Communication-Efficient Vertical Federated Learning via Compressed Error Feedback

Pedro Valdeira, João Xavier, Cláudia Soares, and Yuejie Chi, *Fellow, IEEE*

***Abstract*—Communication overhead is a known bottleneck in federated learning (FL). To address this, lossy compression is commonly used on the information communicated between the server and clients during training. In horizontal FL, where each client holds a subset of the samples, such communication-compressed training methods have recently seen significant progress. However, in their vertical FL counterparts, where each client holds a subset of the features, our understanding remains limited. To address this, we propose an error feedback compressed vertical federated learning (`EF-VFL`) method to train split neural networks. In contrast to previous communication-compressed methods for vertical FL, `EF-VFL` does not require a vanishing compression error for the gradient norm to converge to zero for smooth nonconvex problems. By leveraging error feedback, our method can achieve a $\mathcal{O}(1/T)$ convergence rate for a sufficiently large batch size, improving over the state-of-the-art $\mathcal{O}(1/\sqrt{T})$ rate under $\mathcal{O}(1/\sqrt{T})$ compression error, and matching the rate of uncompressed methods. Further, when the objective function satisfies the Polyak-Łojasiewicz inequality, our method converges linearly. In addition to improving convergence, our method also supports the use of private labels. Numerical experiments show that `EF-VFL` significantly improves over the prior art, confirming our theoretical results.**

***Index Terms*—Vertical federated learning, nonconvex optimization, communication-compressed optimization.**

Received 23 June 2024; revised 2 January 2025; accepted 27 January 2025. Date of publication 11 February 2025; date of current version 4 March 2025. This work was supported in part by the Fundação para a Ciência e a Tecnologia through the Carnegie Mellon Portugal Program under grant SFRH/BD/150738/2020; in part by the U.S. National Science Foundation under Grant CCF-2007911 and Grant ECCS-2318441; in part by NOVA LINCS under Grant UIDB/04516/2020 (https://doi.org/10.54499/UIDB/04516/2020) and Grant UIDP/04516/2020 (https://doi.org/10.54499/UIDP/04516/2020); in part by LARSyS FCT funding (DOI: 10.54499/LA/P/0083/2020); in part by PT Smart Retail project [PRR—02/C05-i11/2024.C645440011-00000062], through IAPMEI—Agência para a Competitividade e Inovação; and in part by TaRDIS Horizon2020 under Grant 101093006. An earlier version of this paper was presented at the EUSIPCO 2024 [DOI: 10.23919/EUSIPCO63174.2024.10715377]. The associate editor coordinating the review of this article and approving it for publication was Qing Ling. *(Corresponding author: Pedro Valdeira.)*

Pedro Valdeira is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA, also with the Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisbon, Portugal, and also with the Laboratory for Robotics and Engineering Systems, Institute for Systems and Robotics, 1600-011 Lisbon, Portugal (e-mail: pvaldeira@cmu.edu).

João Xavier is with the Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisbon, Portugal, and also with the Laboratory for Robotics and Engineering Systems, Institute for Systems and Robotics, 1600-011 Lisbon, Portugal (e-mail: jxavier@isr.ist.utl.pt).

Cláudia Soares is with the Department of Computer Science, NOVA School of Science and Technology, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal (e-mail: claudia.soares@fct.unl.pt).

Yuejie Chi is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: yuejiechi@cmu.edu).

Digital Object Identifier 10.1109/TSP.2025.3540655

I. Introduction

FEDERATED learning (FL) is a machine learning paradigm where a set of clients holding local datasets collaborate to train a model without exposing their local data [2], [3], [4]. FL can be divided into two categories, based on how the data is partitioned across the clients: *horizontal* FL, where each client holds a different set of samples but all clients share the same features, and *vertical* FL, where each client holds a different subset of features but all clients share the same samples. Note that we cannot gather and redistribute the data because it must remain at the clients. Thus, we do not choose under which category a given task falls. Rather, the category is a consequence of how the data arises.

In this work, we focus on vertical FL (VFL) [5]. In VFL, a global dataset $\mathcal{D} = \{\boldsymbol{\xi}_n\}_{n=1}^N$ with N samples is partitioned by features across a set of clients $[K] := \{1, \dots, K\}$. Each sample has K disjoint blocks of features $\boldsymbol{\xi}_n = (\boldsymbol{\xi}_{n1}, \dots, \boldsymbol{\xi}_{nK})$ and the local dataset of each client $k \in [K]$ is $\mathcal{D}_k = \{\boldsymbol{\xi}_{nk}\}_{n=1}^N$, where $\mathcal{D} = \bigcup_k \mathcal{D}_k$. Since different datasets $\mathcal{D}_k$ have different features, VFL suits collaborations of clients with complementary types of information, who tend to have fewer competing interests. This can lead to a greater incentive to collaborate, compared to horizontal FL. A common application of VFL is in settings where multiple entities own distinct features concerning a shared set of users and seek to collaboratively train a predictor; for example, WeBank partners with other companies to jointly build a risk model from data regarding shared customers [6].

To jointly train a model from $\{\mathcal{D}_k\}$ without sharing local data, split neural networks [7] are often considered. To learn the parameters $\boldsymbol{x}$ of such models, we aim to solve the following nonconvex optimization problem:

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) := \frac{1}{N} \sum_{n=1}^{N} \phi_n \left(\boldsymbol{x}_0, \boldsymbol{h}_{1n}(\boldsymbol{x}_1), \dots, \boldsymbol{h}_{Kn}(\boldsymbol{x}_K) \right), \quad (1)$$

where $\boldsymbol{x} := (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_K)$. Here, $\boldsymbol{h}_{kn}(\boldsymbol{x}_k) := \boldsymbol{h}_k(\boldsymbol{x}_k; \boldsymbol{\xi}_{nk})$ is the *representation* of $\boldsymbol{\xi}_{nk}$ extracted by the local model of client $k \in [K]$, which is parameterized by $\boldsymbol{x}_k$. This representation is then sent to the server. The server, in turn, uses

$\{\boldsymbol{h}_{kn}(\boldsymbol{x}_k)\}_{k=1}^K$ as input to ϕ_n, which corresponds to the composition of the loss function and the *fusion* model and is parameterized by $\boldsymbol{x}_0$.[1]

Most FL methods, including ours, assume that the server can communicate with all the clients and that the clients do not communicate with each other. These methods typically require many rounds of client-server communications. Such communications can significantly slow down training. In fact, they can become the main bottleneck during training [8], [9]. To address this, a plethora of communication-efficient FL methods have been proposed. In particular, a popular technique to mitigate the communication overhead is lossy compression. Compression operators, or simply compressors, are operators that map a given vector into another vector that is easier to communicate (that is, requires fewer bits).

Optimization methods employing communication compression have seen great success [10], [11]. Most of these works focus on the prevalent horizontal FL setup and thus consider *gradient* compression, as these are the vectors being communicated in the horizontal setting [12], [13]. Yet, in vertical FL, the clients send *representations* $\{\boldsymbol{h}_{kn}(\boldsymbol{x}_k)\}$ instead. In contrast to gradient compression, compressing these intermediate representations leads the compression-induced error to undergo a nonlinear function ϕ_n before impacting gradient-based updates. Thus, compression in VFL is not covered by these works and our understanding of it remains limited. In fact, to the best of our knowledge, [14] is the only work providing convergence guarantees for compressed VFL. Yet, [14] employs a direct compression method, requiring the compression error to go to zero as the number of gradient steps T increases, achieving a $\mathcal{O}(1/\sqrt{T})$ rate when the compression error is $\mathcal{O}(1/\sqrt{T})$. This makes the method in [14] unsuitable for applications with strict per-round communication limitations, such as bandwidth constraints, and leads to the following question:

Can we design a communication-compressed VFL method that preserves the convergence rate of uncompressed methods without decreasing the amount of compression as training progresses?

A. Our Contributions

In this work, we answer the question above in the affirmative. Our main contributions are as follows.

- We propose error feedback compressed VFL (`EF-VFL`), which leverages an error feedback technique to improve the stability of communication-compressed training in VFL.
- We show that our method achieves a convergence rate of $\mathcal{O}(1/T)$ for nonconvex objectives under nonvanishing compression error for a sufficiently large batch size, improving over the state-of-the-art $\mathcal{O}(1/\sqrt{T})$ rate under $\mathcal{O}(1/\sqrt{T})$ compression error of [14], and matching the rate of the centralized setting. We further show that, under the Polyak-Łojasiewicz (PL) inequality, our method converges linearly to the solution, in the full-batch case, and, more generally, to a neighborhood of size proportional to the mini-batch variance, thus obtaining the first linearly convergent compressed VFL method. Unlike the method in [14], `EF-VFL` supports the use of private labels, broadening its applicability.
- We run numerical experiments and observe empirically that our method improves the state-of-the-art, achieving a better communication efficiency than existing methods.

[1]The server often aggregates the representations $\boldsymbol{h}_{kn}$ via some nonparameterized operation (for example, a sum or an average) before inputting them into the server model. We consider this aggregation to be included in ϕ_n.

B. Related Work

Communication-Efficient FL. The aforementioned communication bottleneck in FL [9] makes communication-efficient methods a particularly active area of research. FL methods often employ multiple local updates between rounds of communication, use only a subset of the clients at a time [2], or even update the global model asynchronously [15]. Another line of research considers fully decentralized methods [9], dispensing with the server and, instead, exploiting communications between clients. This can alleviate the bandwidth limit ensuing from the centralized role of the server. Another popular technique is lossy compression, which is the focus of this work. In both FL and, more generally, in the broader area of distributed optimization [16], [17], communication-compressed methods have recently received significant attention [18], [19], [20], [21].

Communication-compressed optimization methods can exploit different families of compressors. A popular choice is the family of *unbiased* compressors [11], which is appealing in that its properties facilitate the theoretical analysis of the resulting methods. Yet, some widely adopted compressors do not belong to this class, such as top-k sparsification [12], [13] and deterministic rounding [22]. Thus, the broader family of *contractive* compressors [23] has recently attracted a lot of attention. Yet, methods employing them for direct compression are often prone to instability or even divergence [23]. To address this, error feedback techniques have been proposed; first, as a heuristic [10], but, more recently, significant progress has been made on our theoretical understanding of the application of these methods to gradient compression [12], [13], [24]. In the horizontal setting, some works have combined error feedback compression with the aforementioned communication-efficient techniques, such as communication-compressed fully decentralized methods [25], [26].

Vertical FL. To mitigate the communication bottleneck in VFL, we often employ techniques akin to those used in horizontal FL. In particular, [27] performed multiple local updates between communication rounds, [28] updated the local models asynchronously, and [29] proposed a fully decentralized approach. Recently, [30] proposed a semi-decentralized method leveraging both client-server and client-client communications to avoid the slow convergence of fully-decentralized methods on large and sparse networks while alleviating the server bottleneck.

In this work, we focus on communication-compressed methods tailored to VFL. While, as mentioned above, most work in communication-compressed methods focuses on gradient compression and thus does not apply directly to VFL, recently, a few empirical works on VFL have employed compression. Namely, [31] compressed the local data before sending it to the server, where the model is trained, and [32] proposed an asynchronous method with bidirectional (sparse) gradient compression. Nevertheless, [14], where direct compression is used, is the only work on compressed VFL with theoretical guarantees. For a more a detailed discussion on VFL, see [5] and [33].

II. Preliminaries

We now define the class of contractive compressors, which we consider throughout this paper.

Definition 1 (Contractive compressor): A map $\mathcal{C}\colon \mathbb{R}^d \mapsto \mathbb{R}^d$ is a contractive compressor if there exists $\alpha \in (0,1]$ such that,[2]

$$\forall \boldsymbol{v} \in \mathbb{R}^d\colon \quad \mathbb{E}\|\mathcal{C}(\boldsymbol{v}) - \boldsymbol{v}\|^2 \leq (1-\alpha)\|\boldsymbol{v}\|^2, \tag{2}$$

where the expectation is taken with respect to the (possible) randomness in $\mathcal{C}$.

A. Error Feedback

When transmitting a converging sequence of vectors $\{\boldsymbol{v}^t\}$, error feedback mechanisms can reduce the compression error compared to direct compression. In communication-compressed optimization, this allows for faster and more stable convergence.

In direct compression, each $\boldsymbol{v}^t$ is compressed independently, with $\mathcal{C}(\boldsymbol{v}^t)$ simply replacing $\boldsymbol{v}^t$ at the receiver. In contrast, in error feedback compression, the receiver employs a surrogate for $\boldsymbol{v}^t$ that incorporates information from previous steps $i = 0, \ldots, t-1$. This is achieved by resorting to an auxiliary vector that is stored in memory and updated at each step, leveraging *feedback* from the compression of $\{\boldsymbol{v}^i\colon i = 0, \ldots, t-1\}$ to refine the surrogate for $\boldsymbol{v}^t$.

Earlier communication-compressed optimization methods employing error feedback mechanisms were motivated by sigma-delta modulation [34]. In these methods, the auxiliary vector accumulated the compression *error* across steps, adding the accumulated error to the current vector $\boldsymbol{v}^t$ before compressing and transmitting it [10], [12], [35].

More recently, a new type of error feedback mechanism has been proposed, where the auxiliary vector $\boldsymbol{s}^t$ tracks $\boldsymbol{v}^t$ directly, rather than the accumulated compression error. This mechanism uses the (compressed) difference between $\boldsymbol{v}^{t+1}$ and $\boldsymbol{s}^t$, that is, the *error* of the surrogate, as the feedback. This approach was first introduced in [36] for unbiased compressors and was later extended to the more general class of contractive compressors in EF21 [24]. More formally, in EF21, the surrogate $\boldsymbol{s}^t$ is initialized as $\boldsymbol{s}^0 = \mathcal{C}(\boldsymbol{v}^0)$ and updated recursively as:

$$\forall t \geq 0\colon \quad \boldsymbol{s}^{t+1} = \boldsymbol{s}^t + \mathcal{C}(\boldsymbol{v}^{t+1} - \boldsymbol{s}^t). \tag{3}$$

[2]In Definition 1, we use a common, simplified notation, omitting the randomness in $\mathcal{C}\colon \mathbb{R}^d \times \Omega \to \mathbb{R}^d$. We assume that the randomness ω in $\mathcal{C}(\boldsymbol{v}, \omega)$ at different applications of $\mathcal{C}$ is independent.

Note that, unlike $\mathcal{C}(\boldsymbol{v}^t)$, $\boldsymbol{s}^t$ is not necessarily in the range of $\mathcal{C}$. For example, if $\mathcal{C}$ uses sparsification, then the surrogate at the receiver in direct compression, $\mathcal{C}(\boldsymbol{v}^t)$, must be sparse, whereas $\boldsymbol{s}^t$ does not need to be. By continually updating $\boldsymbol{s}^t$ with the compressed error, we ensure that it tracks $\boldsymbol{v}^t$. Moreover, since these updates are compressed, the communication cost remains the same as in direct compression. To maintain consistency, the surrogate $\boldsymbol{s}^t$ is updated at both the sender (client or server) and the receiver. In this work, we adopt an EF21-based error feedback mechanism and henceforth refer to it simply as error feedback.

B. Problem Setup

Let $\boldsymbol{h}_{0n}(\boldsymbol{x}_0) = \boldsymbol{x}_0$ and $\boldsymbol{h}_n(\boldsymbol{x}) = (\boldsymbol{h}_{0n}(\boldsymbol{x}_0), \ldots, \boldsymbol{h}_{Kn}(\boldsymbol{x}_K)) \in \mathbb{R}^E$ where $\boldsymbol{h}_{kn}(\boldsymbol{x}_k) \in \mathbb{R}^{E_k}$ and $E = \sum_{k=0}^K E_k$, for all n. Further, let

$$\boldsymbol{H}_k(\boldsymbol{x}_k) = \begin{bmatrix} \boldsymbol{h}_{k1}(\boldsymbol{x}_k) \\ \vdots \\ \boldsymbol{h}_{kN}(\boldsymbol{x}_k) \end{bmatrix} \in \mathbb{R}^{N \times E_k}$$

and

$$\boldsymbol{H}(\boldsymbol{x}) := [\boldsymbol{H}_0(\boldsymbol{x}_0), \boldsymbol{H}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{H}_K(\boldsymbol{x}_K)] \in \mathbb{R}^{N \times E}.$$

Further, we define $\Phi\colon \mathbb{R}^{N \times E} \mapsto \mathbb{R}$ as follows:

$$f(\boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^N \phi_n\left(\boldsymbol{h}_n(\boldsymbol{x})\right) =: \Phi\left(\boldsymbol{H}(\boldsymbol{x})\right).$$

Throughout most of the paper, we assume that ϕ_n contains the label of $\boldsymbol{\xi}_n$ and that ϕ_n is known by all the clients and the server. This assumption, known as "relaxed protocol" [5], is sometimes made in VFL [14], [37], [38] and has applications, for example, in credit score prediction. We also address the case of private labels, proposing a modified version of our method for that setting in Section III-A.

We assume that f has an optimal value $f^\star := \min_{\boldsymbol{x}} f(\boldsymbol{x}) > -\infty$ and make the following assumptions, where ∇ denotes not only the gradient of scalar-valued functions but, more generally, the derivative of a (possibly multidimensional) map.

Assumption 1 (Smoothness): A function $h\colon \mathbb{R}^d \mapsto \mathbb{R}$ is L-smooth if there exists a positive constant L such that

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d\colon \quad \|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|. \tag{A1}$$

We assume f is L_f-smooth and Φ is L_Φ-smooth and let $L = \max\{L_f, L_\Phi\}$.

Assumption 2 (Bounded derivative): Map $\boldsymbol{F}\colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2 \times p_3}$ has a bounded derivative if there exists a positive constant H such that

$$\forall \boldsymbol{x} \in \mathbb{R}^{p_1}\colon \quad \|\nabla \boldsymbol{F}(\boldsymbol{x})\| \leq H, \tag{A2}$$

where $\|\nabla \boldsymbol{F}(\boldsymbol{x})\|$ is the Euclidean norm of the third-order tensor $\nabla \boldsymbol{F}(\boldsymbol{x})$. We assume (A2) holds for $\{\boldsymbol{H}_k\}_{k=0}^K$.

Note that, in Assumption 2, we do *not* assume that our objective function f has a bounded gradient. We only require the local representation-extracting maps $\{\boldsymbol{H}_k\}$ to have a bounded derivative. The same assumption is also made in [14].

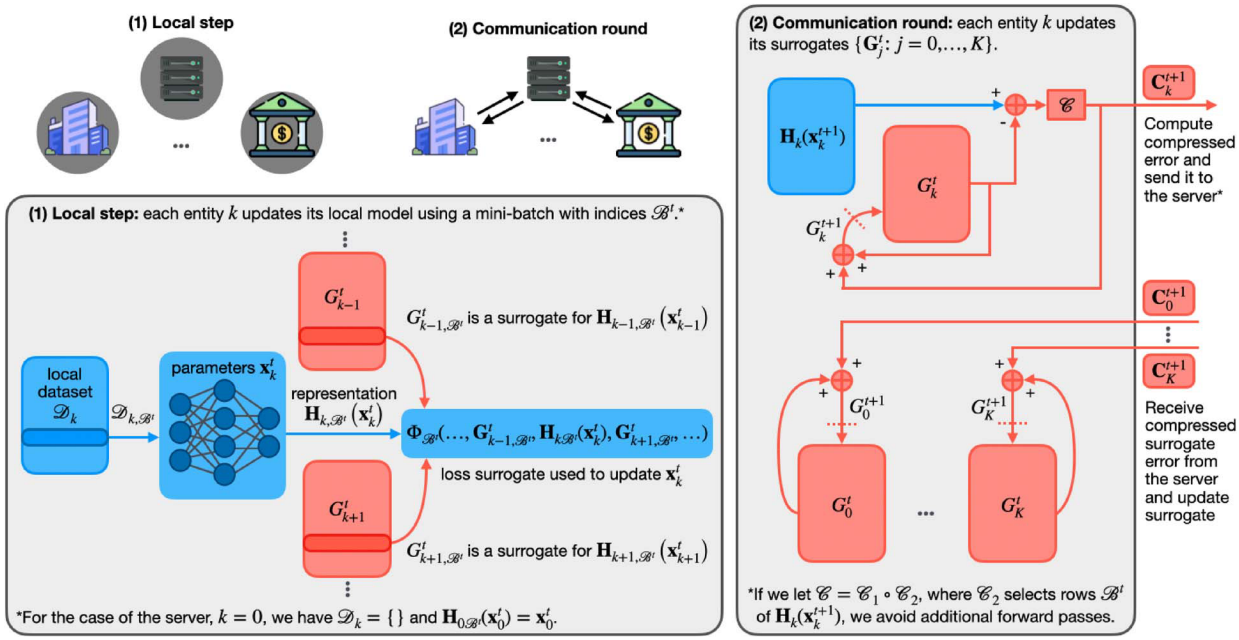


Fig. 1. An illustration of an iteration of the EF-VFL algorithm. Step (1) concerns the model update and step (2) concerns the surrogate update.

III. Proposed Method

To solve Problem (1) with a gradient-based method, we need to perform a forward and a backward pass at each step t to compute the gradient of our objective function. In VFL, a standard uncompressed algorithm employing a single local update per communication round is mathematically equivalent to gradient descent.[3] This algorithm, which our method recovers if we set $\mathcal{C}$ to be the identity map, is as follows:

- In the forward pass, each client k computes $\boldsymbol{H}_k(\boldsymbol{x}_k^t)$ and sends it to the server, which then computes $f(\boldsymbol{x}^t)=\Phi\left(\boldsymbol{H}(\boldsymbol{x}^t)\right)$.
- In the backward pass, first, the server backpropagates through Φ, obtaining $\nabla_0\Phi(\boldsymbol{x}_0^t,\{\boldsymbol{H}_k(\boldsymbol{x}_k^t)\}_{k=1}^K)=\nabla_0 f(\boldsymbol{x}^t)$ and $\nabla_k\Phi(\{\boldsymbol{H}_k(\boldsymbol{x}_k^t)\})$, for all k, where ∇_k denotes the derivative with respect to block k. The former is used to update $\boldsymbol{x}_0^t$, while the latter is sent to each client k, which uses this derivative to continue backpropagation over its local model, allowing it to compute $\nabla_k f(\boldsymbol{x}^t)$.

We repeat these steps until convergence. Let us now cover the general case, where $\mathcal{C}$ may not be the identity map.

Forward Pass. An exact forward pass would require each client k to send $\boldsymbol{H}_k(\boldsymbol{x}_k^t)$ to the server, bringing a significant communication overhead. To address this, in our method, the server model does not have as input $\boldsymbol{H}_k(\boldsymbol{x}_k^t)$, but rather a surrogate for it, $\boldsymbol{G}_k^t$, which is initialized as $\boldsymbol{G}_k^0=\mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^0))$ and is updated as follows, as in (3):

$$\forall k\in[K]:\quad \boldsymbol{G}_k^t:=\boldsymbol{G}_k^{t-1}+\boldsymbol{C}_k^t,\quad \boldsymbol{C}_k^t:=\mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^t)-\boldsymbol{G}_k^{t-1}).$$

[3]When performing multiple local updates, we lose this mathematical equivalence. In that case, we instead get a parallel (block) coordinate descent method where the simultaneous updates use stale information about the other blocks of variables.

This requires keeping $\boldsymbol{G}_k^t$, of size $N\times E_k$, in memory at client k and at the server. (Note that $\boldsymbol{G}_k^t$ is often smaller than the local dataset $\mathcal{D}_k$.) The server thus computes the function $\Phi\left(\boldsymbol{x}_0^t,\boldsymbol{G}_1^t,\ldots,\boldsymbol{G}_K^t\right)$, which acts as a surrogate for the original objective $f(\boldsymbol{x}^t)=\Phi(\boldsymbol{x}_0^t,\boldsymbol{H}_1(\boldsymbol{x}_1^t),\ldots,\boldsymbol{H}_K(\boldsymbol{x}_K^t))$, as illustrated in Fig. 1.

Backward Pass. The server performs a backward pass over the server model, obtaining $\nabla_0\Phi(\boldsymbol{x}_0^t,\{\boldsymbol{G}_k^t\}_{k=1}^K)$, which it uses as a surrogate for $\nabla_0\Phi(\boldsymbol{x}_0^t,\{\boldsymbol{H}_k(\boldsymbol{x}_k^t)\}_{k=1}^K)=\nabla_0 f(\boldsymbol{x}^t)$ to update $\boldsymbol{x}_0^t$. Similarly, to update each local model $\boldsymbol{x}_k^t$, for $k\in[K]$, we want client k to have a surrogate for

$$\nabla_k f(\boldsymbol{x}^t)=\sum_{i,j=1}^{N,E_k}\left[\nabla_k\Phi(\{\boldsymbol{H}_k(\boldsymbol{x}_k^t)\}_{k=0}^K)\right]_{ij}\left[\nabla\boldsymbol{H}_k(\boldsymbol{x}_k^t)\right]_{ij:}.$$

However, while $\nabla\boldsymbol{H}_k(\boldsymbol{x}_k^t)$ can be computed at each client k, the $\nabla_k\Phi$ term cannot, as client k does not have access to $\boldsymbol{H}_\ell(\boldsymbol{x}_\ell^t)$, for $\ell\neq k$. So, we instead use the following surrogate for $\nabla_k f(\boldsymbol{x}^t)$:

$$\boldsymbol{g}_k^t:=\sum_{i,j=1}^{N,E_k}\left[\tilde{\nabla}_k^t\Phi\right]_{ij}\left[\nabla\boldsymbol{H}_k(\boldsymbol{x}_k^t)\right]_{ij:},\tag{4}$$

where $\tilde{\nabla}_k^t\Phi:=\nabla_k\Phi(...,\boldsymbol{G}_{k-1}^t,\boldsymbol{H}_k(\boldsymbol{x}_k^t),\boldsymbol{G}_{k+1}^t,...)$. Since the server does not hold $\boldsymbol{H}_k(\boldsymbol{x}_k^t)$, it cannot compute $\tilde{\nabla}_k^t\Phi$. Thus, the server broadcasts $\{\boldsymbol{C}_k^t\}_{k=0}^K$, so that each client k, which does hold $\boldsymbol{H}_k(\boldsymbol{x}_k^t)$, can compute $\{\boldsymbol{G}_\ell^t\}_{\ell\neq k}$ and use it to perform a forward and a backward pass over the server model locally, obtaining $\tilde{\nabla}_k^t\Phi$. Thus, while the forward pass only requires the error feedback module at each client k to hold the estimate $\boldsymbol{G}_k^t$, when we account for the backward pass too, each machine $k\in\{0,1,\ldots,K\}$ must hold $\{\boldsymbol{G}_\ell^t\}_{\ell=0}^K$. We write our update as:

$$\boldsymbol{x}^{t+1}=\boldsymbol{x}^t-\eta\boldsymbol{g}^t\quad\text{where}\quad\boldsymbol{g}^t:=(\boldsymbol{g}_0^t,\ldots,\boldsymbol{g}_K^t)$$

and η is the stepsize.

Algorithm 1: EF-VFL

Input: initial point $\boldsymbol{x}^0$, stepsize η, and initial surrogates $\{\boldsymbol{G}_k^0 = \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^0))\}$.

for $t = 0, \ldots, T-1$ **do**
- Update $\boldsymbol{x}_k^{t+1} = \boldsymbol{x}_k^t - \eta \tilde{\boldsymbol{g}}_k^t$ in parallel, for $k \in \{0, 1, \ldots, K\}$, based on a shared sample $\mathcal{B}^t \subseteq [N]$.
- Compute and send $\boldsymbol{C}_k^{t+1} = \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^{t+1}) - \boldsymbol{G}_k^t)$ to the server in parallel, for $k \in [K]$.
- Server broadcasts $\{\boldsymbol{C}_k^{t+1}\}_{k=0}^K$ to all clients.
- Update $\boldsymbol{G}_k^{t+1} = \boldsymbol{G}_k^t + \boldsymbol{C}_k^{t+1}$ in parallel, for $k \in \{0, 1, \ldots, K\}$.

Mini-Batch. For the sake of computation efficiency, we further allow for the use of mini-batch approximations of the objective. Without compression, or with direct compression, the use of mini-batches allows client k to send only the entries of $\boldsymbol{H}_k(\boldsymbol{x}_k)$ corresponding to mini-batch $\mathcal{B} \subseteq [N]$, of size B, denoted by $\boldsymbol{H}_{k\mathcal{B}}(\boldsymbol{x}_k) \in \mathbb{R}^{B \times E_k}$, instead of $\boldsymbol{H}_k(\boldsymbol{x}_k) \in \mathbb{R}^{N \times E_k}$. Yet, our method provides all machines with an estimate for all the entries of $\{\boldsymbol{H}_k(\boldsymbol{x}_k)\}$ at all times, the error feedback states $\{\boldsymbol{G}_k\}$. Therefore, our communications, needed to update $\boldsymbol{G}_k$, may not depend on N and B at all. This is determined by our choice of $\mathcal{C}$.

Our mini-batch surrogates depend on the entries of $\boldsymbol{H}_k(\boldsymbol{x}_k)$ and $\boldsymbol{G}_k$ corresponding to $\mathcal{B}$, $\boldsymbol{H}_{k\mathcal{B}}(\boldsymbol{x}_k)$ and $\boldsymbol{G}_{k\mathcal{B}}$. (Note that $\boldsymbol{H}_{0\mathcal{B}}(\boldsymbol{x}_0) = \boldsymbol{H}_0(\boldsymbol{x}_0)$ and $\boldsymbol{G}_{0\mathcal{B}}^t = \boldsymbol{G}_0^t$.) Thus, we approximate the partial derivative of the mini-batch function $f_{\mathcal{B}}(\boldsymbol{x}) = \Phi_{\mathcal{B}}(\{\boldsymbol{H}_{k\mathcal{B}}(\boldsymbol{x}_k)\}) = \frac{1}{B}\sum_{n\in\mathcal{B}} \phi_n(\boldsymbol{h}_n(\boldsymbol{x}))$ with respect to $\boldsymbol{x}_k$ by:

$$\tilde{\boldsymbol{g}}_k^t := \sum_{i\in\mathcal{B}^t}\sum_{j=1}^{E_k} \left[\tilde{\nabla}_k^t \Phi_{\mathcal{B}^t}\right]_{ij} \left[\nabla \boldsymbol{H}_{k\mathcal{B}^t}(\boldsymbol{x}_k^t)\right]_{ij:},$$

where $\tilde{\nabla}_k^t \Phi_{\mathcal{B}^t} := \nabla_k \Phi_{\mathcal{B}^t}(..., \boldsymbol{G}_{k-1,\mathcal{B}^t}^t, \boldsymbol{H}_{k\mathcal{B}^t}(\boldsymbol{x}_k^t), \boldsymbol{G}_{k+1,\mathcal{B}^t}^t, ...)$. We write the mini-batch version of the update as:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \eta \tilde{\boldsymbol{g}}^t \quad \text{where} \quad \tilde{\boldsymbol{g}}^t := \left(\tilde{\boldsymbol{g}}_0^t, \ldots, \tilde{\boldsymbol{g}}_K^t\right).$$

We now describe our method, which we summarize in Algorithm 1.

- **Initialization:** We initialize our model parameters as $\boldsymbol{x}^0$. Each machine $k \in \{0, 1, \ldots, K\}$ must hold $\boldsymbol{x}_k^0$ and our compression estimates $\{\boldsymbol{G}_k^0 = \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^0))\}_{k=0}^K$.
- **Model parameters update:** In parallel, all machines $k \in \{0, 1, \ldots, K\}$ take a (stochastic) coordinate descent step with respect to their local surrogate objective, updating $\boldsymbol{x}_k^{t+1} = \boldsymbol{x}_k^t - \eta \tilde{\boldsymbol{g}}_k^t$ based on a shared batch $\mathcal{B}^t$, sampled locally at each client following a shared seed.
- **Compressed communications:** All clients $k \in [K]$ compute $\boldsymbol{C}_k^{t+1}$ and send it to the server, who broadcasts $\{\boldsymbol{C}_k^{t+1}\}$.
- **Compression estimates update:** Lastly, all machines $k \in \{0, 1, \ldots, K\}$ use the compressed error feedback $\{\boldsymbol{C}_k^{t+1}\}$ to update their (matching) compression estimates $\{\boldsymbol{G}_k^t\}$.

Note that, in the absence of compression (that is, if $\mathcal{C}$ is the identity operator), each client k must send $\boldsymbol{C}_k^{t+1}$, of size $N \times E_k$, at each iteration. Further, the server must broadcast $\{\boldsymbol{C}_k^{t+1}\}_{k=0}^K$. Thus, the total communication complexity of EF-VFL in the absence of compression is $\mathcal{O}(N \cdot E \cdot T)$. When compression is used, $N \cdot E$ is replaced by some smaller amount which depends on the compression mechanism. For example, for top-k sparsification (defined in Section V), the communication complexity is reduced to $\mathcal{O}(k \cdot T)$.

In Algorithm 1, we formulate our method in a general setting which allows for the compression of both $\{\boldsymbol{H}_k(\boldsymbol{x}_k)\}_{k=1}^K$ and $\boldsymbol{x}_0$. This setting is covered by our analysis in Section IV. However, in general, the bottleneck lies in the uplink (client-to-server) communications, rather than the server broadcasting [39]. Therefore, our experiments in Section V focus on the compression of $\{\boldsymbol{H}_k(\boldsymbol{x}_k)\}_{k=1}^K$, rather than $\boldsymbol{x}_0$.

A. Adapting Our Method for Handling Private Labels

In this section, we propose an adaptation of EF-VFL to allow for private labels. That is, we remove the assumption that all clients hold ϕ_n, which contains the label of $\boldsymbol{\xi}_n$. Instead, only the server holds the labels. Further, in this adaptation, the parameters of the server model, $\boldsymbol{x}_0$, are not shared with the clients either.

Note that, without holding ϕ_n, the clients cannot perform the entire forward pass locally. Instead, in this setting, the forward and backward pass over ϕ_n take place at the server, while the forward and backward pass over $\boldsymbol{H}_k(\boldsymbol{x}_k)$ takes place at client k. More precisely, in the forward pass, each client k sends $\boldsymbol{C}_k$ to the server, who holds $\boldsymbol{x}_0$ and the labels, and can thus compute the loss. Then, for the backward pass, the server backpropagates over its model and sends only the derivative of the loss function with respect to $\boldsymbol{G}_k$ to each client k. This requires replacing our surrogate of $\nabla_k f(\boldsymbol{x}^t)$ in (4), which uses the exact local representation $\boldsymbol{H}_k(\boldsymbol{x}_k)$ at each client, by one based on our error feedback surrogates:

$$\boldsymbol{\nabla}_k^t := \sum_{i,j=1}^{N,E_k} \left[\nabla_k \Phi\left(\boldsymbol{H}_0(\boldsymbol{x}_0^t), \{\boldsymbol{G}_j^t\}_{j=1}^K\right)\right]_{ij} \left[\nabla \boldsymbol{H}_k(\boldsymbol{x}_k^t)\right]_{ij:}. \tag{5}$$

Note that we do not backpropagate through the error-feedback update.

More generally, we use the following (possibly) mini-batch update vector:

$$\underbrace{\sum_{i\in\mathcal{B}^t}\sum_{j=1}^{E_k} \left[\nabla_k \Phi_{\mathcal{B}^t}\left(\boldsymbol{H}_{0\mathcal{B}^t}(\boldsymbol{x}_0^t), \{\boldsymbol{G}_{j\mathcal{B}^t}^t\}_{j=1}^K\right)\right]_{ij} \left[\nabla \boldsymbol{H}_{k\mathcal{B}^t}(\boldsymbol{x}_k^t)\right]_{ij:}}_{=:\tilde{\boldsymbol{\nabla}}_k^t}.$$

We summarize the adaption of the EF-VFL method to the private labels setting in Algorithm 2.

Allowing for private labels broadens the range of applications for our method, since many VFL applications require not only private features, but also private labels, rendering any method

Algorithm 2: EF-VFL with private labels

Input: initial point $\boldsymbol{x}^0$, stepsize η, and initial surrogates $\{\boldsymbol{G}_k^0 = \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^0))\}$.

for $t = 0, \ldots, T-1$ **do**

- Update $\boldsymbol{x}_k^{t+1} = \boldsymbol{x}_k^t - \eta \tilde{\boldsymbol{\nabla}}_k^t$ in parallel, for $k \in \{0, 1, \ldots, K\}$, based on a shared sample $\mathcal{B}^t \subseteq [N]$.
- Compute and send $\boldsymbol{C}_k^{t+1} = \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^{t+1}) - \boldsymbol{G}_k^t)$ to the server in parallel, for $k \in [K]$.
- Server sends $\nabla_k \Phi_{\mathcal{B}^t}\left(\boldsymbol{H}_{0\mathcal{B}^t}(\boldsymbol{x}_0^t), \{\boldsymbol{G}_{j\mathcal{B}^t}^t\}_{j=1}^K\right)$ to client k, in parallel, for $k \in \{0, 1, \ldots, K\}$.
- Update $\boldsymbol{G}_k^{t+1} = \boldsymbol{G}_k^t + \boldsymbol{C}_k^{t+1}$ in parallel, for $k \in \{0, 1, \ldots, K\}$.

requiring public labels inapplicable. However, as we will see in Section V-B, using surrogate (5) instead of (4) can slow down convergence. Further, unlike Algorithm 1, which can be easily extendable to allow for multiple local updates at the clients between rounds of communication, Algorithm 2 works only for a single local update. This is because, for an VFL algorithm to perform multiple local updates, ϕ_n and $\boldsymbol{x}_0$ must be available at the clients, so that the forward and backward passes over the server model and the loss function can be performed locally after each update.

IV. Convergence Guarantees

In this section, we provide convergence guarantees for EF-VFL. We present our results for Algorithm 1 only, rather then stating them again for Algorithm 2, since they exhibit only a minor difference in Lemma 1 and in the main theorem, where the constant K is replaced with $K+1$. We defer the details to Appendix A.

First, let us define the following sigma-algebra

$$\mathcal{F}_t := \sigma(\boldsymbol{G}^0, \boldsymbol{x}^1, \boldsymbol{G}^1, \ldots, \boldsymbol{x}^t, \boldsymbol{G}^t),$$

where $\boldsymbol{G}^t := \{\boldsymbol{G}_0^t, \ldots, \boldsymbol{G}_K^t\}$. For the sake of conciseness, we further let $\mathbb{E}_{\mathcal{F}}$ denote the conditional expectation $\mathbb{E}[\cdot \mid \mathcal{F}]$ with sigma-algebra $\mathcal{F}$. We use the following assumptions on our stochastic update vector $\tilde{\boldsymbol{g}}^t$ and the use of mini-batches.

Assumption 3 (Unbiased): We assume that our stochastic update vector is unbiased:

$$\forall (\boldsymbol{x}, t) \in \mathbb{R}^d \times \{0, 1, \ldots, T-1\}: \quad \mathbb{E}_{\mathcal{F}_t}\left[\tilde{\boldsymbol{g}}^t\right] = \boldsymbol{g}^t. \tag{A3}$$

Assumption 4 (Bounded variance): We assume that there exists a constant $\sigma \geq 0$ such that

$$\forall (\boldsymbol{x}, t) \in \mathbb{R}^d \times \{0, 1, \ldots, T-1\}: \quad \mathbb{E}_{\mathcal{F}_t}\|\tilde{\boldsymbol{g}}^t - \boldsymbol{g}^t\|^2 \leq \frac{\sigma^2}{B}. \tag{A4}$$

We now present Lemma 1 and Lemma 2, which we will use to prove our main theorems. We let $D^{(t)} := \sum_{k=0}^K \left\|\boldsymbol{G}_k^t - \boldsymbol{H}_k(\boldsymbol{x}_k^t)\right\|^2$ denote the total distortion (caused by compression) at time t.

Lemma 1 (Surrogate offset bound): If Φ is L-smooth (A1) and $\{\boldsymbol{H}_k\}$ have bounded derivatives (A2), then, for all $t \geq 0$,

$$\|\boldsymbol{g}^t - \nabla f(\boldsymbol{x}^t)\|^2 \leq K H^2 L^2 D^{(t)}. \tag{6}$$

Proof: See Appendix B-A. □

Lemma 2 (Recursive distortion bound): Let $\{\boldsymbol{x}^t\}$ be a sequence generated by Algorithm 1. If $\mathcal{C}$ is a contractive compressor (2), $\{\boldsymbol{H}_k\}$ have bounded derivatives (A2), and (A3) and (A4) hold, then, for all $t \geq 0$ and $\epsilon > 0$:

$$\mathbb{E}D^{(t+1)} \leq (1-\alpha)(1+\epsilon)\mathbb{E}D^{(t)} + (1-\alpha)(1+\epsilon^{-1})\eta^2 H^2 \left(\mathbb{E}\|\boldsymbol{g}^t\|^2 + \frac{\sigma^2}{B}\right). \tag{7}$$

Proof: See Appendix B-B. □

A. Nonconvex Setting

We now present our main convergence result for EF-VFL.

Theorem 1: Let $\{\boldsymbol{x}^t\}$ be a sequence generated by Algorithm 1, $\mathcal{C}$ be a contractive compressor (2), and $f^\star > -\infty$. If (A1) to (A4) hold, then, for $0 < \eta \leq 1/(\sqrt{\rho_{\alpha 1}}L + L)$:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \leq \frac{2\Delta}{\eta T} + (1 + \eta L \rho_{\alpha 1})\frac{\eta L \sigma^2}{B} + \rho_{\alpha 2}\frac{\mathbb{E}D^{(0)}}{T}, \tag{8}$$

where the expectation is over the randomness in $\mathcal{C}$ and in $\{\mathcal{B}_t\}$, $\Delta := f\left(\boldsymbol{\theta}^0\right) - f^\star$, and

$$\rho_{\alpha 1} := K H^4 \left(\frac{1+\sqrt{1-\alpha}}{\alpha} - 1\right)^2 \text{ and } \rho_{\alpha 2} := \frac{K H^2 L^2}{1-\sqrt{1-\alpha}}.$$

Proof: See Appendix C-A. □

If the batch size is large enough, $B = \Omega(\sigma^2/\delta)$, the iteration complexity to reach $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2 \leq \delta$ matches the $\mathcal{O}(1/T)$ rate of the centralized, uncompressed setting. Also, in the absence of compression ($\alpha = 1$ and $D^{(t)} = 0$), we recover that, for $\eta \in (0, 1/L]$, we can output an $\boldsymbol{x}^{\text{out}}$ such that $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 \leq \frac{2\Delta}{\eta T} + \frac{\eta L \sigma^2}{B}$. If we are also in the full-batch case ($\sigma = 0$), we recover the gradient descent bound exactly: $\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 \leq \frac{2\Delta}{\eta T}$.

Note that, if we start our method by sending noncompressed representations (at $t = 0$ only), we can drop the last term in the upper bound in (8).

Our results improve over the prior state-of-the-art compressed vertical FL (CVFL) [14], whose convergence result for a fixed stepsize is presented below:

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \leq \frac{4\Delta}{\eta T} + \mathcal{O}\left(\frac{\eta L \sigma^2}{B}\right) + \mathcal{O}\left(\frac{1}{T}\sum_{t=0}^{T-1} D_d^{(t)}\right).$$

Note how, even for full-batch updates, the upper bound above does not go to zero as $T \to \infty$ unless $D_d^{(t)} \to 0$, where $D_d^{(t)}$ is the distortion resulting from direct compression $D_d^{(t)} = \sum_{k=0}^K \|\mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^t)) - \boldsymbol{H}_k(\boldsymbol{x}_k^t)\|^2$. That is, to achieve $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \to 0$, CVFL requires a vanishing compression error, which necessitates that $\alpha \to 1$. This means that, despite reducing the total amount of communications across

TABLE I
TOTAL COMMUNICATION COST TO REACH ERROR LEVEL $\delta > 0$ (TOP-k; FULL-BATCH; SINGLE LOCAL UPDATE)

$\mathcal{C}$	SVFL [19]	CVFL [12]	EF-VFL (Ours)	EF-VFL With Private Labels (Ours)
Uplink (total)	$N\bar{E}K \cdot \mathcal{O}(1/\delta)$	$\min\{k \cdot \mathcal{O}(1/\sqrt{\delta}), N\bar{E}\} \cdot K \cdot \mathcal{O}(1/\delta^2)$	$kK \cdot \mathcal{O}(1/\delta)$	$kK \cdot \mathcal{O}(1/\delta)$
Downlink (total/broadcast)	$N\bar{E}K \cdot \mathcal{O}(1/\delta)$	$\min\{k \cdot \mathcal{O}(1/\sqrt{\delta}), N\bar{E}\} \cdot (K+1) \cdot \mathcal{O}(1/\delta^2)$	$k(K+1) \cdot \mathcal{O}(1/\delta)$	$kK \cdot \mathcal{O}(1/\delta)$

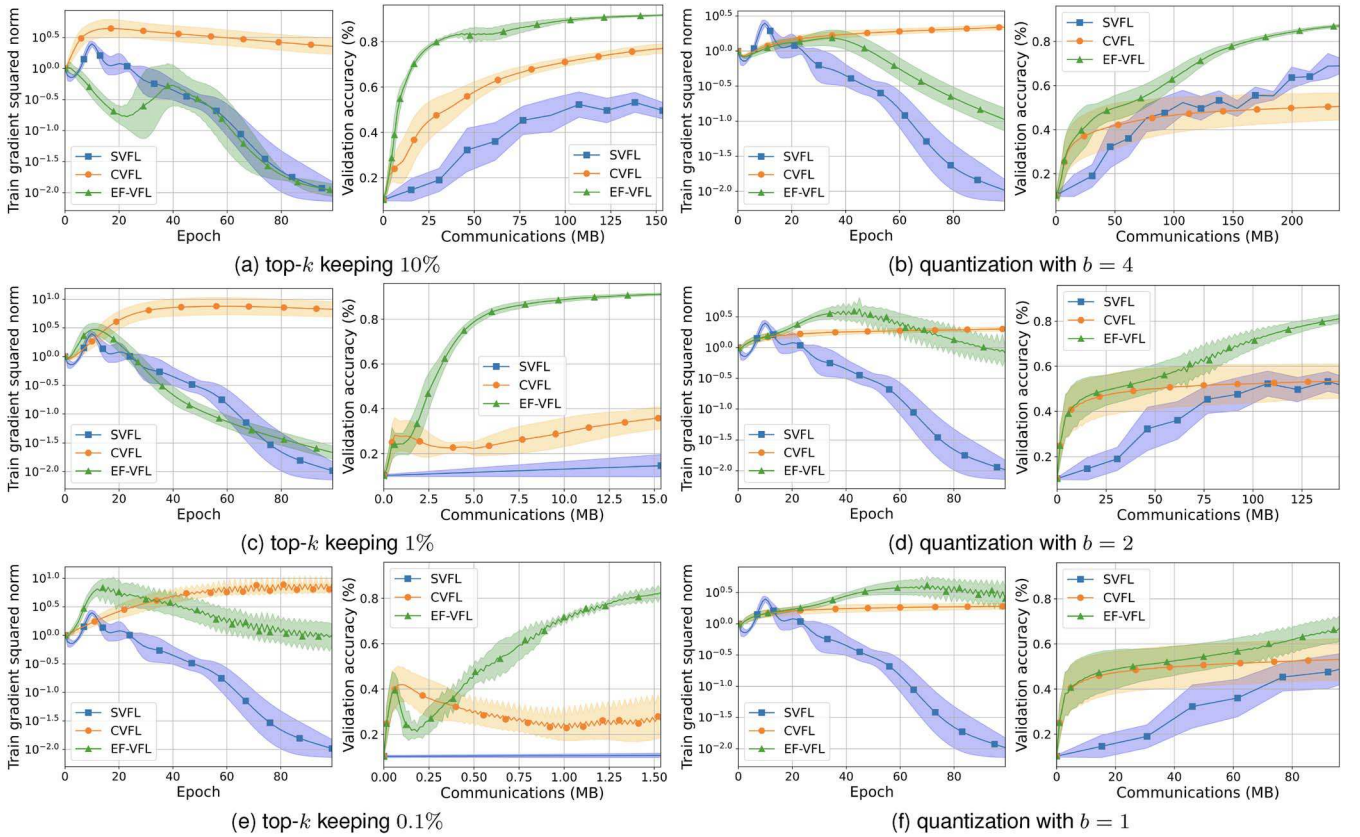


Fig. 2. The (relative) training gradient squared norm with respect to epochs and validation accuracy with respect to communication cost for the training of a shallow neural network on MNIST. On the left, CVFL and EF-VFL employ top-k sparsification with a decreasing k across rows. On the right, they employ stochastic quantization with a decreasing number of bits across rows. SVFL is the same throughout.

the training, CVFL does not reduce the maximum amount of communications per round. In contrast, by allowing for nonvanishing compression, EF-VFL ensures small communication cost at every round.

B. Under the PL Inequality

In this section, we establish the linear convergence of EF-VFL under the PL inequality [40].

Assumption 5 (PL inequality): We assume that there exists a positive constant μ such that

$$\forall \boldsymbol{x} \in \mathbb{R}^d: \quad \|\nabla f(\boldsymbol{x})\|^2 \geq 2\mu(f(\boldsymbol{x}) - f^\star). \tag{A5}$$

We resort to the following Lyapunov function to show linear convergence:

$$V_t := \mathbb{E}f(\boldsymbol{x}^t) - f^\star + c\mathbb{E}D^{(t)}, \tag{9}$$

where c is a positive constant. We now present Theorem 2.

Theorem 2 Let $\{\boldsymbol{x}^t\}$ be a sequence generated by Algorithm 1, $\mathcal{C}$ be a contractive compressor (2), and $f^\star > -\infty$. If (A1) to (A5) hold, then, for η such that $\eta^2 L^2 (1 - \mu/L) + \eta\mu \leq \alpha^2$, we have:

$$V_T \leq (1 - \eta\mu)^T V_0 + \frac{\sigma^2}{2B\mu}.$$

Proof: See Appendix C-B. □

Since $\mu \leq L$, we have that $\eta \in (0, 1/L)$ implies that $1 - \eta\mu \in (0, 1)$. Hence, EF-VFL converges linearly to a $\mathcal{O}(\sigma^2)$ neighborhood around the global optimum.

In Table I, where $\bar{E} = E_j$ for $j \in [K]$ is the embedding size for each sample at each client (assumed to match for simplicity), we present the total communication cost to reach $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f\left(\boldsymbol{\theta}^t\right)\right\|^2 \leq \delta$, where $\delta > 0$ (top-k; full-batch; single local update). We discuss different downlink communication schemes for EF-VFL in Appendix D.

V. EXPERIMENTS

We compare EF-VFL with two baselines: **(1)** standard VFL (SVFL), which corresponds to the method in [27] and is

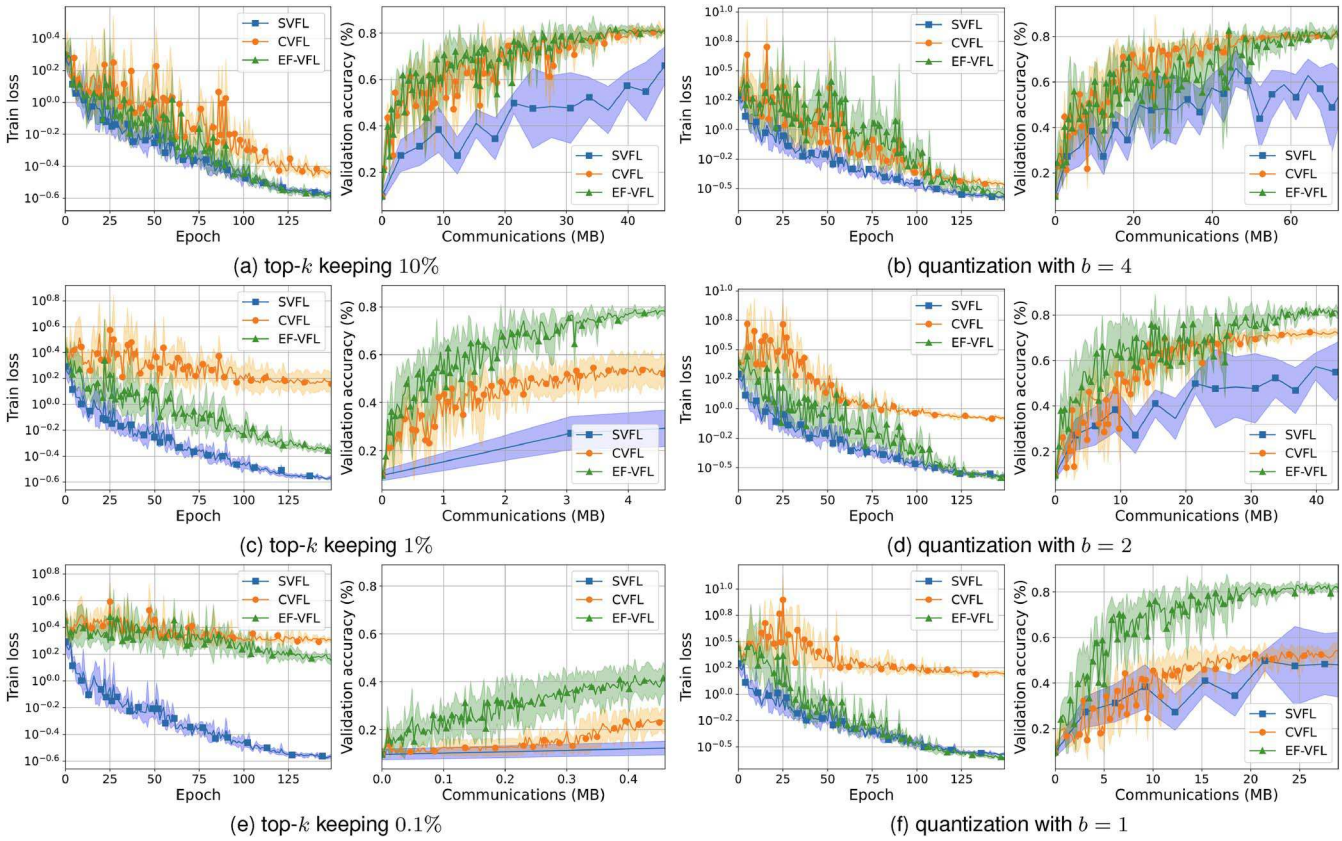


Fig. 3. Train loss with respect to the number of epochs and validation accuracy with respect to the communication cost for the training of an MVCNN on ModelNet10. On the left, CVFL and EF-VFL employ top-k sparsification with a decreasing k across rows. On the right, they employ stochastic quantization with a decreasing number of bits across rows. SVFL is the same throughout.

mathematically equivalent to stochastic gradient descent when a single local update is used, and **(2)** CVFL [14], which recovers SVFL when an identity compressor is used (that is, without employing compression). All of our results correspond to the mean and standard deviation for five different seeds. We employ two popular compressors in our experiments.

- Top-k sparsification [12], [13] is a map $\text{top}_k \colon \mathbb{R}^d \mapsto \mathbb{R}^d$ defined as

$$\text{top}_k(\boldsymbol{v}) := \boldsymbol{v} \odot \boldsymbol{u}(\boldsymbol{v}),$$

  where $\odot$ denotes the Hadamard product and $\boldsymbol{u}(\boldsymbol{v})$ is such that its entry i is 1 if v_i is one of the k largest entries of $\boldsymbol{v}$ in absolute value and 0 otherwise. We have that (2) holds for $\alpha = k/d$.
- Stochastic quantization [11] is a map $\text{qsgd}_s \colon \mathbb{R}^d \mapsto \mathbb{R}^d$, with $s > 1$ quantization levels defined as

$$\text{qsgd}_s(\boldsymbol{v}) := \frac{\|\boldsymbol{v}\| \cdot \text{sign}(\boldsymbol{v})}{s\tau} \cdot \left\lfloor s\frac{|\boldsymbol{v}|}{\|\boldsymbol{v}\|} + \xi \right\rfloor,$$

  where $\tau = 1 + \min\{d/s^2, \sqrt{d}/s\}$ and $\xi \sim \mathcal{U}([0,1]^d)$, where $\mathcal{U}$ denotes the uniform distribution. In practice, we are interested in values of s such that $s = 2^b$, where b is the number of bits. We have that (2) holds for $\alpha = 1/\tau$.

For the sake of the comparison with CVFL, we employ compressors $\mathcal{C} = \mathcal{C}_2 \circ \mathcal{C}_1$, where $\mathcal{C}_2$ is either $\text{top}_k(\boldsymbol{v})$ or qsgd_s and $\mathcal{C}_1$ selects the rows in $\mathcal{B}^t$, similarly to CVFL. Further, as explained in Section III, our experiments focus on the compression of $\{\boldsymbol{H}_k(\boldsymbol{x}_k)\}_{k=1}^K$, and not $\boldsymbol{x}_0$.

A. Comparison With SVFL and CVFL

The detailed hyperparameters for the following experiments can be found in the provided code.

MNIST. We train a shallow neural network (one hidden layer) on the MNIST digit recognition dataset [41]. The 28×28 images in the original dataset $\mathcal{D}$ are split into four local datasets $\mathcal{D}_k$ of 14×14 images, its quadrants ($K = 4$). The local models $\boldsymbol{h}_{kn}$ are maps $\boldsymbol{v} \mapsto \text{sigmoid}(\boldsymbol{W}_{k1}\boldsymbol{v})$, with $\boldsymbol{W}_{k1} \in \mathbb{R}^{128\times 196}$, and the server model is $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_4) \mapsto \boldsymbol{W}_2(\sum_{k=1}^4 \boldsymbol{v}_k)$, with $\boldsymbol{W}_2 \in \mathbb{R}^{10\times 16}$. We use cross-entropy loss. In Fig. 2, we present the results for when EF-VFL and CVFL employ top_k, keeping 10%, 1%, and 0.1% of the entries, and when they employ qsgd_s, sending $b \in \{4, 2, 1\}$ bits per entry, instead of the uncompressed $b = 32$. In both figures, we see that EF-VFL outperforms SVFL and CVFL in communication efficiency. In terms of results per epoch, EF-VFL significantly outperforms CVFL and, for a sufficiently large k (for top_k) or b (for qsgd_s) EF-VFL achieves a similar performance to SVFL. As predicted in Section IV, the train gradient squared norm during training goes to zero for EF-VFL, as it does for SVFL, but not for CVFL.

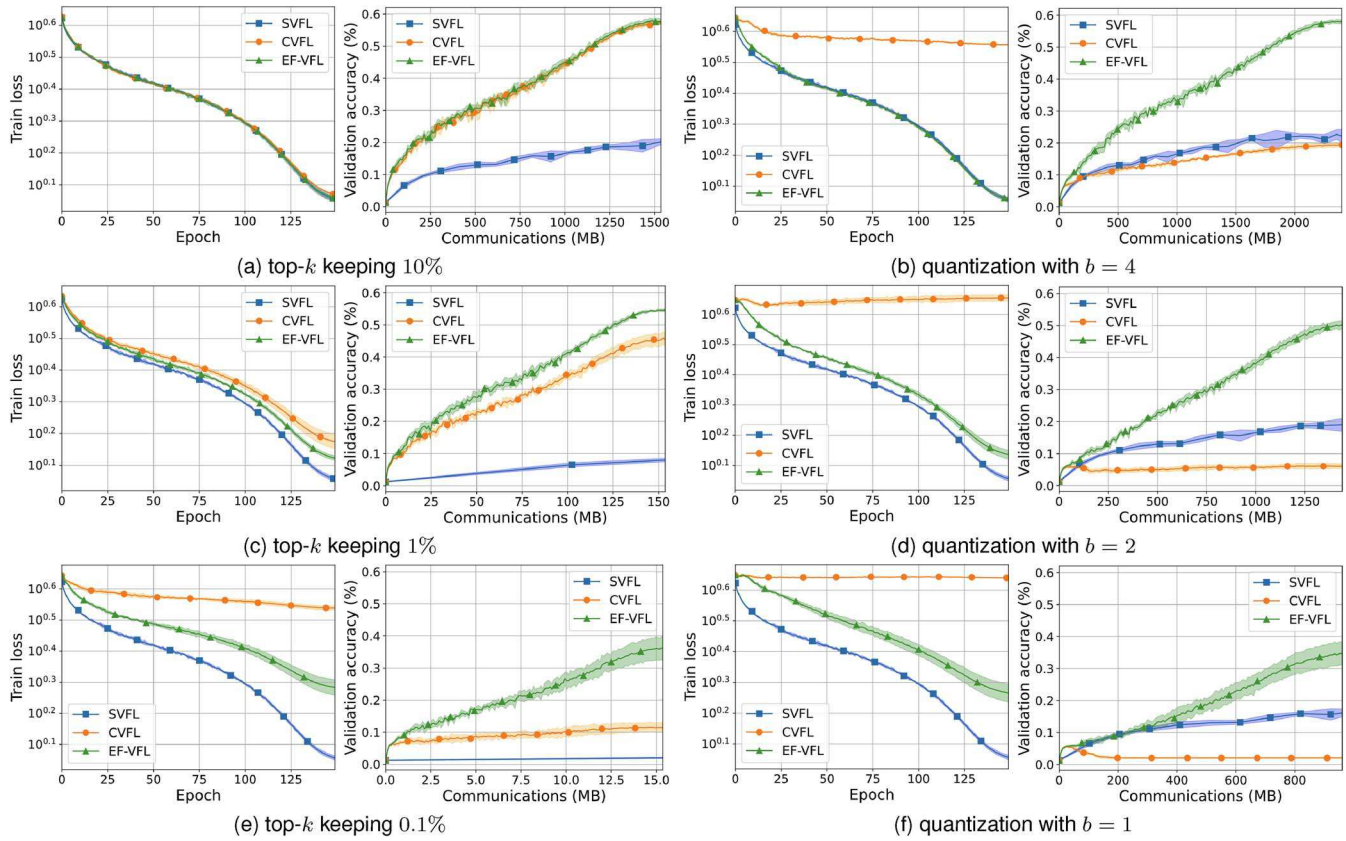


Fig. 4. Train loss with respect to the number of epochs and the validation accuracy with respect to the communication cost for the training of a ResNet18-based model on CIFAR-100. On the left, CVFL and EF-VFL employ top-k sparsification with a decreasing k across rows. On the right, they employ stochastic quantization with a decreasing number of bits across rows. SVFL is the same throughout.

ModelNet10. We train a multi-view convolutional neural network (MVCNN) [42] on ModelNet10 [43], a dataset of three-dimensional CAD models. We use a preprocessed version of ModelNet10, where each sample is represented by 12 two-dimensional views. We assign a view per client ($K = 12$). In Fig. 3, we present the results for when EF-VFL and CVFL employ top$_k$, keeping 10%, 1%, and 0.1% of the entries, and when they employ qsgd$_s$, with $b \in \{4, 2, 1\}$. We plot the train loss with respect to the number of epochs and the validation accuracy with respect to the communication cost. We observe that, for EF-VFL, the training loss decreases more rapidly than for CVFL. Further, if the compression is not excessively aggressive, EF-VFL performs similarly to SVFL. In terms of communication efficiency, EF-VFL outperforms both SVFL and CVFL.

CIFAR-100. We train a model based on a residual neural network, ResNet18 [44], on CIFAR-100 [45]. More precisely, we divide each image into four quadrants and allocate one quadrant to each client ($K = 4$), with each client using a ResNet18 model as its local model. The server model is linear (a single layer). In Fig. 4, we present the results for when EF-VFL and CVFL employ top$_k$, keeping 10%, 1%, and 0.1% of the entries, and when they employ qsgd$_s$, with $b \in \{4, 2, 1\}$. We plot the train loss with respect to the number of epochs and the validation accuracy with respect to the communication cost. Regarding the results with respect to the number of epochs, EF-VFL achieves a similar performance to that of SVFL, significantly outperforming CVFL. In terms of communication efficiency, EF-VFL outperforms both SVFL and CVFL.

We summarize the test metrics for all three tasks in Table II. In brief, while CVFL performs well for less aggressive compression is employed, the improved performance of EF-VFL is significant when more aggressive compression is employed.

B. Performance Under Private Labels

In this section, we run experiments on the adaption of EF-VFL to handle private labels, proposed in Section III-A.

In [14], the authors assume that the labels are available at all clients and do not propose an adaptation of CVFL to deal with private labels. Yet, to get a baseline for a compressed VFL method allowing for label privacy, we adapt CVFL in a similar manner to how we adapt EF-VFL (that is, sending back the derivative from the server to the clients, instead of ϕ_n and $\boldsymbol{x}_0$, and without backpropagating through the compression operator).

We run an experiment on MNIST, training the same shallow neural network as in Section V-A, with all the same settings, except for the use of batch size $B = 1024$. Further, we train a ResNet18-based model with a linear server model (a single

TABLE II
TEST ACCURACY FOR SVFL, CVFL, AND EF-VFL ACROSS DIFFERENT TASKS, FOR A FIXED NUMBER OF EPOCHS. WE RUN REACH EXPERIMENT FOR 5 SEEDS AND PRESENT THE MEAN ACCURACY ± STANDARD DEVIATION, HIGHLIGHTING THE HIGHEST ACCURACY IN BOLD

	MNIST (accuracy, %)						
		top-k compressor			qsgd compressor		
	Uncompressed	keep 10%	keep 1%	keep 0.1%	$b=4$	$b=2$	$b=1$
SVFL	91.6 ± 0.1	—	—	—	—	—	—
CVFL	—	77.2 ± 1.9	35.7 ± 5.0	25.7 ± 7.6	50.3 ± 6.1	53.0 ± 7.4	52.7 ± 8.7
EF-VFL	—	**91.8** ± 0.1	**91.1** ± 0.3	**82.4** ± 2.1	**87.2** ± 0.4	**81.1** ± 1.6	**66.8** ± 5.9
	ModelNet10 (accuracy, %)						
		top-k compressor			qsgd compressor		
	Uncompressed	keep 10%	keep 1%	keep 0.1%	$b=4$	$b=2$	$b=1$
SVFL	81.2 ± 0.8	—	—	—	—	—	—
CVFL	—	**80.7** ± 3.1	53.2 ± 6.5	24.5 ± 4.2	**80.3** ± 2.0	70.7 ± 1.6	52.0 ± 3.2
EF-VFL	—	80.4 ± 1.9	**77.4** ± 2.5	**40.3** ± 4.8	79.4 ± 3.7	**80.4** ± 2.7	**81.1** ± 2.8
	CIFAR-100 (accuracy, %)						
		top-k compressor			qsgd compressor		
	Uncompressed	keep 10%	keep 1%	keep 0.1%	$b=4$	$b=2$	$b=1$
SVFL	57.7 ± 0.6	—	—	—	—	—	—
CVFL	—	56.8 ± 0.6	45.1 ± 2.3	11.6 ± 1.4	19.2 ± 0.6	5.9 ± 0.7	2.0 ± 0.1
EF-VFL	—	**57.2** ± 0.8	**54.8** ± 0.9	**36.4** ± 2.8	**57.8** ± 0.5	**50.2** ± 1.3	**34.7** ± 2.8

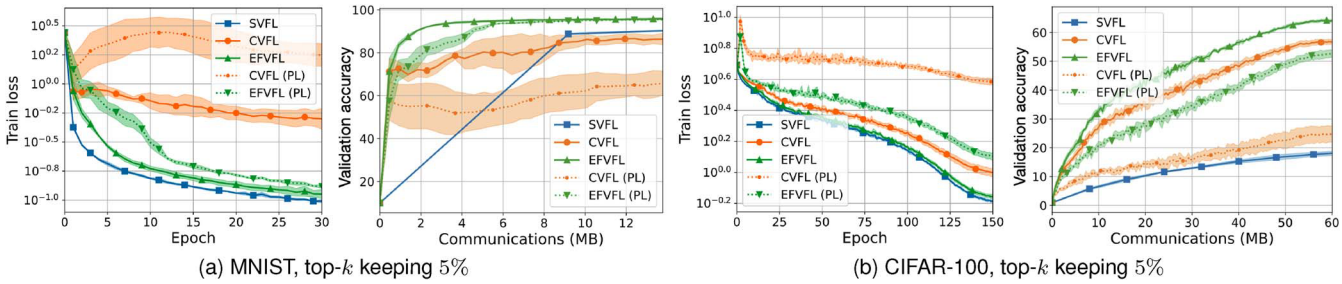


Fig. 5. Train loss with respect to the number of epochs and validation accuracy with respect to the communication cost for the training of a shallow neural network on MNIST and a ResNet18-based model on CIFAR-100. In the legend, PL stands for private labels. The communication compressed methods—CVFL, EF-VFL, CVFL (PL), and EF-VFL (PL)—employ top-k sparsification.

layer) on CIFAR-100 [45]. For all the optimizers, we use an initial stepsize of $\eta = 0.01$ and a cosine annealing scheduler with a minimum stepsize of $1/100$ the initial value and use a batch size $B = 128$ and a weight decay of 0.01. The compressed-communication methods employ top_k, keeping 5% of the entries.

In Fig. 5, we observe that, although the modified EF-VFL for handling private labels converges noticeably slower than the original method, it still performs effectively. For both the MNIST experiment and the CIFAR-100 experiment, we see that, while adapting CVFL to handle private labels leads to a severe drop in performance, EF-VFL slows down much less noticeably. In fact, for the MNIST experiment, EF-VFL with private labels still outperforms CVFL, even with public labels.

C. Performance Under Multiple Local Updates

As mentioned earlier, some VFL works employ $Q > 1$ local updates per round [27], using stale information from the other machines. We now show that, although our analysis focuses on the case where each client performs a single local update at each round of communications (that is, $Q = 1$), EF-VFL performs well in the $Q > 1$ case too. In particular, to study the performance of EF-VFL when carrying out multiple local updates, we train an MVCNN on ModelNet10 and a ResNet18 on CIFAR-10.

For ModelNet10, all three VFL optimizers use a batch size $B = 128$, a stepsize $\eta = 0.004$, and a weight decay of 0.01. Further, we use a learning rate scheduler, halving the learning rate at epochs 50 and 75. The results are presented in Fig. 6(a) and Fig. 6(b). For CIFAR-10, all three VFL optimizers use a batch size $B = 128$, a stepsize $\eta = 0.0025$, and a weight decay of 0.01. Further, we use a learning rate scheduler, halving the learning rate at epochs 40, 60, and 80. The results are presented in Fig. 6(c) and Fig. 6(d).

For both ModelNet10 and CIFAR-10, we see that, similarly to the $Q = 1$ case, our method outperforms SVFL and CVFL in communication efficiency. In terms of results per epoch, EF-VFL performs similarly to SVFL and significantly better than CVFL. Interestingly, for the CIFAR-10 task, EF-VFL even outperforms SVFL with respect to the number of epochs. We suspect this may be due to the fact that compression helps to mitigate the overly greedy nature of the parallel updates based on stale information.

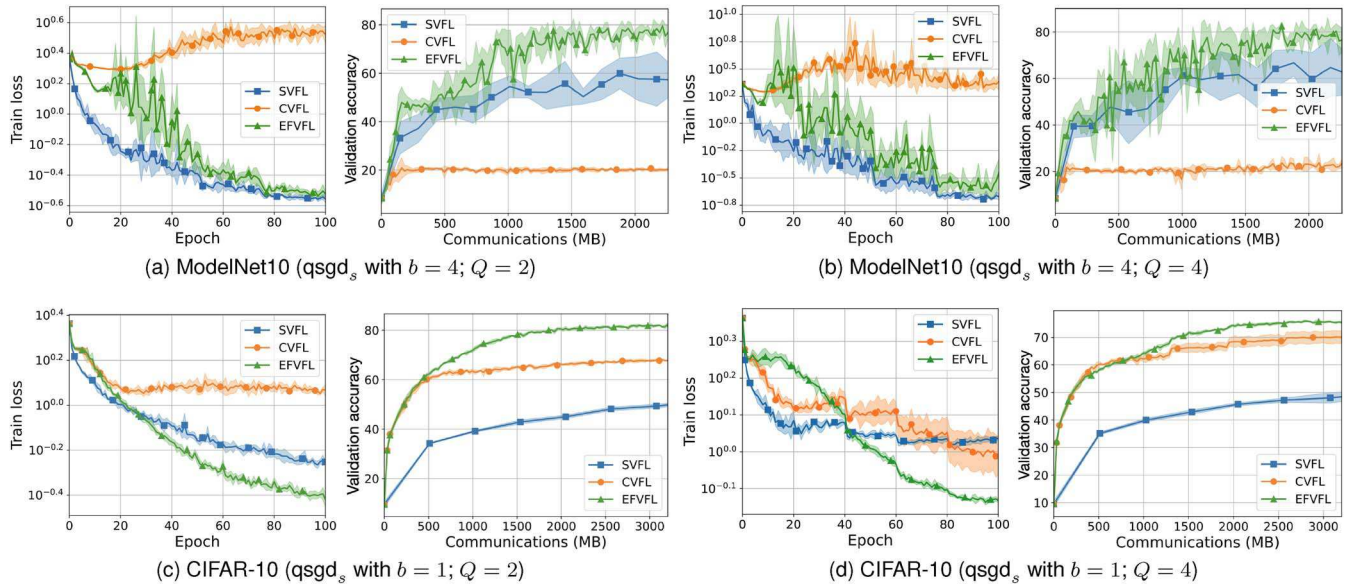


Fig. 6. Train loss with respect to the number of epochs and validation accuracy with respect to the communication cost for the training of a multi-view convolutional neural network on ModelNet10 and a residual neural network on CIFAR-10. CVFL and EF-VFL use stochastic quantization. On the left, all three vertical FL optimizers use $Q = 2$ local updates and, on the right, they all use $Q = 4$ local updates.

VI. Conclusion

In this work, we proposed EF-VFL, a method for compressed vertical federated learning. Our method leverages an error feedback mechanism to achieve a $\mathcal{O}(1/T)$ convergence rate for a sufficiently large batch size, improving upon the state-of-the-art rate of $\mathcal{O}(1/\sqrt{T})$. Numerical experiments further demonstrate the faster convergence of our method. We further show that, under the PL inequality, our method converges linearly and introduce a modification of EF-VFL supporting the use of private labels. In the future, it would be interesting to study the use of error feedback based compression methods for VFL in the fully-decentralized and semi-decentralized settings, in setups with asynchronous updates, and in combination with privacy mechanisms, such as differential privacy as done in the horizontal setting [46], [47].

Appendix A
Preliminaries

If a function is L-smooth (A1), then the following quadratic upper bound holds:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d: \quad f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{10}$$

It follows from Assumption (2) that the following inequality holds:

$$\|\boldsymbol{H}_k(\boldsymbol{x}) - \boldsymbol{H}_k(\boldsymbol{y})\| \leq H\|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall\, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_k}. \tag{11}$$

Letting $\epsilon > 0$, we use the following standard inequality in our analysis:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d: \quad \|\boldsymbol{x} + \boldsymbol{y}\|^2 \leq (1 + \epsilon)\|\boldsymbol{x}\|^2 + (1 + \epsilon^{-1})\|\boldsymbol{y}\|^2. \tag{12}$$

We define the distortion associated with block k at time t as

$$D_k^{(t)} := \left\|\boldsymbol{G}_k^t - \boldsymbol{H}_k(\boldsymbol{x}_k^t)\right\|^2$$

and, recall, we denote the total distortion at time t as $D^{(t)} = \sum_{k=0}^K D_k^{(t)}$. We also define $\nu := (1 - \alpha) \in [0, 1)$.

In Section IV, we introduced the following sigma-algebra

$$\mathcal{F}_t = \sigma(\boldsymbol{G}^0, \boldsymbol{x}^1, \boldsymbol{G}^1, \ldots, \boldsymbol{x}^t, \boldsymbol{G}^t),$$

where $\boldsymbol{G}^t = \{\boldsymbol{G}_0^t, \ldots, \boldsymbol{G}_K^t\}$. We now further define

$$\mathcal{F}_t' := \sigma(\boldsymbol{G}^0, \boldsymbol{x}^1, \boldsymbol{G}^1, \ldots, \boldsymbol{x}^t, \boldsymbol{G}^t, \boldsymbol{x}^{t+1}).$$

Recall that we let $\mathbb{E}_{\mathcal{F}}$ denote the conditional expectation $\mathbb{E}[\cdot \mid \mathcal{F}]$.

Note that, while we write our proofs for Algorithm 1, they can be easily adjusted to cover Algorithm 2. To do so, it suffices to adjust the notation, replacing $\boldsymbol{g}^t$ and $\tilde{\boldsymbol{g}}^t$ by $\boldsymbol{\nabla}_k^t$ and $\tilde{\boldsymbol{\nabla}}_k^t$, respectively, and to make minor changes to the proof of Lemma 1, which cause the constant K in Lemma 1 to be replaced with $K + 1$. This, in turn, leads to a similar adjustment in the constants of our main theorems.

Appendix B
Supporting Lemmas

A. Proof of Lemma 1

Decoupling the offset across blocks, we get that

$$\begin{aligned}
&\|\boldsymbol{g}^t - \nabla f(\boldsymbol{x}^t)\|^2 \\
&\quad = \sum_{k=0}^K \left\|\boldsymbol{g}_k^t - \nabla_k f\left(\boldsymbol{x}_k\right)\right\|^2, \\
&\quad \leq \sum_{k=0}^K \|\nabla \boldsymbol{H}_k(\boldsymbol{x}_k)\|^2 \left\|\tilde{\nabla}_k^t \Phi - \nabla_k \Phi(\{\boldsymbol{H}_k(\boldsymbol{x}_k^t)\}_{k=0}^K)\right\|^2,
\end{aligned}$$

where we use the chain rule and the fact that $\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{x}\|$. Now, it follows from the bounded gradient assumption (A2) on $\{\boldsymbol{H}_k\}$ and the L-smoothness (A1) of Φ that

$$\begin{aligned}\|\boldsymbol{g}^t - \nabla f(\boldsymbol{x}^t)\|^2 &\leq H^2 L^2 \sum_{k=0}^{K} \sum_{j\neq k} \left\|\boldsymbol{G}_j^t - \boldsymbol{H}_j(\boldsymbol{x}_j^t)\right\|^2 \\ &= H^2 L^2 \sum_{k=0}^{K} \sum_{j\neq k} D_j^{(t)} \\ &= K H^2 L^2 D^{(t)},\end{aligned}$$

as we set out to prove. For Algorithm 2, the sum $\sum_{j\neq k}$ would instead be $\sum_{j=1}^{K}$, leading to $\|\boldsymbol{g}^t - \nabla f(\boldsymbol{x}^t)\|^2 \leq (K+1) H^2 L^2 D^{(t)}$. However, note that, for Algorithm 2, $D_0^{(t)} = 0$.

B. Proof of Lemma 2

It follows from the definition of distortion and from the update of our compression estimate that

$$\begin{aligned}\mathbb{E}_{\mathcal{F}_t'}\left[D_k^{(t+1)}\right] &= \mathbb{E}_{\mathcal{F}_t'}\left\|\boldsymbol{G}_k^{t+1} - \boldsymbol{H}_k(\boldsymbol{x}_k^{t+1})\right\|^2 \\ &= \mathbb{E}_{\mathcal{F}_t'}\left\|\boldsymbol{G}_k^t + \mathcal{C}(\boldsymbol{H}_k(\boldsymbol{x}_k^{t+1}) - \boldsymbol{G}_k^t) - \boldsymbol{H}_k(\boldsymbol{x}_k^{t+1})\right\|^2.\end{aligned}$$

Now, from the definition of contractive compressor (2) and from (12), we have that

$$\begin{aligned}\mathbb{E}_{\mathcal{F}_t'}\left[D_k^{(t+1)}\right] &\leq \nu\left\|\boldsymbol{G}_k^t - \boldsymbol{H}_k(\boldsymbol{x}_k^{t+1})\right\|^2 \\ &\leq \nu(1+\epsilon)\left\|\boldsymbol{G}_k^t - \boldsymbol{H}_k(\boldsymbol{x}_k^t)\right\|^2 \\ &\quad + \nu(1+\epsilon^{-1})\left\|\boldsymbol{H}_k(\boldsymbol{x}_k^{t+1}) - \boldsymbol{H}_k(\boldsymbol{x}_k^t)\right\|^2,\end{aligned}$$

where, recall, $\nu = (1-\alpha) \in [0,1)$. Further, from the bounded gradient assumption—in particular, from (11)—we arrive at

$$\begin{aligned}\mathbb{E}_{\mathcal{F}_t'}\left[D_k^{(t+1)}\right] &\leq \nu(1+\epsilon)D_k^{(t)} + \nu(1+\epsilon^{-1})H^2\|\boldsymbol{x}_k^{t+1} - \boldsymbol{x}_k^t\|^2 \\ &= \nu(1+\epsilon)D_k^{(t)} + \nu(1+\epsilon^{-1})\eta^2 H^2\|\tilde{\boldsymbol{g}}_k^t\|^2,\end{aligned}$$

where, recall, $\tilde{\boldsymbol{g}}_k^t$ is our (possibly stochastic) update vector. Summing over $k = 0, 1, \ldots, K$ and taking the nonconditional expectation of both sides of the inequality, we get that

$$\mathbb{E}D^{(t+1)} \leq \nu(1+\epsilon)\mathbb{E}D^{(t)} + \nu(1+\epsilon^{-1})\eta^2 H^2 \mathbb{E}\|\tilde{\boldsymbol{g}}^t\|^2.$$

Lastly, using the fact that, under (A3), (A4) is equivalent to $\mathbb{E}\|\tilde{\boldsymbol{g}}^t\|^2 \leq \mathbb{E}\|\boldsymbol{g}^t\|^2 + \frac{\sigma^2}{B}$, we arrive at (7).

Appendix C
Main Theorems

First, let us define some shorthand notation for terms we will be using throughout our proof, whose expectation is with respect to the (possible) randomness in the compression across all steps:

$$\begin{aligned}\text{(compression error)} \quad \Omega_1^t &:= \mathbb{E}D^{(t)}, \\ \text{(surrogate norm)} \quad \Omega_2^t &:= \mathbb{E}\|\boldsymbol{g}^t\|^2.\end{aligned}$$

A. Proof of Theorem 1

From the L-smoothness of f—more specifically, from (10)—we have that

$$\begin{aligned}f(\boldsymbol{x}^{t+1}) - f(\boldsymbol{x}^t) &\leq \langle \nabla f(\boldsymbol{x}^t), \boldsymbol{x}^{t+1} - \boldsymbol{x}^t\rangle + \frac{L}{2}\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^t\|^2 \\ &= -\eta\langle \nabla f(\boldsymbol{x}^t), \tilde{\boldsymbol{g}}^t\rangle + \frac{\eta^2 L}{2}\|\tilde{\boldsymbol{g}}^t\|^2.\end{aligned}$$

Taking the conditional expectation over the batch selection, it follows from the unbiasedness of $\tilde{\boldsymbol{g}}^t$ (A3) that

$$\mathbb{E}_{\mathcal{F}_t} f(\boldsymbol{x}^{t+1}) - f(\boldsymbol{x}^t) \leq -\eta\langle \nabla f(\boldsymbol{x}^t), \boldsymbol{g}^t\rangle + \frac{\eta^2 L}{2}\mathbb{E}_{\mathcal{F}_t}\|\tilde{\boldsymbol{g}}^t\|^2.$$

From (A4), we have that $\mathbb{E}_{\mathcal{F}_t}\|\tilde{\boldsymbol{g}}^t - \boldsymbol{g}^t\|^2 \leq \frac{\sigma^2}{B}$, which, under (A3), is equivalent to $\mathbb{E}_{\mathcal{F}_t}\|\tilde{\boldsymbol{g}}^t\|^2 \leq \|\boldsymbol{g}^t\|^2 + \frac{\sigma^2}{B}$, so

$$\begin{aligned}&\mathbb{E}_{\mathcal{F}_t} f(\boldsymbol{x}^{t+1}) - f(\boldsymbol{x}^t) \\ &\quad\leq -\eta\langle \nabla f(\boldsymbol{x}^t), \boldsymbol{g}^t\rangle + \frac{\eta^2 L}{2}\|\boldsymbol{g}^t\|^2 + \frac{\eta^2 L\sigma^2}{2B} \\ &\quad= -\frac{\eta}{2}\|\nabla f(\boldsymbol{x}^t)\|^2 - \frac{\eta}{2}(1-\eta L)\|\boldsymbol{g}^t\|^2 \\ &\qquad+ \frac{\eta}{2}\|\boldsymbol{g}^t - \nabla f(\boldsymbol{x}^t)\|^2 + \frac{\eta^2 L\sigma^2}{2B},\end{aligned}$$

where the last equation follows from the polarization identity $\langle a, b\rangle = \frac{1}{2}(\|a\|^2 + \|b\|^2 - \|a-b\|^2)$. Now, using our surrogate offset bound (6) and taking the (non-conditional) expectation, we get that:

$$\begin{aligned}\mathbb{E}f(\boldsymbol{x}^{t+1}) - \mathbb{E}f(\boldsymbol{x}^t) &\leq -\frac{\eta}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 - \frac{\eta}{2}(1-\eta L)\mathbb{E}\|\boldsymbol{g}^t\|^2 \\ &\quad + \frac{\eta K H^2 L^2}{2}\mathbb{E}D^{(t)} + \frac{\eta^2 L\sigma^2}{2B}. \qquad (13)\end{aligned}$$

Using the Ω_1^t and Ω_2^t notation defined earlier and recalling that $\nu = (1-\alpha) \in [0,1)$, we rewrite (7) and (13), respectively, as

$$\Omega_1^{t+1} \leq \nu(1+\epsilon)\Omega_1^t + \nu(1+\epsilon^{-1})\eta^2 H^2 \Omega_2^t + \frac{\nu(1+\epsilon^{-1})\eta^2 H^2\sigma^2}{B}$$

and

$$\begin{aligned}\mathbb{E}f(\boldsymbol{x}^{t+1}) - \mathbb{E}f(\boldsymbol{x}^t) &\leq -\frac{\eta}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 + \frac{\eta K H^2 L^2}{2}\Omega_1^t \\ &\quad - \frac{\eta}{2}(1-\eta L)\Omega_2^t + \frac{\eta^2 L\sigma^2}{2B}.\end{aligned}$$

Multiplying the first inequality by a positive constant w and adding it to the second one, we get

$$\begin{aligned}&\mathbb{E}f(\boldsymbol{x}^{t+1}) - \mathbb{E}f(\boldsymbol{x}^t) + w\Omega_1^{t+1} - \psi_1(w)\Omega_1^t \\ &\quad\leq -\frac{\eta}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 + \psi_2(w)\Omega_2^t \\ &\qquad+ \left(w\nu(1+\epsilon^{-1})H^2 + \frac{L}{2}\right)\frac{\eta^2\sigma^2}{B}, \qquad (14)\end{aligned}$$

where

$$\psi_1(w) := w\nu(1+\epsilon) + \frac{\eta K H^2 L^2}{2}$$

and

$$\psi_2(w) := w\nu(1+\epsilon^{-1})\eta^2H^2 - \frac{\eta}{2}(1-\eta L).$$

Looking at (14), we see that, if $\psi_2(w) \leq 0$, we can drop the Ω_2^t term. Further, if $\psi_1(w) \leq w$, we can telescope the Ω_1^t term as we sum the inequalities for $t = 0, \ldots, T-1$, as we do for the $\mathbb{E}f(\boldsymbol{x}^t)$ terms. We thus get that:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \leq \frac{2(f(\boldsymbol{x}^0)-\mathbb{E}f(\boldsymbol{x}^T))}{\eta T} + \frac{2w(\Omega_1^0-\Omega_1^T)}{\eta T} + \left(2w\nu(1+\epsilon^{-1})H^2+L\right)\frac{\eta\sigma^2}{B}, \tag{15}$$

for

$$w \in \mathcal{W}_\epsilon := \left\{w\colon \frac{\eta K H^2L^2}{2(1-\nu(1+\epsilon))} \leq w \leq \frac{1-\eta L}{2\eta H^2\nu(1+\epsilon^{-1})}\right\},$$

where the lower bound follows from $\psi_1(w) \leq w$ and the upper bound from $\psi_2(w) \leq 0$.

Bounding η and Choosing ϵ. To ensure that $\mathcal{W}_\epsilon$ is not empty, we need

$$\eta^2\gamma(\epsilon)L^2 + \eta L \leq 1 \quad \text{where} \quad \gamma(\epsilon) := KH^4\frac{\nu(1+\epsilon^{-1})}{1-\nu(1+\epsilon)}.$$

From Lemma 5 of [24], we know that, if $a, b > 0$, then $0 \leq \eta \leq \frac{1}{\sqrt{a}+b}$ implies $a\eta^2 + b\eta \leq 1$. Thus, we can ensure that $\mathcal{W}_\epsilon$ is not empty by requiring

$$\eta \leq \frac{1}{\sqrt{\gamma(\epsilon)L^2}+L} = \left(\sqrt{\gamma(\epsilon)}L + L\right)^{-1}.$$

Further, to ensure that all $w \in \mathcal{W}_\epsilon$ are positive, we need $\nu(1+\epsilon) < 1$, which holds for $\epsilon < \frac{1-\nu}{\nu}$. Thus, to have the largest upper bound possible on the stepsize η, we want ϵ to be the solution to the following optimization problem, solved in Lemma 3 of [24]:

$$\epsilon^\star := \operatorname{argmin}_\epsilon \left\{\tilde{\gamma}(\epsilon) := \frac{\nu(1+\epsilon^{-1})}{1-\nu(1+\epsilon)} : 0 < \epsilon < \frac{1-\nu}{\nu}\right\} = \frac{1}{\sqrt{\nu}} - 1.$$

It follows that $\sqrt{\tilde{\gamma}(\epsilon^\star)} = \frac{1+\sqrt{1-\alpha}}{\alpha} - 1$ and thus $\gamma(\epsilon^\star) = KH^4\left(\frac{1+\sqrt{1-\alpha}}{\alpha} - 1\right)^2 =: \rho_{\alpha 1}$. We therefore need

$$\eta \leq \left(\sqrt{\gamma(\epsilon^\star)}L + L\right)^{-1} = \left(\sqrt{\rho_{\alpha 1}}L + L\right)^{-1}.$$

Note that, for $\alpha = 1$, we recover $\eta \leq 1/L$.

Choosing w. Now, since $f^\star \leq f(\boldsymbol{x})$ for all $\boldsymbol{x}$ and $\Omega_1^T \geq 0$, we have from (15) that, for all $w \in \mathcal{W}_\epsilon$:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \leq \frac{2\Delta}{\eta T} + \frac{2w\Omega_1^0}{\eta T} + \left(2w\nu(1+\epsilon^{-1})H^2+L\right)\frac{\eta\sigma^2}{B},$$

where $\Delta := f(\boldsymbol{x}^0) - f^\star$. From the inequality above, we see that we want $w \in \mathcal{W}_\epsilon$ to be as small as possible. Therefore, we take w to be the lower bound in $\mathcal{W}_\epsilon$. Since $1-\nu(1+\epsilon^\star) = 1-\sqrt{\nu}$, this corresponds to setting $w = \frac{\eta K H^2L^2}{2(1-\sqrt{\nu})}$. Recalling that $\Omega_1^t = \mathbb{E}D^{(t)}$ and $\nu = 1-\alpha$, we thus arrive at (8):

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \leq \frac{2\Delta}{\eta T} + \frac{KH^2L^2}{1-\sqrt{1-\alpha}}\cdot\frac{\mathbb{E}D^{(0)}}{T} + (\eta L\rho_{\alpha 1}+1)\frac{\eta L\sigma^2}{B}.$$

B. Proof of Theorem 2

Recall that, using the Ω_1^t and Ω_2^t notation, we can rewrite (7) and (13), respectively, as

$$\Omega_1^{t+1} \leq \nu(1+\epsilon)\Omega_1^t + \nu(1+\epsilon^{-1})\eta^2H^2\Omega_2^t + \frac{\nu(1+\epsilon^{-1})\eta^2H^2\sigma^2}{B}$$

and

$$\mathbb{E}f(\boldsymbol{x}^{t+1}) - \mathbb{E}f(\boldsymbol{x}^t) \leq -\frac{\eta}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 + \frac{\eta KH^2L^2}{2}\Omega_1^t - \frac{\eta}{2}(1-\eta L)\Omega_2^t + \frac{\eta^2L\sigma^2}{2B}.$$

Now, from our earlier introduced Lyapunov function (9), $V_t = \mathbb{E}f(\boldsymbol{x}^t) - f^\star + c\Omega_1^t$, we have that:

$$\begin{aligned}
V_{t+1} &= \mathbb{E}f(\boldsymbol{x}^{t+1}) - f^\star + c\Omega_1^{t+1} \\
&\overset{\text{(i)}}{\leq} \mathbb{E}f(\boldsymbol{x}^t) - f^\star - \frac{\eta}{2}\mathbb{E}\|\nabla f(\boldsymbol{x}^t)\|^2 \\
&\quad + \left(\frac{\eta KH^2L^2}{2} + c\nu(1+\epsilon)\right)\Omega_1^t + \psi_2(c)\Omega_2^t \\
&\quad + (L + 2c\nu(1+\epsilon^{-1})H^2)\frac{\eta^2\sigma^2}{2B} \\
&\overset{\text{(ii)}}{\leq} (1-\eta\mu)(\mathbb{E}f(\boldsymbol{x}^t) - f^\star) \\
&\quad + \left(\frac{\eta KH^2L^2}{2} + c\nu(1+\epsilon)\right)\Omega_1^t \\
&\quad + \psi_2(c)\Omega_2^t + (L + 2c\nu(1+\epsilon^{-1})H^2)\frac{\eta^2\sigma^2}{2B} \\
&= (1-\eta\mu)V_t \\
&\quad + \underbrace{\left(\frac{\eta KH^2L^2}{2} + c\nu(1+\epsilon) - c(1-\eta\mu)\right)}_{=:\psi_3(c)}\Omega_1^t \\
&\quad + \psi_2(c)\Omega_2^t + (L + 2c\nu(1+\epsilon^{-1})H^2)\frac{\eta^2\sigma^2}{2B},
\end{aligned}$$

where (i) follows from (7), (13), $c > 0$, and $\psi_2(w) = w\nu(1+\epsilon^{-1})\eta^2H^2 - \frac{\eta}{2}(1-\eta L)$ and (ii) follows from the PL inequality (A5). Looking the inequality above, we see that, if there is a c such that $\psi_2(c), \psi_3(c) \leq 0$, then

$$V_{t+1} \leq (1-\eta\mu)V_t + (L + 2cH^2\nu(1+\epsilon^{-1}))\frac{\eta^2\sigma^2}{2B}. \tag{16}$$

Note that, similarly to what we had in the proof for Theorem 1, $\psi_2(c) \leq 0$ corresponds to a upper bound on c, while $\psi_3(c) \leq 0$ corresponds to an lower bound on c. We therefore want $c \in \mathcal{W}'_\epsilon$, where

$$\mathcal{W}'_\epsilon := \left\{ c\colon \frac{\eta K H^2 L^2}{2(1-\nu(1+\epsilon)-\eta\mu)} \leq c \leq \frac{1-\eta L}{2\eta\nu(1+\epsilon^{-1})H^2} \right\}.$$

Recursing (16), we get

$$\begin{aligned} V_T &\leq (1-\eta\mu)^T V_0 + (L + 2cH^2\nu(1+\epsilon^{-1}))\frac{\eta^2\sigma^2}{2B}\sum_{t=0}^{T-1}(1-\eta\mu)^t \\ &\leq (1-\eta\mu)^T V_0 + \left(L + 2cH^2\nu(1+\epsilon^{-1})\right)\frac{\eta\sigma^2}{2B\mu}, \end{aligned}$$

where the second inequality follows from the sum of a geometric series, arriving at the result we set out to prove.

Choosing ϵ and Bounding η So That $\mathcal{W}'_\epsilon$ Is Nonempty. Note that the lower bound defining $\mathcal{W}'_\epsilon$ is positive if $\eta < \frac{1-\nu(1+\epsilon)}{\mu}$, where $1-\nu(1+\epsilon) > 0$ as long as $\epsilon < \frac{1-\nu}{\nu}$. Further, $\mathcal{W}'_\epsilon$ is not empty, as long as

$$\frac{\eta K H^2 L^2}{2(1-\nu(1+\epsilon)-\eta\mu)} \leq \frac{1-\eta L}{2\eta\nu(1+\epsilon^{-1})H^2},$$

which is equivalent to

$$\eta^2 L^2\left(\beta_\epsilon(\alpha)KH^4 - \mu/L\right) + \eta L(\theta_\epsilon(\alpha) + \mu/L) \leq \theta_\epsilon(\alpha),$$

where

$$\theta_\epsilon(\alpha) = 1-(1-\alpha)(1+\epsilon) \quad \text{and} \quad \beta_\epsilon(\alpha) = (1-\alpha)(1+\epsilon^{-1}).$$

If $\epsilon \leq \min\left\{\frac{1-\alpha}{\alpha}, \frac{\alpha}{1-\alpha}\right\}$, we have that $\beta_\epsilon(\alpha) \geq 1$ and $\theta_\epsilon(\alpha) \geq 0$ for all α. It follows from $\beta_\epsilon(\alpha) \geq 1$ that $\beta_\epsilon(\alpha)KH^4 \geq \ldots$, where the last inequality follows from (A2) holding for $\boldsymbol{H}_0$. We thus get that

$$\eta^2 L^2(1-\mu/L) + \eta L(\mu/L) \leq \theta_\epsilon(\alpha). \tag{17}$$

Choosing ϵ to be

$$\epsilon^\star = \begin{cases} \alpha, & 0 < \alpha \leq 1/2, \\ 1-\alpha, & 1/2 < \alpha \leq 1, \end{cases}$$

we get that

$$\theta_{\epsilon^\star}(\alpha) = \begin{cases} \alpha^2, & 0 < \alpha \leq 1/2, \\ -1+3\alpha-\alpha^2, & 1/2 < \alpha \leq 1. \end{cases}$$

Since $\alpha^2 \leq -1+3\alpha-\alpha^2$ for $\alpha \in (1/2, 1]$, we have that $\alpha^2 \leq \theta_{\epsilon^\star}(\alpha)$ for all $\alpha \in (0, 1]$. Thus, (17) holds if

$$\eta^2 L^2(1-\mu/L) + \eta L(\mu/L) \leq \alpha^2. \tag{18}$$

Further, from (A1) and (A5), we get that $0 \leq \mu/L \leq 1$. Therefore, for a sufficiently small η, there exists a positive $c \in \mathcal{W}'_\epsilon$ such that $\psi_2(c), \psi_3(c) \leq 0$. Lastly, note that we can also guarantee that $\eta < \frac{1-\nu(1+\epsilon^\star)}{\mu} = \theta_{\epsilon^\star}(\alpha)/\mu$ by having $\eta < \alpha^2/\mu$, which follows from (18).

Choosing c to Minimize the Upper Bound. From (17), we see that we want c to be as small as possible. So, we choose c as the lower bound in the definition of $\mathcal{W}'_\epsilon$, arriving at

$$\begin{aligned} V_T &\leq (1-\eta\mu)^T V_0 \\ &\quad + \left(L + 2\left(\frac{\eta K H^2 L^2}{2(1-\nu(1+\epsilon)-\eta\mu)}\right)H^2\nu\left(1+\epsilon^{-1}\right)\right)\frac{\eta\sigma^2}{2B\mu}. \end{aligned}$$

Now, we know that, for $\epsilon = \epsilon^\star$ and $\eta^2 L^2(1-\mu/L) + \eta\mu \leq \alpha^2$, the lower bound in the definition of $\mathcal{W}'_\epsilon$ is less than or equal to the upper bound. We therefore have that

$$2\left(\frac{\eta K H^2 L^2}{2(1-\nu(1+\epsilon)-\eta\mu)}\right)H^2\nu\left(1+\epsilon^{-1}\right) \leq \frac{1-\eta L}{\eta}.$$

Using this inequality in the bound above it follows that

$$V_T \leq (1-\eta\mu)^T V_0 + \frac{\sigma^2}{2B\mu},$$

thus arriving at the statement that we set out to prove.

Appendix D
Comparison of Different Downlink Communication Schemes

As in most communication-compressed optimization literature [39], our primary concern is uplink communications, which are typically the main bottleneck in training. Nevertheless, this appendix discusses three alternative downlink communication schemes in `EF-VFL`: **1)** the one in Algorithm 1, **2)** the one in Algorithm 2, and **3)** a modified version of the one in Algorithm 1 for common VFL fusion models, enabling broadcasts of a size that is independent of the number of clients, K. Approaches **1)** and **3)** are mathematically equivalent, yet Approach **2)** is not, as discussed earlier. Each approach has its pros and cons, making it suitable for different applications. For simplicity, this discussion focuses on top-k sparsification and the full-batch case.

1) In Algorithm 1, each round of downlink communications consists of a broadcast of size $k(K+1)$—a compressed object of size k for each client (the intermediate representations) and one for the server (the fusion model).

2) In Algorithm 2, each client receives only the derivative of the loss function with respect to its representation, resulting in a total downlink communication cost of kK. Recall that this is only an option when performing a single local update.

Approach **2)** avoids the dependency of the downlink communications to each client on K, seen in Approach **1)**, but requires K different communications (one to each client), rather than a single broadcast, thus the total communication cost still depends on K. The one-to-many nature of broadcasting makes it is more appropriate to compare broadcasted information with the total downlink communications, rather than the communication to a single client, as the latter ignores the cost of contacting the other $K-1$ clients.

3) To ensure that the downlink communication cost does not depend on K, we can often exploit the structure of the fusion model ϕ. In particular, a common choice is $\phi(\boldsymbol{x}) =$

$\phi_2(\boldsymbol{x}_0, \phi_1(\boldsymbol{H}_1(\boldsymbol{x}_1), \ldots, \boldsymbol{H}_K(\boldsymbol{x}_K)))$, where ϕ_1 is a nonparameterized representation aggregator, such as a sum or an average, and ϕ_2 is a map parameterized by $\boldsymbol{x}_0$. In this case, instead of broadcasting $K+1$ objects, as in Approach **1)**, the server can broadcast the aggregation of the representations, $\phi_1(\{\boldsymbol{H}_j(\boldsymbol{x}_j^t)\})$. This allows us to collapse the dimension of length K, as long as each client i can replace $\boldsymbol{H}_i(\boldsymbol{x}_i^t)$ with $\boldsymbol{H}_i(\boldsymbol{x}_i^{t+1})$ in $\phi_1(\{\boldsymbol{H}_j(\boldsymbol{x}_j^t)\})$ using its local knowledge of its own representation. For example, if ϕ_1 is a sum, client i can subtract its previous intermediate representation and add the updated one to obtain an updated aggregation. This allows client i to perform forward and backward passes over both its local model and the fusion model, and thus perform multiple local updates without requiring further communications. Yet, this downlink communication of the aggregated representations will no longer be in the range of the compressor. For example, if $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are within the range of top-k, their sum, $\boldsymbol{v}_1 + \boldsymbol{v}_2$, will generally not be. Therefore, we have a broadcast of up to size $N\bar{E} + d_0$, where d_0 is the size of the parameters of the fusion model. That is, we avoid the dependency on K, but this comes at the cost of losing the compressed nature of the downlink communications. (This sum may still lie in a lower-dimensional manifold, but this typically recovers the dependency on K, e.g., for top-k sparsification, we can upper bound the number of nonzero entries of the sum of K k-sparse vectors by $\min\{kK, N\bar{E}\} + d_0$.) Like Approach **1)**, Approach **3)** does not allow for private labels.

We present Approach **1)** in Algorithm 1, rather than Approach **3)**, because most VFL applications are in the cross-silo setting [5] and thus the number of clients K is small, therefore $k(K+1) \ll N\bar{E} + d_0$. Yet, for applications where K is large, Approach **3)** may be preferable.

Data Availability Statement

The code for this work can be found at https://github.com/Valdeira/EF-VFL.

References

[1] P. Valdeira, J. Xavier, C. Soares, and Y. Chi, "Communication-efficient vertical federated learning via compressed error feedback," in *Proc. 32nd Eur. Signal Process. Conf. (EUSIPCO)*, 2024, pp. 1037–1041.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, PMLR, 2017, pp. 1273–1282.

[3] X. Zhang, M. Hong, S. Dhople, W. Yin, and Y. Liu, "FedPD: A federated learning framework with adaptivity to non-IID data," *IEEE Trans. Signal Process.*, vol. 69, pp. 6055–6070, 2021.

[4] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "Over-the-air federated learning from heterogeneous data," *IEEE Trans. Signal Process.*, vol. 69, pp. 3796–3811, 2021.

[5] Y. Liu et al., "Vertical federated learning: Concepts, advances, and challenges," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3615–3634, Jul. 2024.

[6] Y. Cheng, Y. Liu, T. Chen, and Q. Yang, "Federated learning for privacy-preserving AI," *Commun. ACM*, vol. 63, no. 12, pp. 33–36, 2020.

[7] I. Ceballos et al., "SplitNN-driven vertical partitioning," 2020, *arXiv: 2008.04137*.

[8] J. Dean et al., "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1223–1231.

[9] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5336–5346.

[10] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. 15th Annu. Conf. Int. Speech Commun. Assoc.*, 2014, pp. 1058–1062.

[11] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1707–1718.

[12] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 5977–5987.

[13] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 4452–4463.

[14] T. J. Castiglia, A. Das, S. Wang, and S. Patterson, "Compressed-VFL: Communication-efficient learning with vertically partitioned data," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 2738–2766.

[15] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.

[16] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4289–4305, Aug. 2012.

[17] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[18] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, 2020.

[19] Y. Du, S. Yang, and K. Huang, "High-dimensional stochastic gradient quantization for communication-efficient edge learning," *IEEE Trans. Signal Process.*, vol. 68, pp. 2128–2142, 2020.

[20] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 500–514, 2020.

[21] R. Nassif, S. Vlaski, M. Carpentiero, V. Matta, M. Antonini, and A. H. Sayed, "Quantization for decentralized learning under subspace constraints," *IEEE Trans. Signal Process.*, vol. 71, pp. 2320–2335, 2023.

[22] A. Sapio et al., "Scaling distributed machine learning with in-network aggregation," in *Proc. 18th USENIX Symp. Netw. Syst. Des. Implement. (NSDI)*, 2021, pp. 785–808.

[23] A. Beznosikov, S. Horváth, P. Richtárik, and M. Safaryan, "On biased compression for distributed learning," *J. Mach. Learn. Res.*, vol. 24, no. 276, pp. 1–50, 2023.

[24] P. Richtárik, I. Sokolov, and I. Fatkhullin, "EF21: A new, simpler, theoretically better, and practically faster error feedback," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 4384–4396.

[25] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 3478–3487.

[26] H. Zhao, B. Li, Z. Li, P. Richtárik, and Y. Chi, "Beer: Fast $O(1/T)$ rate for decentralized nonconvex optimization with communication compression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 31653–31667.

[27] Y. Liu et al., "FedBCD: A communication-efficient collaborative learning framework for distributed features," *IEEE Trans. Signal Process.*, vol. 70, pp. 4277–4290, 2022.

[28] T. Chen, X. Jin, Y. Sun, and W. Yin, "VAFL: A method of vertical asynchronous federated learning," 2020, *arXiv:2007.06081*.

[29] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 977–992, Feb. 2019.

[30] P. Valdeira, Y. Chi, C. Soares, and J. Xavier, "A multi-token coordinate descent method for semi-decentralized vertical federated learning," 2023, *arXiv:2309.09977*.

[31] A. Khan, M. ten Thij, and A. Wilbik, "Communication-efficient vertical federated learning," *Algorithms*, vol. 15, no. 8, p. 273, 2022.

[32] M. Li, Y. Chen, Y. Wang, and Y. Pan, "Efficient asynchronous vertical federated learning via gradient prediction and double-end sparse compression," in *Proc. 16th Int. Conf. Control, Automat., Robot. Vis. (ICARCV)*, Piscataway, NJ, USA: IEEE, 2020, pp. 291–296.

[33] L. Yang et al., "A survey on vertical federated learning: From a layered perspective," 2023, *arXiv:2304.01829*.

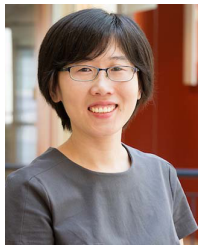
[34] H. Inose, Y. Yasuda, and J. Murakami, "A telemetering system by code modulation-δ-σmodulation," *IRE Trans. Space Electron. Telemetry*, no. 3, pp. 204–209, 1962.

[35] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signsgd and other gradient compression schemes," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 3252–3261.
[36] K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik, "Distributed learning with compressed gradient differences," *Optim. Methods Softw.*, pp. 1–16, 2024.
[37] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 2232–2240.
[38] T. Castiglia, S. Wang, and S. Patterson, "Flexible vertical federated learning with heterogeneous parties," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17878–17892, Dec. 2024.
[39] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2021, pp. 2350–2358.
[40] B. T. Polyak, "Gradient methods for the minimisation of functionals," *USSR Comput. Math. Math. Phys.*, vol. 3, no. 4, pp. 864–878, 1963.
[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[42] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.
[43] Z. Wu, S. Song et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
[46] Z. Li, H. Zhao, B. Li, and Y. Chi, "SoteriaFL: A unified framework for private federated learning with communication compression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 4285–4300.
[47] B. Li and Y. Chi, "Convergence and privacy of decentralized nonconvex optimization with gradient clipping and communication compression," 2023, *arXiv:2305.09896*.

Pedro Valdeira received the M.Sc. degree in aerospace engineering from the Instituto Superior Técnico (IST), University of Lisbon, Portugal, in 2019. He has been working toward the dual Ph.D. degree in electrical and computer engineering with Carnegie Mellon University, USA, and IST, since 2020. His research interests include machine learning and optimization, focusing on efficient optimization methods for distributed machine learning systems.

João Xavier received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Lisbon, Portugal, in 2002. Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, IST. He is also a Researcher with the Institute of Systems and Robotics, Lisbon. His research interests include optimization and statistical inference for distributed systems.

Cláudia Soares received the Diploma in modern languages and literature from Nova University of Lisbon, Lisbon, Portugal, and the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico, Lisbon, Portugal. Currently, she is an Assistant Professor with the Department of Computer Science, NOVA School of Science and Technology, and a Researcher with NOVA LINCS, Portugal. She uses real-world data problems to identify shortcomings of current machine learning, data science, and big data methods. She applies optimization, statistics, and probability theory to address those gaps, developing robust and interpretable learning methods that can be trusted in real life. Her application areas are environmental and urban sciences, healthcare, transportation, and space.



Yuejie Chi (Fellow, IEEE) received the B.E. (Hons.) degree in electrical engineering from Tsinghua University, Beijing, China, in 2007, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, in 2012 and 2009, respectively. Currently, she is the Sense of Wonder Group Endowed Professor in electrical and computer engineering of artificial intelligence (AI) systems with Carnegie Mellon University, with courtesy appointments with the Machine Learning Department and CyLab. Her research interests include the theoretical and algorithmic foundations of data science, signal processing, machine learning, and inverse problems, with applications in sensing, imaging, decision making, and AI systems. Among others, she was a recipient of the Presidential Early Career Award for Scientists and Engineers (PECASE), the SIAM Activity Group on Imaging Science Best Paper Prize, the IEEE Signal Processing Society Young Author Best Paper Award, and the inaugural IEEE Signal Processing Society Early Career Technical Achievement Award for contributions to high-dimensional structured signal processing. She was named a Goldsmith Lecturer by the IEEE Information Theory Society, a Distinguished Lecturer by the IEEE Signal Processing Society, and a Distinguished Speaker by ACM. She currently serves or served as an Associate Editor of IEEE TRANSACTIONS ON INFORMATION THEORY, IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON PATTERN RECOGNITION AND MACHINE INTELLIGENCE, *Information and Inference: A Journal of the IMA*, and *SIAM Journal on Mathematics of Data Science*.