^{t_0}\left\|\nabla F(\Theta^{t_0})\right\|^2 \leq & \frac{4\left[F(\Theta^{t_0})-\mathbb{E}^{t_0}\left[F(\Theta^{t_0^+})\right]\right]}{Q} \\ & + \frac{8}{3}(LQ(\eta^{t_0})^2+8Q^2L_m^2(\eta^{t_0})^3+8Q^3LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} \\ & + 86Q^2\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M} H_m^2 G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \end{aligned} \tag{A.85}$$

Summing over all global rounds $t_0 = 0, \ldots, R-1$ and taking total expectation:

$$\sum_{t_0=0}^{R-1} \eta^{t_0} \mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \leq \frac{4\left[F(\Theta^0) - \mathbb{E}\left[F(\Theta^T)\right]\right]}{Q} + \frac{8}{3}\sum_{t_0=0}^{R-1}(LQ(\eta^{t_0})^2 + 8Q^2L_m^2(\eta^{t_0})^3 + 8Q^3LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} + 86Q^2\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M}H_m^2G_m^2\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2 \tag{A.86}$$

$$\leq \frac{4\left[F(\Theta^0) - \mathbb{E}\left[F(\Theta^T)\right]\right]}{QR} + \frac{8}{3}\sum_{t_0=0}^{R-1}(QL(\eta^{t_0})^2 + 8Q^2L_m^2(\eta^{t_0})^3 + 8Q^3LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} + 86Q^2\sum_{t_0=0}^{R-1}\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M}H_m^2G_m^2\mathbb{E}\left[\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2\right]. \tag{A.87}$$

Note that:

$$\sum_{m=0}^{M}H_m^2G_m^2\mathbb{E}\left[\left\|E_m^{t_0}\right\|_{\mathcal{F}}^2\right] = \sum_{m=0}^{M}H_m^2G_m^2\sum_{j\neq m}\mathbb{E}\left[\left\|\epsilon_j^{t_0}\right\|_{\mathcal{F}}^2\right] \tag{A.88}$$

$$= \sum_{m=0}^{M}H_m^2G_m^2\sum_{j\neq m}\mathcal{E}_j^{t_0} \tag{A.89}$$

where (A.89) follows from Definition 4.1.

Plugging this into (A.87)

$$\sum_{t_0=0}^{R-1} \eta^{t_0} \mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \leq \frac{4\left[F(\Theta^0) - \mathbb{E}\left[F(\Theta^T)\right]\right]}{QR} + \frac{8}{3}\sum_{t_0=0}^{R-1}(QL(\eta^{t_0})^2 + 8Q^2L_m^2(\eta^{t_0})^3 + 8Q^3LL_m^2(\eta^{t_0})^4)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} + 86Q^2\sum_{t_0=0}^{R-1}\eta^{t_0}(1+2LQ\eta^{t_0})\sum_{m=0}^{M}H_m^2G_m^2\sum_{j\neq m}\mathcal{E}_j^{t_0}. \tag{A.90}$$

Suppose that $\eta^{t_0} = \eta$ for all global rounds t_0. Then, averaging over R global rounds, we have:

$$\frac{1}{R}\sum_{t_0=0}^{R-1}\mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \leq \frac{4\left[F(\Theta^0) - \mathbb{E}\left[F(\Theta^T)\right]\right]}{QR\eta} + \frac{8}{3}\sum_{m=0}^{M}(QL\eta + 8Q^2L_m^2\eta^2 + 8Q^3LL_m^2\eta^3)\frac{\sigma_m^2}{B} + \frac{86Q^2}{R}\sum_{t_0=0}^{R-1}(1+2LQ\eta)\sum_{m=0}^{M}H_m^2G_m^2\sum_{j\neq m}\mathcal{E}_j^{t_0}. \tag{A.91}$$

$$\leq \frac{4\left[F(\Theta^0) - \mathbb{E}\left[F(\Theta^T)\right]\right]}{QR\eta} + 6\sum_{m=0}^{M}QL\eta\frac{\sigma_m^2}{B} + \frac{92Q^2}{R}\sum_{t_0=0}^{R-1}\sum_{m=0}^{M}H_m^2G_m^2\sum_{j\neq m}\mathcal{E}_j^{t_0}. \tag{A.92}$$

where (A.92) follows from our assumption that $\eta^{t_0} \leq \frac{1}{16Q\max\{L, \max_m L_m\}}$. This completes the proof of Theorem 4.2.

We continue our analysis to prove Theorem 4.4. Starting from (A.90), we bound the left-hand side with the minimum over all iterations:

$$\min_{t_0=0,\dots,R-1} \mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \leq \frac{4\left[F(\Theta^0) - \mathbb{E}^{t_0}\left[F(\Theta^T)\right]\right]}{Q\sum_{t_0=0}^{R-1}\eta^{t_0}} + 4\left(QL\frac{\sum_{t_0=0}^{R-1}(\eta^{t_0})^2}{\sum_{t_0=0}^{R-1}\eta^{t_0}} + 16Q^2L_m^2\frac{\sum_{t_0=0}^{R-1}(\eta^{t_0})^3}{\sum_{t_0=0}^{R-1}\eta^{t_0}} + 16Q^3LL_m^2\frac{\sum_{t_0=0}^{R-1}(\eta^{t_0})^4}{\sum_{t_0=0}^{R-1}\eta^{t_0}}\right)\sum_{m=0}^{M}\frac{\sigma_m^2}{B} + 86Q^2\sum_{m=0}^{M}H_m^2G_m^2\frac{\sum_{t_0=0}^{R-1}\eta^{t_0}\sum_{j\neq m}\mathcal{E}_j^{t_0}}{\sum_{t_0=0}^{R-1}\eta^{t_0}} + 172LQ^3\sum_{m=0}^{M}H_m^2G_m^2\frac{\sum_{t_0=0}^{R-1}(\eta^{t_0})^2\sum_{j\neq m}\mathcal{E}_j^{t_0}}{\sum_{t_0=0}^{R-1}\eta^{t_0}} \tag{A.93}$$

As $R \to \infty$, if $\sum_{t_0=0}^{R-1}\eta^{t_0} = \infty$, $\sum_{t_0=0}^{R-1}(\eta^{t_0})^2 < \infty$, and $\sum_{t_0=0}^{R-1}\eta^{t_0}\sum_{j\neq m}\mathcal{E}_j^{t_0} < \infty$, then $\min_{t_0=0,\dots,R-1}\mathbb{E}\left[\left\|\nabla F(\Theta^{t_0})\right\|^2\right] \to 0$. This completes the proof of Theorem 4.4.

B. Common Compressors

In this section, we calculate the compression error and parameter bounds for uniform scalar quantization, lattice vector quantization and top-k sparsification, as well as discuss implementation details of these compressors in C-VFL.

We first consider a uniform scalar quantizer (Bennett, 1948) with a set of 2^q quantization levels, where q is the number of bits to represent compressed values. We define the range of values that quantize to the same quantization level as the quantization bin. In C-VFL, a scalar quantizer quantizes each individual component of embeddings. The error in each embedding of a batch $\mathbf{B}$ in scalar quantization is $\leq P_m\frac{\Delta^2}{12} = P_m\frac{(h_{max}-h_{min})^2}{12}2^{-2q}$ where Δ the size of a quantization bin, P_m is the size of the m-th embedding, h_{max} and h_{min} are respectively the maximum and minimum value $h_m(\theta_m^t; x_m^i)$ can be for all iterations t, parties m, and x_m^i. We note that if h_{max} or h_{min} are unbounded, then the error is unbounded as well. By Theorem 4.2, we know that $\frac{1}{R}\sum_{t_0=0}^{R-1}\sum_{m=0}^{M}\mathcal{E}_m^{t_0} = O(\frac{1}{\sqrt{T}})$ to obtain a convergence rate of $O(\frac{1}{\sqrt{T}})$. If we use the same q for all parties and iterations, we can solve for q to find that the value q must be lower bounded by $q = \Omega(\log_2(P_m(h_{max}-h_{min})^2\sqrt{T}))$ to reach a convergence rate of $O(\frac{1}{\sqrt{T}})$. For a diminishing compression error, required by Theorem 4.4, we let $T = t_0$ in this bound, indicating that q, the number of quantization bins, must increase as training continues.

A vector quantizer creates a set of d-dimensional vectors called a codebook (Zamir & Feder, 1996). A vector is quantized by dividing the components into sub-vectors of size d, then quantizing each sub-vector to the nearest codebook vector in Euclidean distance. A cell in vector quantization is defined as all points in d-space that quantizes to a single codeword. The volume of these cells are determined by how closely packed codewords are. We consider the commonly applied 2-dimensional hexagonal lattice quantizer (Shlezinger et al., 2021). In C-VFL, each embedding is divided into sub-vectors of size two, scaled to the unit square, then quantized to the nearest vector by Euclidean distance in the codebook. The error in this vector quantizer is $\leq \frac{VP_m}{24}$ where V is the volume of a lattice cell. The more bits available for quantization, the smaller the volume of the cells, the smaller the compression error. We can calculate an upper bound on V based on Theorem 4.2: $V = O(\frac{1}{P_m\sqrt{T}})$. If a diminishing compression error is required, we can set $T = t_0$ in this bound, indicating that V must decrease at a rate of $O(\frac{1}{P_m\sqrt{t_0}})$. As the number of iterations increases, the smaller V must be, and thus the more bits that must be communicated.

In top-k sparsification (Lin et al., 2018), when used in distributed SGD algorithms, the k largest magnitude components of the gradient are sent while the rest are set to zero. In the case of embeddings in C-VFL, a large element may be as important as an input to the server model as a small one. We can instead select the k embedding elements to send with the largest magnitude partial derivatives in $\nabla_{\theta_m}h_m(\theta_m^t)$. Since a party m cannot calculate $\nabla_{\theta_m}h_m(\theta_m^t)$ until all parties send their embeddings, party m can use the embedding gradient calculated in the previous iteration, $\nabla_{\theta_m}h_m(\theta_m^{t-1})$. This is an intuitive method, as we assume our gradients are Lipschitz continuous, and thus do not change too rapidly. The error of sparsification is $\leq (1-\frac{k}{P_m})(\|h\|^2)_{max}$ where $(\|h\|^2)_{max}$ is the maximum value of $\|h_m(\theta_m^t; x_m^i)\|^2$ for all iterations t, parties m, and x_m^i. Note that if $(\|h\|^2)_{max}$ is unbounded, then the error is unbounded. We can calculate a lower bound on k: $k = \Omega(P_m - \frac{P_m}{(\|h\|^2)_{max}\sqrt{T}})$. Note that the larger $(\|h\|^2)_{max}$, the larger k must be. More components must be sent if embedding magnitude is large in order to achieve a convergence rate of $O(\frac{1}{\sqrt{T}})$. When considering a diminishing

compression error, we set $T = t_0$, showing that k must increase over the course of training.

C. Experimental Details

For our experiments, we used an internal cluster of 40 compute nodes running CentOS 7 each with $2\times$ 20-core 2.5 GHz Intel Xeon Gold 6248 CPUs, $8\times$ NVIDIA Tesla V100 GPUs with 32 GB HBM, and 768 GB of RAM.

C.1. MIMIC-III

The MIMIC-III dataset can be found at: mimic.physionet.org. The dataset consists of time-series data from $\sim$60,000 intensive care unit admissions. The data includes many features about each patient, such as demographic, vital signs, medications, and more. All the data is anonymized. In order to gain access to the dataset, one must take the short online course provided on their website.

Our code for training with the MIMIC-III dataset can be found in in the folder titled "mimic3". This is an extension of the MIMIC-III benchmarks repo found at: github.com/YerevaNN/mimic3-benchmarks. The original code preprocesses the MIMIC-III dataset and provides starter code for training LSTMs using centralized SGD. Our code has updated their existing code to TensorFlow 2. The new file of interest in our code base is "mimic3models/in_hospital_mortality/quant.py" which runs C-VFL. Both our code base and the original are under the MIT License. More details on installation, dependencies, and running our experiments can be found in "README.md". Each experiment took approximately six hours to run on a node in our cluster.

The benchmarking preprocessing code splits the data up into different prediction cases. Our experiments train models to predict for in-hospital mortality. For in-hospital mortality, there are $14{,}681$ training samples, and $3{,}236$ test samples. In our experiments, we use a step size of 0.01, as is standard for training an LSTM on the MIMIC-III dataset.

C.2. ModelNet10 and CIFAR10

Details on the ModelNet10 dataset can be found at: modelnet.cs.princeton.edu/. The specific link we downloaded the dataset from is the following Google Drive link: https://drive.google.com/file/d/0B4v2jR3WsindMUE3N2xiLVpyLW8/view. The dataset consists of 3D CAD models of different common objects in the world. For each CAD model, there are 12 views from different angles saved as PNG files. We only trained our models on the following 10 classes: bathtub, bed, chair, desk, dresser, monitor, night_stand, sofa, table, toilet. We used a subset of the data with $1{,}008$ training samples and 918 test samples. In our experiments, we use a step size of 0.001, as is standard for training a CNN on the ModelNet10 dataset.

Our code for learning on the ModelNet10 dataset is in the folder "MVCNN_Pytorch" and is an extension of the MVCNN-PyTorch repo: github.com/RBirkeland/MVCNN-PyTorch. The file of interest in our code base is "quant.py" which runs C-VFL. Both our code base and the original are under the MIT License. Details on how to run our experiments can be found in the "README.md". Each experiment took approximately six hours to run on a node in our cluster.

In the same folder, "MVCNN_Pytorch", we include our code for running CIFAR-10. The file of interest is "quant_cifar.py" which trains C-VFL with CIFAR-10. We use the version of CIFAR-10 downloaded through the torchvision library. More information on the CIFAR-10 dataset can be found at: cs.toronto.edu/ kriz/cifar.html.

C.3. ImageNet

In Section D, we include additional experiments that use the ImageNet dataset. Details on ImageNet can be found at: image-net.org/. We specifically use a random 100-class subset from the 2012 ILSVRC version of the data.

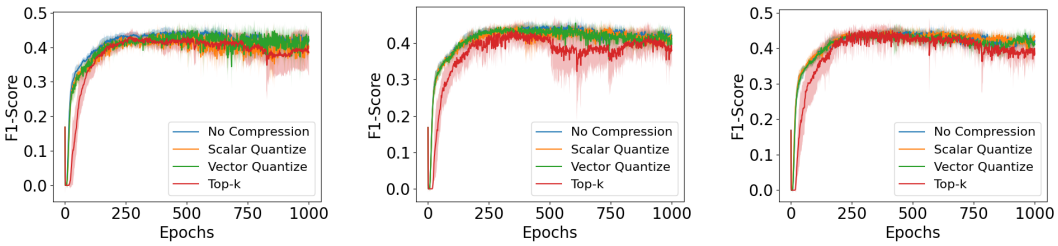
Our code for learning on the ImageNet dataset is in the folder "ImageNet_CVFL" and is a modification on the moco_align_uniform repo: https://github.com/SsnL/moco_align_uniform. The file of interest in our code base is "main_cvfl.py" which runs C-VFL. Both our code base and the original are under the CC-BY-NC 4.0 license. Details on how to run our experiments can be found in the "README.txt". Each experiment took approximately 24 hours to run on a node in our cluster.

D. Additional Plots and Experiments

In this section, we include additional plots using the results from the experiments introduced in Section 5 of the main paper. We also include new experiments with the ImageNet100 dataset. Finally, we include additional experiments with an alternate C-VFL for $Q = 1$.

D.1. Additional Plots

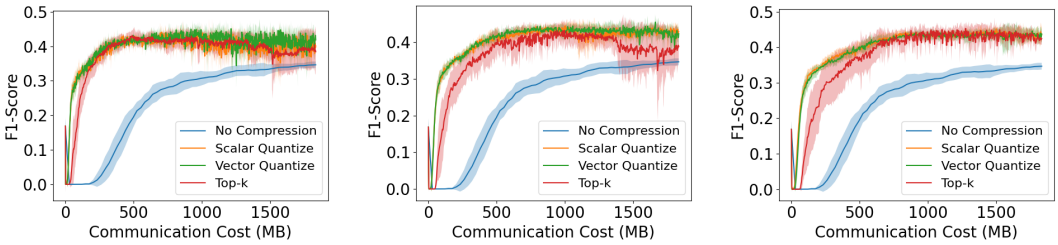
We first provide additional plots from the experiements in the main paper. The setup for the experiments is described in the main paper. These plots provide some additional insight into the effect of each compressor on convergence in all datasets. As with the plots in the main paper, the solid lines in each plot are the average of five runs and the shaded regions represent one standard deviation.



(a) 2 bits per parameter (b) 3 bits per parameter (c) 4 bits per parameter

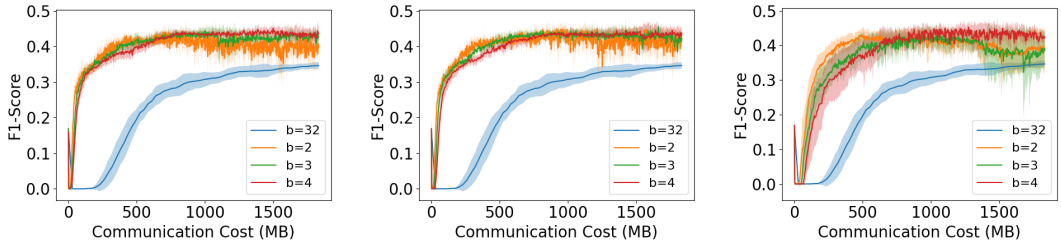
Figure D.1. Test F_1-Score on MIMIC-III dataset. Scalar and vector In these experiments, $Q = 10$ and $M = 4$. quantization achieve similar test F_1-Score even when only using 2 bits in quantization. On the other hand, top-k sparsification performs worse than the other compressors in the MIMIC-III dataset.

Figure D.1 plots the test F_1-Score for training on the MIMIC-III dataset for different levels of compression. We can see that scalar and vector quantization perform similarly to no compression and improve as the number of bits available increase. We can also see that top-k sparsification has high variability on the MIMIC-III dataset and generally performs worse than the other compressors.



(a) 2 bits per parameter (b) 3 bits per parameter (c) 4 bits per parameter

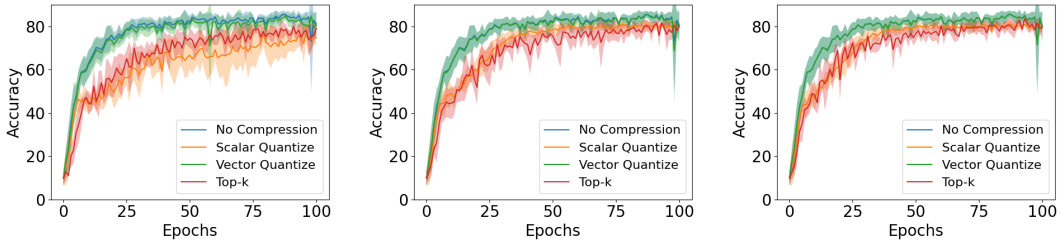
Figure D.2. Test F_1-Score on MIMIC-III dataset plotted by communication cost. In these experiments, $Q = 10$ and $M = 4$. We can see that all compressors reach higher F_1-Scores with lower communication cost than no compression. We can see that the standard deviation for each compressor decreases as the number of bits available increases. Top-k sparsification generally performs worse than the other compressors on the MIMIC-III-dataset.



(a) Scalar quantization (b) Vector quantization (c) Top-k sparsification

Figure D.3. Test F_1-Score on MIMIC-III dataset plotted by communication cost. In these experiments, $Q = 10$ and $M = 4$. We can see that all compressors reach higher F_1-Scores with lower communication cost than no compression. We can see that the variability for each compressor decreases as the number of bits available increases.

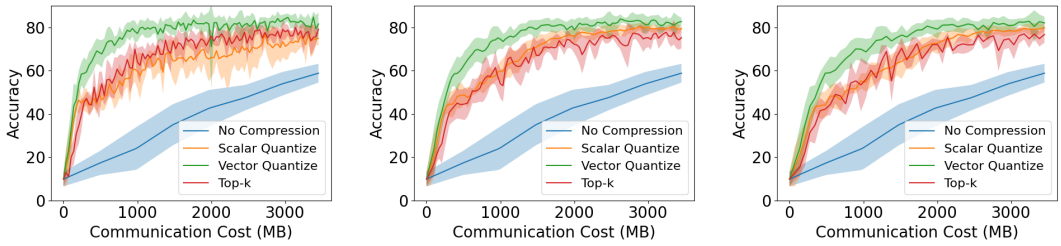
Figures D.2 and D.3 plot the test F_1-Score for training on the MIMIC-III dataset plotted against the communication cost. The plots in Figure D.2 include all compression techniques for a given level of compression, while the plots in Figure D.3 include all levels of compression for a given compression technique. We can see that all compressors reach higher F_1-Scores with lower communication cost than no compression. It is interesting to note that increasing the number of bits per parameter reduces the variability in all compressors.



(a) 2 bits per parameter (b) 3 bits per parameter (c) 4 bits per parameter

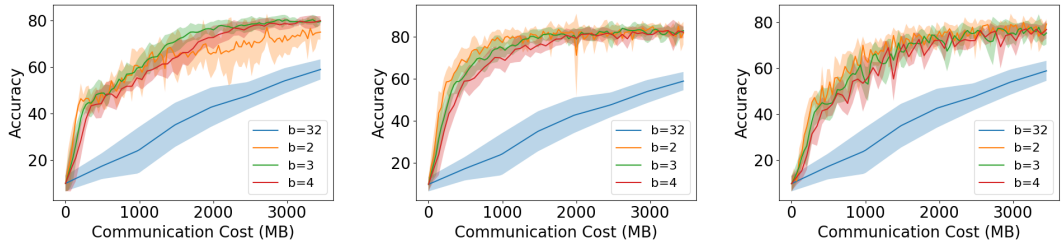
Figure D.4. Test accuracy on ModelNet10 dataset. Vector quantization In these experiments, $Q = 10$ and $M = 4$. and top-k sparsification perform similarly to no compression, even when only 2 bits are available. Scalar quantization converges to a lower test accuracy, and has high variability on the ModelNet10 dataset.

Figure D.4 plots the test accuracy for training on the ModelNet10 dataset. Vector quantization and top-k sparsification perform similarly to no compression, even when only 2 bits are available. We can see that scalar quantization has high variability on the ModelNet10 dataset.



(a) 2 bits per parameter (b) 3 bits per parameter (c) 4 bits per parameter

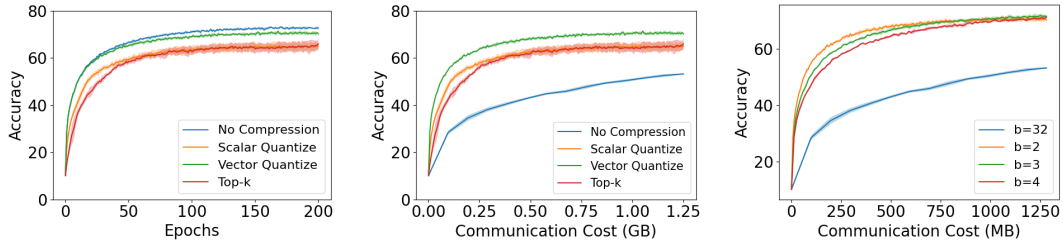
Figure D.5. Test accuracy on ModelNet10 dataset plotted by communication cost. In these experiments, $Q = 10$ and $M = 4$. We can see that all compressors reach higher accuracies with lower communication cost than no compression. Scalar quantization generally performs worse than the other compressors on the ModelNet10 dataset.



(a) Scalar quantization (b) Vector quantization (c) Top-k sparsification

Figure D.6. Test accuracy on ModelNet10 dataset plotted by communication cost. In these experiments, $Q = 10$ and $M = 4$. We can see that all compressors reach higher accuracies with lower communication cost than no compression. We can see that when less bits are used in each compressor, higher test accuracies are reached at lower communication costs. Scalar quantization generally performs worse than the other compressors on the ModelNet10 dataset.

Figures D.5 and D.6 plot the test accuracy for training on the ModelNet10 dataset against the communication cost. The plots in Figure D.5 include all compression techniques for a given level of compression, while the plots in Figure D.6 include all levels of compression for a given compression technique. We can see that all compressors reach higher accuracies with lower communication cost than no compression. Scalar quantization generally performs worse than the other compressors on the ModelNet10 dataset. From Figure D.6, we also see that when fewer bits are used in each compressor, higher test accuracies are reached at lower communication costs.



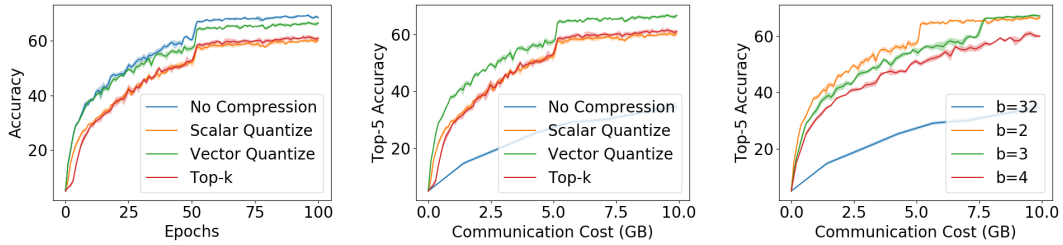
(a) Plotted by epochs (b) Plotted by cost (c) Vector quantization

Figure D.7. Test accuracy on CIFAR-10 dataset with the number of parties $M = 4$ and number of local iterations $Q = 10$. In the first two plots, the compressors have $b = 2$, where b is the number of bits used to represent embedding components. In the third plot, $b = 32$ indicates there is no compression. The results show vector quantization performs the best our of the compressors, and all compressors show improvement over no compression in terms of communication cost to reach target test accuracies.

In Figure D.7, we plot the test accuracy for the CIFAR-10 dataset. The test accuracy is fairly low compared to typical baseline accuracies, which is expected, as learning object classification from only a quadrant of a 32×32 pixel image is difficult. Figure D.7a shows the test accuracy plotted by epochs. We can see that vector quantization performs almost as well as no compression in the CIFAR-10 dataset. When plotting by communication cost, seen in Figure D.7b, we can see that vector quantization performs the best, though scalar quantization and top-k sparsification show communication savings as well. In Figure D.7c, we plot the test accuracy of C-VFL using vector quantization for different values of b, the number of bits to represent compressed values. Similar to previous results, lower b tends to improve test accuracy reached with the same amount of communication cost.

D.2. Additional Experiments With ImageNet

We also run C-VFL on ImageNet100 (Deng et al., 2009). ImageNet is a large image dataset for object classification. We use a random subset of 100 classes (ImageNet100) from the ImageNet dataset (about 126,000 images). We train a set of 4 parties, each storing a different quadrant of every image. Each party trains ResNet18, and the server trains a fully-connected layer. We use a variable step size, that starts at 0.001, and drops to 0.0001 after 50 epochs. We use a batch size of 256 and train for 100 epochs.



(a) Plotted by epochs (b) Plotted by cost (c) Vector quantization

Figure D.8. Test accuracy on ImageNet-100 dataset with the number of parties $M = 4$ and number of local iterations $Q = 10$. In the first two plots, the compressors have $b = 2$, where b is the number of bits used to represent embedding components. In the third plot, $b = 32$ indicates there is no compression. The results show vector quantization performs the best our of the compressors, and all compressors show improvement over no compression in terms of communication cost to reach target test accuracies.

In Figure D.8, we plot the top-5 test accuracy for ImageNet100. Figure D.8a shows the test accuracy plotted by epochs. We can see that vector quantization performs almost as well as no compression in the ImageNet100 dataset. When plotting by communication cost, seen in Figure D.8b, we can see that vector quantization performs the best, though scalar quantization and top-k sparsification show communication savings as well. In Figure D.8c, we plot the test accuracy of C-VFL using vector quantization for different values of b, the number of bits to represent compressed values. Similar to previous results, lower b tends to improve test accuracy reached with the same amount of communication cost.

D.3. Comparison With Alternative C-VFL Algorithm For $Q = 1$

In C-VFL, the server distributes party embeddings to all parties along with the server model parameters. This allows parties to calculate their partial derivatives for local model updates for multiple local iterations. However, if the number of local iterations $Q = 1$, then a more efficient method of communication is for the server to compute partial derivative updates for the parties (Hu et al., 2019; Ceballos et al., 2020; Romanini et al., 2021), avoiding the need for parties to receive embeddings from other parties. This approach can be applied to C-VFL as well.

Algorithm 2 Compressed Vertical Federated Learning for $Q = 1$

Initialize: θ_m^0 for all parties m and server model θ_0^0
for $t \leftarrow 0, \ldots, T-1$ **do**
  Randomly sample $\mathbf{B}^t \in \{\mathbf{X}, \mathbf{y}\}$
  for $m \leftarrow 1, \ldots, M$ in parallel **do**
    Send $\mathcal{C}_m(h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t}))$ to server
  end for
  $\hat{\Phi}^t \leftarrow \{\mathcal{C}_0(\theta_0), \mathcal{C}_1(h_1(\theta_1^t)), \ldots, \mathcal{C}_M(h_M(\theta_M^t))\}$
  $\theta_0^{t+1} = \theta_0^t - \eta^t \nabla_0 F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t})$
  Server sends $\nabla_{h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t})} F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t})$ to each party m
  for $m \leftarrow 1, \ldots, M$ in parallel **do**
    $\nabla_m F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t}) = h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t}) \nabla_{h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t})} F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t})$
    $\theta_m^{t+1} = \theta_m^t - \eta^t \nabla_m F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t})$
  end for
end for

The pseudo-code for this method is presented in Algorithm 2. In this version of C-VFL, parties send their compressed embeddings to the server. The server calculates the loss by feeding the embeddings through the server model. The server calculates the gradient with respect to the loss. The server then sends to each party m the partial derivative with respect to its embedding: $\nabla_{h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t})} F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t})$. Each party m calculates the following partial derivative with respect to its local

Table D.1. MIMIC-III communication cost to reach a target test F_1-Score of 0.4. Value shown is the mean of 5 runs, $\pm$ the standard deviation. The first row has no embedding compression, while the second row employs vector quantization on embeddings with $b = 3$. For the cases where $Q = 1$, Algorithm 2 is used, and for cases where $Q > 1$, Algorithm 1 is used. In these experiments, the number of clients $M = 4$.

Compressor	**Cost (MB) to Reach Target F_1-Score** 0.4		
	$Q = 1$	$Q = 10$	$Q = 25$
None $b = 32$	4517.59 $\pm$ 465.70	3777.21 $\pm$ 522.43	1536.69 $\pm$ 201.99
Vector $b = 3$	433.28 $\pm$ 79.83	330.47 $\pm$ 10.63	125.09 $\pm$ 4.56

model parameters:

$$\nabla_m F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t}) = h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t}) \nabla_{h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t})} F_{\mathbf{B}}(\hat{\Phi}^t; \mathbf{y}^{\mathbf{B}^t}). \tag{D.1}$$

Using this partial derivative, the party updates its local model:

$$\theta_m^{t+1} = \theta_m^t - \eta^{t_0} \nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t; \mathbf{y}^{\mathbf{B}^{t_0}}). \tag{D.2}$$

Note that this process is mathematically equivalent to C-VFL when $Q = 1$; thus the analysis in Section 4 holds. The communication cost of Algorithm 2 per communication round without compression is $O(B \cdot \sum_m P_m)$, a reduction in communication compared to the communication cost per round of Algorithm 1: $O(M \cdot (B \cdot \sum_m P_m + |\theta_0|))$. Although Algorithm 2 reduces communication in a given round, it is limited to the case when $Q = 1$. For $Q > 1$, we must use Algorithm 1.

We run experiments on the MIMIC-III dataset to compare C-VFL using Algorithm 2 with C-VFL using Algorithm 1 with values of $Q > 1$. We show the results of these experiments in Table D.1. Here, we show the communication cost to reach a target F_1-Score of 0.4. The results in the column labeled $Q = 1$ are from running Algorithm 2, while all other results are from running Algorithm 1. We include results for the case where C-VFL is run without embedding compression, as well as results for the case where vector quantization with $b = 3$ is used to compress embeddings. We can see that for this dataset, values of $Q > 1$ reduce the cost of communication to reach the target F_1-Score. In all cases, Algorithm 1 achieves lower communication cost to reach a target model accuracy than Algorithm 2, despite Algorithm 2 having a lower communication cost per communication round than Algorithm 1. The use of multiple local iterations in Algorithm 1 decreased the number of global rounds required to attain the target accuracy compared to Algorithm 2.