VERTICAL FEDERATED LEARNING WITH MISSING FEATURES DURING TRAINING AND INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Vertical federated learning trains models from feature-partitioned datasets across multiple clients, who collaborate without sharing their local data. Standard approaches assume that all feature partitions are available during both training and inference. Yet, in practice, this assumption rarely holds, as for many samples only a subset of the clients observe their partition. However, not utilizing incomplete samples during training harms generalization, and not supporting them during inference limits the utility of the model. Moreover, if any client leaves the federation after training, its partition becomes unavailable, rendering the learned model unusable. Missing feature blocks are therefore a key challenge limiting the applicability of vertical federated learning in real-world scenarios. To address this, we propose `LASER-VFL`, a vertical federated learning method for efficient training and inference of split neural network-based models that is capable of handling arbitrary sets of partitions. Our approach is simple yet effective, relying on the strategic sharing of model parameters and on task-sampling to train a family of predictors. We show that `LASER-VFL` achieves a $\mathcal{O}(1/\sqrt{T})$ convergence rate for nonconvex objectives in general, $\mathcal{O}(1/T)$ for sufficiently large batch sizes, and linear convergence under the Polyak-Łojasiewicz inequality. Numerical experiments show improved performance of `LASER-VFL` over the baselines. Remarkably, this is the case even in the absence of missing features. For example, for CIFAR-100, we see an improvement in accuracy of 18.2% when each of four feature blocks is observed with a probability of 0.5 and of 7.4% when all features are observed.

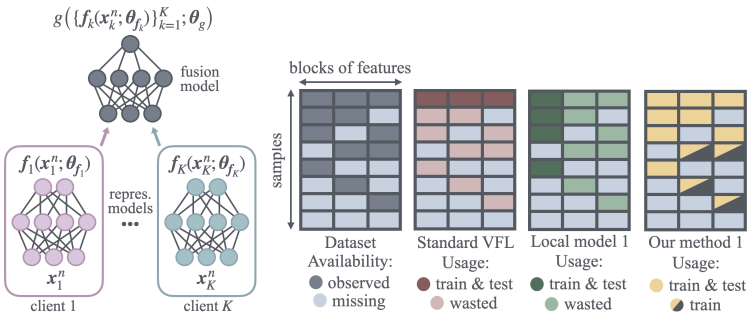
1 INTRODUCTION

In federated learning (FL), a set of clients collaborates to jointly train a model using their local data without sharing it (Kairouz et al., 2021). In horizontal FL, data is distributed by samples, meaning each client holds a different set of samples but shares the same feature space. In contrast, vertical FL (VFL) involves data distributed by features, where each client holds different parts of the feature space for overlapping sets of samples. Whether an application is horizontal or vertical FL is dictated by how the data arises, as it is not possible to redistribute the data. This work focuses on VFL.

In VFL (Liu et al., 2024), the global dataset $\mathcal{D} \coloneqq \{\boldsymbol{x}^1, \dots, \boldsymbol{x}^N\}$, where N is the number of samples, is partitioned across clients $\mathcal{K} \coloneqq \{1, \dots, K\}$. Each client $k \in \mathcal{K}$ typically holds a local dataset $\mathcal{D}_k \coloneqq \{\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^N\}$, where $\boldsymbol{x}_k^n$ is the block of features of sample n observed by client k. We have that $\boldsymbol{x}^n = (\boldsymbol{x}_1^n, \dots, \boldsymbol{x}_K^n)$. Unlike horizontal FL, in VFL, different clients collect local datasets with distinct types of information (features). Such setups—e.g., an online retail company and a social media platform holding different features on shared users—typically involve entities from different sectors, reducing competition and increasing the incentive to collaborate. To train VFL models without sharing the local datasets, split neural networks (Ceballos et al., 2020) are often considered.

In split neural networks, each client k has a *representation model* $\boldsymbol{f}_k$, parameterized by $\boldsymbol{\theta}_{\boldsymbol{f}_k}$. The representations extracted by the clients are then used as input to a *fusion model* g, parameterized by $\boldsymbol{\theta}_g$. This fusion model can be at one of the clients or at a server. Thus, to learn the parameters $\boldsymbol{\theta}_{\mathcal{K}} \coloneqq (\boldsymbol{\theta}_{\boldsymbol{f}_1}, \dots, \boldsymbol{\theta}_{\boldsymbol{f}_K}, \boldsymbol{\theta}_g)$ of the resulting *predictor* h, we can solve the following problem:

$$\min_{\boldsymbol{\theta}_{\mathcal{K}}} \quad \frac{1}{N} \sum_{n=1}^{N} \ell(h(\boldsymbol{x}^n; \boldsymbol{\theta}_{\mathcal{K}}), y^n) \quad \text{where} \quad h(\boldsymbol{x}^n; \boldsymbol{\theta}_{\mathcal{K}}) \coloneqq g\left(\{\boldsymbol{f}_k(\boldsymbol{x}_k^n; \boldsymbol{\theta}_{\boldsymbol{f}_k})\}_{k=1}^{K}; \boldsymbol{\theta}_g\right), \tag{1}$$



Method	p_{miss} 0.0	p_{miss} 0.5
	(F_1-score×100)	
`Local`	74.9	73.0
`Standard VFL`	91.6	51.8
`LASER-VFL` (ours)	**92.0**	**82.0**

(a) Split neural network. (b) Dataset availability and usage. (c) Results preview.

Figure 1: In Figure 1a, we illustrate a split neural network. In Figure 1b, we show the availability of a dataset with $K = 3$ blocks of features, each with a 0.5 probability of being observed, and its usage and waste by three key methods: standard VFL, a local approach, and our method. Standard VFL trains a single predictor, while the local approach and our method train different predictors at different clients (we show the data usage for client 1). In Figure 1c, we present a preview of our results: a mortality prediction task using the MIMIC-IV dataset, when the probability of each block of features missing, p_{miss}, is in $\{0.0, 0.5\}$, for both data and test data.

with ℓ denoting a loss function and y^n the label of sample n, which we assume to be held by the same entity (client or server) as the fusion model. We illustrate this family of models in Figure 1a.

Generalization and availability in standard VFL. We see in (1) that, even for a single sample n, predictor h depends on all the blocks of features, $\{\boldsymbol{x}_k^n : \forall k\}$. Thus, for both training and inference, (1) requires the observations of all the clients to be available.[1] Building on our example of an online retailer and a social media company sharing users, each company is also likely to have unique users. This applies to both training and test data. Further, if any client drops from the federation during inference, its block will permanently stop being observed. In both the case of nonshared users and of clients leaving the federation, standard VFL predictors become unusable. Figure 1b illustrates this phenomenon: for both training and inference, if each of $K = 3$ clients observes its block of features with an (independent) probability of 0.5, only $0.5^3 = 12.5\%$ of the original data is usable. Thus, restricting training to fully-observed samples hinders generalization, while restricting inference to such samples limits the availability and utility of the model.

Dealing with missing features. To handle missing features in training data, some approaches expand the dataset, filling the missing features before collaborative training (Kang et al., 2022b), while others use their partition of partially-observed samples for local representation learning but exclude them from collaboration (He et al., 2024). These methods can outperform standard VFL when partially observed samples are present, but they add new training stages and auxiliary modules, making them more complex. Moreover, they train a joint predictor that requires the collaboration of all clients during inference. On the other hand, to improve robustness against missing blocks at inference, each client can train a local predictor using only its own features, avoiding collaboration altogether. This also addresses missing features in the training data. Alternatively, collaborative methods can employ techniques such as knowledge distillation (Huang et al., 2023) and data augmentation (Gao et al., 2024) to train local predictors, also adding complexity with multiple stages. However, while robust to missing features, these local predictors cannot utilize additional feature blocks if available at test time, resulting in wasted data and reduced predictive power. We illustrate this in Figure 1b.

Therefore, there is a gap in the literature when it comes to leveraging all the available data without either dropping incomplete samples or ignoring existing features. This raises the following question:

Can we design an efficient VFL method that is robust to missing features at both training and inference with provable convergence guarantees, without wasting data?

[1]This contrasts with horizontal FL, where the local data of one client suffices to estimate mini-batch gradients and to perform inference.

In this work, we answer this question in the affirmative. We propose LASER-VFL (**L**everaging **A**ll **S**amples **E**fficiently for **R**eal-world **VFL**), a novel method that enables both training and inference using any and all available blocks of features. To the best of our knowledge, this is the first method to achieve this. Our approach avoids the multi-stage pipelines that are common in this area, providing a simple yet effective solution that leverages strategic sharing of model parameters alongside a task-sampling mechanism. By fully utilizing all data (see Figure 1b), our method leads to significant performance improvements, as demonstrated in Figure 1c. Remarkably, our approach even surpasses standard VFL when all samples are fully observed. We attribute this to a dropout-like regularization effect introduced by task sampling.

Our contributions. The main contributions of this work are as follows.

- We propose LASER-VFL, a simple, hyperparameter-free, and efficient VFL method that is flexible to a varying number of feature blocks during both training and inference. To the best of our knowledge, this is the first method to achieve such flexibility and avoid wasting either training data or test data.
- We show that LASER-VFL converges at a $\mathcal{O}(1/\sqrt{T})$ rate for nonconvex objectives in general, and at a $\mathcal{O}(1/T)$ rate for a sufficiently large batch size. Further, under the Polyak-Łojasiewicz (PL) inequality, we achieve linear convergence.
- Numerical experiments show that LASER-VFL consistently outperforms baselines across multiple datasets and varying data availability patterns. It demonstrates superior robustness to missing features and, notably, when all features are available, it still outperforms even standard VFL.

Related work. VFL shares challenges with horizontal FL, such as communication efficiency (Liu et al., 2022; Valdeira et al., 2024) and privacy preservation (Yu et al., 2024), but also faces unique obstacles, such as missing feature blocks. During training, these unavailable partitions render the observed blocks of other clients unusable, and at test time, they can prevent inference altogether. Therefore, most VFL literature assumes that all features are available for both training and inference—an often unrealistic assumption that has hindered broader adoption of VFL (Liu et al., 2024).

To address the problem of missing features during VFL training, some works use nonoverlapping, or nonaligned, samples (that is, samples with missing feature blocks) to improve generalization. In particular, Feng (2022) and He et al. (2024) apply self-supervised learning to leverage nonaligned samples locally for better representation learning, while using overlapping samples for collaborative training of a joint predictor. Alternatively, Kang et al. (2022b), Yang et al. (2022), and Sun et al. (2023) employ semi-supervised learning to take advantage of nonaligned samples. Although these methods enable VFL to utilize data that conventional approaches would discard, they still train a joint predictor and thus require all features to be available for inference, which remains a limitation.

A recent line of research leverages information from the entire federation to train local predictors. This can be achieved via transfer learning, as in the works by Feng & Yu (2020); Kang et al. (2022a) for overlapping training samples and in the works by Liu et al. (2020); Feng et al. (2022) for handling missing features. Knowledge distillation is another approach, as used by Ren et al. (2022); Li et al. (2023b) for overlapping samples and by Li et al. (2023a); Huang et al. (2023) which leverage nonaligned samples when training local predictors (only the latter considers scenarios with more than two clients). Xiao et al. (2024) recently proposed using a distributed generative adversarial network for collaborative training on nonoverlapping data and synthetic data generation. These methods yield local predictors that outperform naive local approaches but fail to utilize valuable information when other feature blocks are available during inference.

A few recent works enable a varying number of clients to collaborate during inference. Sun et al. (2024) employ party-wise dropout during training to mitigate performance drops from missing feature blocks at inference; Gao et al. (2024) introduce a multi-stage approach with complementary knowledge distillation, enabling inference with different client subsets; and Ganguli et al. (2024) deal with the related task of handling communication failure during inference in cross-device VFL. However, all of these methods require fully-observed training data and lack convergence guarantees.

In contrast to prior work, LASER-VFL can handle any subset of feature blocks being present during both training and inference without wasting data, while requiring only minor modifications to the standard VFL approach. It differs from conventional VFL solely in its use of strategic sharing of model parameters and task-sampling mechanisms, without the need for additional stages.

2 DEFINITIONS AND PRELIMINARIES

Our global dataset $\mathcal{D}$ is drawn from an input space $\mathcal{X}$ which is partitioned into K feature spaces $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K$ such that $\mathcal{D}_k \subseteq \mathcal{X}_k$, where, recall, $\mathcal{D}_k$ is the local dataset of client k. Ideally, we would train a general, unconstrained predictor $\tilde{h}\colon \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{Y}$ is the label space, by solving $\min_{\tilde{\boldsymbol{\theta}}} \frac{1}{N} \sum_{n=1}^{N} \ell(\tilde{h}(\boldsymbol{x}^n; \tilde{\boldsymbol{\theta}}), y^n)$. However, VFL brings the additional constraint that this must be achieved without sharing the local datasets. That is, for all k, the local dataset $\mathcal{D}_k$ must remain at client k. As mentioned in Section 1, standard VFL methods approximate $\tilde{h}$ by $h\colon \mathcal{X} \mapsto \mathcal{Y}$, as defined in (1), allowing for collaborative training without sharing the local datsets, but they cannot handle missing features during training nor inference.

Another way to train predictors without sharing local data is to learn local predictors. In particular, each client $k \in \mathcal{K}$ can learn the parameters $\boldsymbol{\theta}_k$ of a predictor $h_k : \mathcal{X}_k \mapsto \mathcal{Y}$ to approximate $\tilde{h}$:

$$h_k(\boldsymbol{x}_k^n; \boldsymbol{\theta}_k) \coloneqq g_k(\boldsymbol{f}_k(\boldsymbol{x}_k^n; \boldsymbol{\theta}_{\boldsymbol{f}_k}); \boldsymbol{\theta}_{g_k}) \quad \text{where} \quad \boldsymbol{\theta}_k \coloneqq (\boldsymbol{\theta}_{\boldsymbol{f}_k}, \boldsymbol{\theta}_{g_k}). \tag{2}$$

The representation models $\boldsymbol{f}_k : \mathcal{X}_k \mapsto \mathcal{E}_k$, where $\mathcal{E}_k$ is the representation space of client k, are as in (1), yet the fusion models $g_k : \mathcal{E}_k \mapsto \mathbb{R}$ differ from $g : \mathcal{E}_1 \times \cdots \times \mathcal{E}_K \mapsto \mathbb{R}$ in (1) and h_k differs from h. This approach is useful in that h_k allows client k to perform inference (and be trained) independently from all the other clients, but it does not make use of the features observed by clients $j \neq k$.

More generally, to achieve robustness to missing blocks in the test data while avoiding wasting other features, we wish to be able to perform inference based on any possible subset of blocks, $\mathscr{P}(\mathcal{K}) \setminus \{\varnothing\}$, where $\mathscr{P}(\mathcal{K})$ denotes the power set of $\mathcal{K}$. Standard VFL allows us to obtain a predictor for the blocks $\mathcal{K}$ and the local approach provides us with predictors for the singletons $\{\{i\}\colon i \in \mathcal{K}\}$ in the power set. However, none of the other subsets of $\mathcal{K}$ is covered by either approach.

A naive way to achieve this would be to train a predictor for each set in $\mathscr{P}(\mathcal{K}) \setminus \{\varnothing\}$ in a decoupled manner. That is, we could train each of the following predictors $\{h_{\mathcal{J}}\colon \prod_{k \in \mathcal{J}} \mathcal{X}_k \mapsto \mathcal{Y}\}$ independently, using the collaborative training approach of standard VFL for $\mathcal{J}$:

$$\left\{h_{\mathcal{J}}(\boldsymbol{x}_{\mathcal{J}}^n; \boldsymbol{\theta}_{\mathcal{J}}) \coloneqq g_{\mathcal{J}}((\boldsymbol{f}_k(\boldsymbol{x}_k^n; \boldsymbol{\theta}_{\boldsymbol{f}_k(\mathcal{J})})\colon k \in \mathcal{J}); \boldsymbol{\theta}_{g_{\mathcal{J}}})\colon \mathcal{J} \in \mathscr{P}(\mathcal{K}) \setminus \{\varnothing\}\right\}, \tag{3}$$

where $\boldsymbol{\theta}_{\mathcal{J}} = ((\boldsymbol{\theta}_{\boldsymbol{f}_k}(\mathcal{J})\colon k \in \mathcal{J}), \boldsymbol{\theta}_{g_{\mathcal{J}}})$ and $\boldsymbol{x}_{\mathcal{J}}^n = (\boldsymbol{x}_k^n\colon k \in \mathcal{J})$. These predictors can either be at one of the clients or at the server. The set of predictors in (3) includes the local predictors $\{h_k\}$ and the standard VFL predictor h, but also all the other nonempty sets in the power set $\mathscr{P}(\mathcal{K})$.[2] This approach addresses the issue of limited flexibility and robustness in prior methods, which struggle with varying numbers of available or participating clients during inference. However, by requiring the independent training of $2^K - 1$ distinct predictors, it introduces a new challenge: the number of models would grow exponentially with the number of clients, K. Consequently, the associated memory, computation, and communication costs would also increase exponentially.

Further, if the predictors in (3) are all held by a single entity, this setup introduces a dependency of all clients on that entity. To enhance robustness against clients dropping from the federation, we would like each client to be able to ensure inference whenever it has access to its corresponding block of the sample. That is, we want each client $k \in \mathcal{K}$ to train a predictor for every nonempty subset of $\mathcal{K}$ that includes k, defined as $\mathscr{P}_k(\mathcal{K}) \coloneqq \{\mathcal{J} \in \mathscr{P}(\mathcal{K})\colon k \in \mathcal{J}\}$. This allows k to make predictions regardless of whether the information on the blocks observed by other clients is available and can be leveraged to improve performance.

In the next section, we present our method, `LASER-VFL`, which enables the efficient training of a family of predictors that can handle scenarios where features from an arbitrary set of clients are missing. We have included a table of notation and a diagram illustrating our method in Appendix B.

3 OUR METHOD

Key idea. When striving to have a predictor at each client to perform inference using any subset of blocks that includes its own, the main challenge is to circumvent the exponential complexity that can ensue from the exponential number of possible combinations of available blocks. To address

[2] The notation in (3) differs slightly from (1) and (2), which use a simpler, more specific formulation.

this, in LASER-VFL, we *share model parameters* across the predictors leveraging different subsets of blocks and train them so that the representation models allow for good performance across the different combinations of blocks. Further, we train the fusion model at each client to handle any combination of representations that includes its own. To train the predictors on an exponential number of combinations of missing blocks while avoiding an exponential computational complexity, we employ a *sampling* mechanism during training, which allows us to essentially estimate an exponential combination of objectives with a subexponential complexity.

3.1 A TRACTABLE FAMILY OF PREDICTORS SHARING MODEL PARAMETERS

In our approach, each client $k \in \mathcal{K}$ has a set of predictors $\{h_{(k,\mathcal{J})} \colon \mathcal{J} \in \mathscr{P}_k(\mathcal{K})\}$. Each predictor $h_{(k,\mathcal{J})} \colon \prod_{k\in\mathcal{J}} \mathcal{X}_k \mapsto \mathcal{Y}$ has the same target, but a different domain. Therefore, we say that each predictor performs a distinct *task*. We define the predictors of client k as follows:

$$\left\{ h_{(k,\mathcal{J})}\left(\boldsymbol{x}^n_{\mathcal{J}}; \boldsymbol{\theta}_{(k,\mathcal{J})}\right) \coloneqq g_k\left(\frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} \boldsymbol{f}_j(\boldsymbol{x}^n_j; \boldsymbol{\theta}_{\boldsymbol{f}_j}); \boldsymbol{\theta}_{g_k}\right) : \mathcal{J} \in \mathscr{P}_k(\mathcal{K}) \right\}, \tag{4}$$

where $\boldsymbol{\theta}_{(k,\mathcal{J})} = ((\boldsymbol{\theta}_{\boldsymbol{f}_j} : j \in \mathcal{J}), \boldsymbol{\theta}_{g_k})$ and $|\mathcal{J}|$ denotes the cardinality of set $\mathcal{J}$.

Model parameter sharing. Note that, although we can see (4) as $|\mathscr{P}_k(\mathcal{K})| = 2^{K-1}$ different predictors, the predictors are made up of different combinations of only K representation models and K fusion models. That is, we only require parameters $\boldsymbol{\theta} \coloneqq (\boldsymbol{\theta}_{\boldsymbol{f}_1}, \ldots, \boldsymbol{\theta}_{\boldsymbol{f}_K}, \boldsymbol{\theta}_{g_1}, \ldots, \boldsymbol{\theta}_{g_K})$. In particular, in contrast to (3), where the predictors used different representation models for each task, in (4), we share the representation models across predictors. Similarly to the representation models, we have a single fusion model per client.

It is also important to note that each fusion model takes the specific form $g_k\left((\boldsymbol{f}_i(\boldsymbol{x}^n_i; \boldsymbol{\theta}_{\boldsymbol{f}_i}) \colon i \in \mathcal{J}); \boldsymbol{\theta}_{g_k}\right) = g_k(\frac{1}{|\mathcal{J}|}\sum_{i\in\mathcal{J}} \boldsymbol{f}_i(\boldsymbol{x}^n_i; \boldsymbol{\theta}_{\boldsymbol{f}_i}); \boldsymbol{\theta}_{g_k})$. By employing a nonparameterized aggregation mechanism on the extracted representations whose output does not depend on the number of aggregated representations, the same fusion model can handle different sets of representations of different sizes. In particular, we opt for an average (rather than, say, a sum) because neural networks perform better when their inputs have similar distributions (Ioffe & Szegedy, 2015). Note that the representation model can be adjusted so that using averaging instead of concatenation as the aggregation mechanism does not reduce the flexibility of the overall model, but simply shifts it from the fusion model to the representation models.

With this approach, we have avoided an exponential memory complexity by sharing the weights across an exponential number of predictors such that we have K representation models and K fusion models. In the next subesection, we will go over the efficient training of this family of predictors.

3.2 EFFICIENT TRAINING VIA TASK-SAMPLING

The family of predictors introduced in Section 3.1 can efficiently perform inference for an exponential number of tasks. We will now discuss how to train this family of predictors while avoiding exponential computation and communication complexity during the training process. The key to this approach lies in carefully sampling tasks at each gradient step of model training.

Our optimization problem. As mentioned in Section 1, we are not only interested in dealing with missing blocks of features during inference, but also during training. In particular, we assume that the availability of each block k in sample n follows a random distribution and denote by $\mathcal{K}^n_o$ the (random) subset of blocks of features of sample n that is observed in the training data. Let the loss corresponding to fusion model k using blocks of features $\mathcal{I}$ for prediction be denoted as $\tilde{\mathcal{L}}_{k,\mathcal{I}}(\boldsymbol{\theta}; n) \coloneqq \ell\left(h_{(k,\mathcal{I})}\left(\boldsymbol{x}^n_{\mathcal{I}}; \boldsymbol{\theta}_{(k,\mathcal{I})}\right), y^n\right)$, to train our predictors (4), we consider the following optimization problem:

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{L}(\boldsymbol{\theta}) \coloneqq \mathbb{E}\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}) \coloneqq \frac{1}{N}\sum_{n=1}^{N} \tilde{\mathcal{L}}_n(\boldsymbol{\theta})\right]\right\} \quad \text{where} \quad \tilde{\mathcal{L}}_n(\boldsymbol{\theta}) \coloneqq \sum_{k\in\mathcal{K}^n_o} \sum_{\mathcal{I}\in\mathscr{P}_k(\mathcal{K}^n_o)} \frac{1}{|\mathcal{I}|} \cdot \tilde{\mathcal{L}}_{k,\mathcal{I}}(\boldsymbol{\theta}; n). \tag{5}$$

Algorithm 1: LASER-VFL Training

Input: initial point $\boldsymbol{\theta}^0$, train data $\mathcal{D}$.

for $t = 0, \ldots, T-1$ **do**
- Clients $\mathcal{K}$ sample the same mini-batch $\mathcal{B}^t$ with $\mathcal{K}_o^{\mathcal{B}^t} = \mathcal{K}_o^n, \forall n \in \mathcal{B}^t$, via a shared seed.
- **for** $k \in \mathcal{K}_o^{\mathcal{B}^t}$ **in parallel do**
  - Broadcast $\boldsymbol{f}_k(\mathcal{B}_k^t; \boldsymbol{\theta}_{\boldsymbol{f}_k}^t)$ and receive $\boldsymbol{f}_i(\mathcal{B}_i^t; \boldsymbol{\theta}_{\boldsymbol{f}_i}^t)$, for all $i \in \mathcal{K}_o^{\mathcal{B}^t} \setminus \{k\}$.
  - Sample a task-defining set of blocks $\mathcal{S}_{kj}^t \sim \mathcal{U}(\mathscr{P}_k^j(\mathcal{K}_o^{\mathcal{B}^t}))$, for all $j \in [|\mathcal{K}_o^{\mathcal{B}^t}|]$.
  - Compute $\tilde{\mathcal{L}}_{k,\mathcal{S}_k^t}(\boldsymbol{\theta}^t; \mathcal{B}^t)$, as in (6), and backpropagate over fusion model g_k.
  - Send derivative of $\tilde{\mathcal{L}}_{k,\mathcal{S}_k^t}(\boldsymbol{\theta}^t; \mathcal{B}^t)$ with respect to each $k \in \mathcal{K}_o^{\mathcal{B}^t}$ and receive theirs.
  - Sum the received gradients and backpropagate over $\boldsymbol{f}_k$.
- Update the weights $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}^t)$, where $\nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}^t)$ is as in (7).

where the expectation is over the set of available feature blocks $\{\mathcal{K}_o^n : \forall n\}$. In (5), we employ the normalization $1/|\mathcal{I}|$ to avoid having a larger weight being assigned to prediction tasks concerning a larger number of blocks, as these will show up at $|\mathcal{I}|$ different clients. Further, note that, if different blocks of features have different probabilities of being observed, we can also normalize the loss with respect to the different probabilities which can be computed during the entity alignment stage of VFL (see below), before taking the expectation in (5). We omit this here for simplicity. We assume that the data is missing completely at random (Zhou et al., 2024).

Looking at (5), we see that, while we are interested in tackling $K \times 2^{K-1}$ different tasks, which falls in the realm of multi-objective optimization and multi-task learning (Caruana, 1997), we opt for combining them into a single objective, rather than employing a specialized multi-task optimizer. This corresponds to a weighted form of unitary scalarization (Kurin et al., 2022).

Our optimization method. We learn parameters $\boldsymbol{\theta}$ using a gradient-based method summarized in Algorithm 1, which we now describe. At the start of step t, we have iterate $\boldsymbol{\theta}^t$ and choose a mini-batch $\mathcal{B}^t \subseteq [N] := \{1, \ldots, N\}$ such that the set of observed feature blocks of sample n, $\mathcal{K}_o^n$, is the same for all samples $n \in \mathcal{B}^t$. We denote $\mathcal{K}_o^{\mathcal{B}^t} = \mathcal{K}_o^n$ for all $n \in \mathcal{B}^t$. Then, each client $k \in \mathcal{K}_o^{\mathcal{B}^t}$ broadcasts its representation $\boldsymbol{f}_k(\mathcal{B}_k^t; \boldsymbol{\theta}_{\boldsymbol{f}_k}^t)$, where $\mathcal{B}_k^t := \{\boldsymbol{x}_k^n : \forall n \in \mathcal{B}^t\}$, to the other clients in $\mathcal{K}_o^{\mathcal{B}^t}$.

At this point, each client $k \in \mathcal{K}_o^{\mathcal{B}^t}$ holds the representations corresponding to the set of observed feature blocks of the current mini-batch. Now, to train for all the possible combinations of feature blocks while avoiding an exponential complexity, each client k samples $K_o^t := |\mathcal{K}_o^{\mathcal{B}^t}|$ tasks—the number of blocks observed for the samples in $\mathcal{B}^t$—from $\mathscr{P}_k\left(\mathcal{K}_o^{\mathcal{B}^t}\right)$, the subset of the powerset whose elements contain k. These sampled tasks are then used to update the model parameters according to (5) while avoiding exponential complexity (note that $K_o^t \leq K$). The losses corresponding to these tasks are weighted, to get an unbiased estimator of the gradient. This allows us to train the model in such a way that it can perform inference for any subset of available blocks of features.

More precisely, the task-defining feature blocks sampled at client k are given by $\mathcal{S}_k^t := \left\{\mathcal{S}_{k1}^t, \ldots, \mathcal{S}_{kK_o^t}^n\right\}$, where $\mathcal{S}_{kj}^t$ is a set containing j feature blocks. Each set $\mathcal{S}_{kj}^t$ is sampled from a uniform distribution over a subset of $\mathscr{P}_k\left(\mathcal{K}_o^{\mathcal{B}^t}\right)$ that contains its sets of size j. That is:

$$\mathcal{S}_{kj}^t \sim \mathcal{U}\left(\mathscr{P}_k^j\left(\mathcal{K}_o^{\mathcal{B}^t}\right)\right) \quad \text{where} \quad \mathscr{P}_k^j\left(\mathcal{K}_o^{\mathcal{B}^t}\right) := \left\{\mathcal{J} \in \mathscr{P}_k(\mathcal{K}_o^{\mathcal{B}^t}) : |\mathcal{J}| = j\right\}.$$

This task-sampling mechanism provides us with $K_o^t \leq K$ representations that correspond to the blocks in $\mathcal{S}_k^t$. This allows us to efficiently estimate the loss (linear complexity), whose exact computation would lead to an exponential complexity. Based on these sampled representations, we compute the following estimate for the loss:

$$\tilde{\mathcal{L}}_{k,\mathcal{S}_k^t}(\boldsymbol{\theta}^t; \mathcal{B}^t) := \frac{1}{|\mathcal{B}^t|} \sum_{n \in \mathcal{B}^t} \sum_{j \in [K_o^t]} a_j^n \cdot \tilde{\mathcal{L}}_{k,\mathcal{S}_{kj}^t}(\boldsymbol{\theta}^t; n) \quad \text{where} \quad a_j^n = \binom{K_o^t - 1}{j-1} / j. \tag{6}$$

Algorithm 2: LASER-VFL Inference

Input: test sample $\boldsymbol{x}'$, trained parameters $\boldsymbol{\theta}^T$.
The blocks of features $\mathcal{K}'_o$ of sample $\boldsymbol{x}'$ are observed.
for $k \in \mathcal{K}'_o$ **in parallel do**
  Broadcast $\boldsymbol{f}_k(\boldsymbol{x}'_k; \boldsymbol{\theta}^T_{\boldsymbol{f}_k})$ and receive $\boldsymbol{f}_i(\boldsymbol{x}'_i; \boldsymbol{\theta}^T_{\boldsymbol{f}_i})$, for all $i \in \mathcal{K}'_o \setminus \{k\}$.
  Client k predicts the label using $h_{(k,\mathcal{K}'_o)}\left(\boldsymbol{x}^n_{\mathcal{K}'_o}; \boldsymbol{\theta}_{(k,\mathcal{K}'_o)}\right)$.

The weighting a^n_j counteracts the probability of each task being sampled, allowing us to obtain an unbiased estimate of the gradient. Now, we use this to minimize (5) by doing the following update:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta \nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}^t), \tag{7}$$

where $\mathcal{S}^t \coloneqq \{\mathcal{S}^t_1, \ldots, \mathcal{S}^t_K\}$ and

$$\nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}^t) \coloneqq \frac{1}{|\mathcal{B}^t|} \sum_{n \in \mathcal{B}^t} \sum_{k \in \mathcal{K}^n_o} \sum_{j \in [|\mathcal{K}^n_o|]} a^n_j \cdot \nabla \tilde{\mathcal{L}}_{k, \mathcal{S}^t_{kj}}(\boldsymbol{\theta}^t; n).$$

After completing the training in Algorithm 1, we can perform inference, as described in Algorithm 2. This algorithm consists simply of sharing of representations of the observed blocks test sample $\boldsymbol{x}'$ by the clients which observe their partitions for this sample, $\mathcal{K}'_o$, following by each of these clients $k \in \mathcal{K}'_o$ using $h_{(k,\mathcal{K}'_o)}$ to predict the label.

Aligning the samples based on block availability. Entity alignment is a process the precedes training in VFL where the samples of the different local datasets are matched so that their features are correctly aligned, thus allowing for collaborative model training (Liu et al., 2024). The identification of the available blocks for each sample can be performed during this prelude to VFL training, allowing for a batch selection procedure that ensures that each batch contains samples with the same observed blocks, which we use in our method.

Extensions. It is important to highlight that our method can also be easily employed in situations where different clients hold different labels, as it naturally fits multi-task learning problems. Further, it can easily be applied to setups where any subset of $\mathcal{K}$ holds the labels, sufficing to drop fusion models from the clients which do not hold the labels or have them tackle unsupervised or self-supervised learning tasks instead.

4 Convergence guarantees

In this section, we present convergence guarantees for our method. Let us start by stating the assumptions used in our results.

Assumption 1 (L-smoothness and finite infimum)**.** *Function $\tilde{\mathcal{L}}\colon \boldsymbol{\Theta} \mapsto \mathbb{R}$ is differentiable and there exists a constant $L \in (0, \infty)$ such that:*

$$\forall\, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \boldsymbol{\Theta}\colon \quad \|\nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}_1) - \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}_2)\| \leq L \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|. \tag{A1}$$

The inequality above holds if $\mathcal{L}$ is L'-smooth. We assume and define $\mathcal{L}^\star \coloneqq \inf_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) > -\infty$.

Assumption 2 (Unbiased)**.** *The mini-batch gradient estimate is unbiased. That is, let $\tilde{\mathcal{L}}_{\mathcal{B}^t}(\boldsymbol{\theta}) \coloneqq \frac{1}{|\mathcal{B}^t|} \sum_{n \in \mathcal{B}^t} \tilde{\mathcal{L}}_n(\boldsymbol{\theta})$, we have for any mini-batch $\mathcal{B}^t \subseteq [N]$ that:*

$$\forall (\boldsymbol{\theta}, t) \in \boldsymbol{\Theta} \times \{0, 1, \ldots, T-1\}\colon \quad \mathbb{E}\left[\nabla \tilde{\mathcal{L}}_{\mathcal{B}^t}(\boldsymbol{\theta})\right] = \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}), \tag{A2}$$

where the expectation is with respect to the mini-batch $\mathcal{B}$.

Assumption 3 (Bounded variance)**.** *There exists a constant $\sigma \in (0, \infty)$ such that, for any mini-batch $\mathcal{B}^t \subseteq [N]$ of size B and any set of task-defining blocks $\mathcal{S}^t$:*

$$\forall (\boldsymbol{\theta}, t) \in \boldsymbol{\Theta} \times \{0, 1, \ldots, T-1\}\colon \quad \mathbb{E}\left\|\nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}^t) - \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 \leq \frac{\sigma^2}{B}, \tag{A3}$$

where the expectation is with respect to $\mathcal{B}^t$ and $\mathcal{S}^t$.

Using the assumptions above, we can derive the results below, where we use the following filtration:

$$\mathcal{F}^t \coloneqq \sigma(\mathcal{B}^0, \mathcal{S}^0, \mathcal{B}^1, \mathcal{S}^1, \ldots, \mathcal{B}^{t-1}, \mathcal{S}^{t-1}).$$

We also define a random variable I_k^n that is equal to one if block k of sample n is observed and zero otherwise. Further, we let $\boldsymbol{I}^n = (I_1^n, \ldots, I_K^n)$ and $\boldsymbol{I} = (I^1, \ldots, I^N)$.

We use the following lemma, showing the unbiasedness of our update vector, in our proof of Theorem 1.

Lemma 1 (Unbiased update vector)**.** *If the mini-batch gradient estimate is unbiased* (A2)*, then, for all $t \geq 0$:*

$$\mathbb{E}\left[\nabla \tilde{\mathcal{L}}_{\mathcal{B}^t \mathcal{S}^t}(\boldsymbol{\theta}) \mid \sigma(\boldsymbol{I}, \mathcal{F}^t)\right] = \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}^t), \tag{8}$$

where the expectation is with respect to mini-batch $\mathcal{B}^t$ and the sampled set of tasks $\mathcal{S}^t$.

Theorem 1 (Main result)**.** *Let $\{\boldsymbol{\theta}^t\}$ be a sequence generated by Algorithm 1, if $\mathcal{L}$ is L-smooth and has a finite infimum* (A1)*, the mini-batch estimate of the gradient is unbiased* (A2)*, and our update vector has a bounded variance* (A3)*, we then have that, for $\eta \in (0, 1/L]$:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^t)\right\|^2 \leq \frac{2\Delta}{\eta T} + \frac{\eta L \sigma^2}{B}, \tag{9}$$

where $\Delta \coloneqq \mathcal{L}(\boldsymbol{\theta}^0) - \mathcal{L}^\star$ and the expectation is with respect to $\{\mathcal{B}^t\}$, $\{\mathcal{S}^t\}$, and $\boldsymbol{I}$.

If we take the stepsize to be $\eta = \sqrt{\frac{2\Delta B}{L\sigma^2 T}}$, we get from (9) that

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^t)\right\|^2 \leq \sqrt{\frac{8\Delta L \sigma^2}{BT}},$$

thus achieving a $\mathcal{O}(1/\sqrt{T})$ convergence rate. Yet, if we assume that the batch size is sufficiently large, $B = \Omega(\sigma^2/\epsilon)$, we get from (9) that we can achieve an error level $\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^t)\right\|^2 \leq \epsilon$ in $\mathcal{O}(1/T)$ iterations.

We can further establish linear convergence under the Polyak-Łojasiewicz inequality Polyak (1963).

Assumption 4 (PL inequality)**.** *We assume that there exists a positive constant μ such that*

$$\forall \boldsymbol{\theta} \in \boldsymbol{\Theta}: \quad \left\|\nabla \mathcal{L}(\boldsymbol{\theta})\right\|^2 \geq 2\mu(\mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}^\star). \tag{A4}$$

We use the expected suboptimality $\delta^t \coloneqq \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^t) - \mathcal{L}^\star$, where the expectation is with respect to $\{\mathcal{B}^t\}$, $\{\mathcal{S}^t\}$, and $\boldsymbol{I}$, as our Lyapunov function in the following result.

Theorem 2 (Linear convergence)**.** *Let $\{\boldsymbol{\theta}^t\}$ be a sequence generated by Algorithm 1, if $\mathcal{L}$ is L-smooth and has a finite infimum* (A1)*, the mini-batch estimate of the gradient is unbiased* (A2)*, our update vector has a bounded variance* (A3)*, and the PL inequality* (A4) *holds for $\mathcal{L}$, we then have that, for $\eta \in (0, 1/L]$:*

$$\delta^T \leq (1 - \mu\eta)^T \delta^0 + \frac{\eta L \sigma^2}{2\mu B}.$$

It follows from $\mu \leq L$ that $1 - \mu\eta \in (0, 1)$. Therefore, we have linear convergence to the global optimum for the full-batch case ($\sigma = 0$) and, more generally, we have linear convergence to a $\mathcal{O}(\sigma^2)$ neighborhood around it.

We defer the proofs of Lemma 1, Theorem 1, and Theorem 2 to Appendix C.

5 Experiments

In this section, we describe the experiments and analyze the results to empirically evaluate the performance of `LASER-VFL` and compare it against baseline approaches. We first provide a brief overview of the baseline methods and tasks considered. Next, we discuss the metrics used for evaluation and present the experimental results, followed by a brief discussion.

We compare our method to the following baselines:

Table 1: Test metrics for $K = 4$ clients across different datasets and varying probabilities of missing blocks during training and inference, averaged over five seeds ($\pm$ standard deviation).

Training p_{miss}	0.0			0.1			0.5		
Inference p_{miss}	0.0	0.1	0.5	0.0	0.1	0.5	0.0	0.1	0.5
HAPT (accuracy, %)									
`Local`	82.1 ± 0.6	82.2 ± 0.7	81.9 ± 1.6	82.3 ± 0.4	82.4 ± 0.4	81.8 ± 1.0	80.7 ± 1.0	80.8 ± 1.3	80.3 ± 1.7
`Standard VFL`	92.7 ± 0.4	61.9 ± 3.8	17.8 ± 5.1	91.3 ± 1.0	61.1 ± 3.7	17.6 ± 5.0	81.7 ± 2.2	54.9 ± 3.4	16.4 ± 4.2
`VFedTrans`	82.5 ± 0.4	82.5 ± 0.4	82.3 ± 0.7	82.9 ± 0.3	82.9 ± 0.2	82.8 ± 0.7	82.2 ± 0.5	82.2 ± 0.5	82.2 ± 0.8
`Ensemble`	90.5 ± 0.9	90.0 ± 0.7	84.3 ± 1.4	90.9 ± 0.7	90.2 ± 1.0	84.4 ± 1.1	88.7 ± 1.3	88.5 ± 1.6	82.6 ± 1.7
`Combinatorial`	**92.9** ± 0.6	**92.4** ± 0.7	88.3 ± 1.9	89.0 ± 1.7	89.7 ± 1.4	87.0 ± 1.9	81.9 ± 4.8	82.6 ± 4.1	83.3 ± 1.7
`PlugVFL`	92.5 ± 0.3	92.1 ± 0.5	**90.3** ± 1.3	90.6 ± 1.1	90.2 ± 0.9	**89.0** ± 0.6	88.5 ± 0.9	88.3 ± 0.8	**87.6** ± 1.5
`LASER-VFL` (ours)	**92.9** ± 0.6	92.2 ± 0.6	87.0 ± 1.2	**93.6** ± 0.5	**93.1** ± 0.5	88.2 ± 1.2	**92.3** ± 1.1	**91.7** ± 0.9	86.0 ± 1.8
Credit (F_1-score$\times 100$)									
`Local`	37.7 ± 3.1	37.6 ± 3.1	37.5 ± 3.2	37.6 ± 3.0	37.6 ± 2.9	37.5 ± 3.2	36.2 ± 3.5	36.2 ± 3.6	36.0 ± 3.6
`Standard VFL`	45.7 ± 2.6	38.8 ± 2.5	31.0 ± 0.7	42.4 ± 1.2	38.3 ± 0.7	31.0 ± 0.7	32.4 ± 2.4	32.5 ± 1.3	30.3 ± 0.5
`VFedTrans`	37.7 ± 1.5	37.6 ± 1.4	37.2 ± 1.6	39.5 ± 0.8	39.4 ± 0.8	39.2 ± 0.9	35.7 ± 0.3	35.6 ± 0.3	35.6 ± 0.4
`Ensemble`	42.1 ± 1.0	42.1 ± 1.0	40.1 ± 0.8	41.4 ± 1.3	40.8 ± 0.9	39.2 ± 1.1	**42.4** ± 2.1	**41.8** ± 1.7	40.1 ± 1.1
`Combinatorial`	42.8 ± 2.9	42.7 ± 2.5	41.7 ± 1.5	**44.4** ± 1.0	41.8 ± 2.2	**41.7** ± 1.5	32.8 ± 3.6	36.3 ± 0.6	37.6 ± 2.1
`PlugVFL`	44.4 ± 3.7	41.3 ± 4.0	32.7 ± 4.6	40.8 ± 2.0	39.8 ± 1.6	37.9 ± 1.7	38.6 ± 2.1	37.8 ± 0.9	35.4 ± 3.4
`LASER-VFL` (ours)	**46.5** ± 2.8	**45.0** ± 2.5	**43.7** ± 1.2	43.1 ± 4.2	**41.9** ± 4.0	41.3 ± 1.9	41.5 ± 4.2	40.9 ± 4.0	**41.4** ± 1.7
MIMIC-IV (F_1-score$\times 100$)									
`Local`	74.9 ± 0.5	74.9 ± 0.5	74.8 ± 0.5	75.0 ± 0.3	75.0 ± 0.2	75.0 ± 0.3	73.0 ± 0.5	73.0 ± 0.5	73.0 ± 0.5
`Standard VFL`	91.6 ± 0.2	77.0 ± 1.3	52.5 ± 2.2	90.9 ± 0.3	76.8 ± 1.3	52.5 ± 2.2	81.1 ± 0.4	70.2 ± 0.8	51.8 ± 1.8
`Ensemble`	84.2 ± 0.3	83.9 ± 0.3	77.9 ± 1.6	84.1 ± 0.4	83.7 ± 0.5	78.1 ± 1.1	82.1 ± 0.3	81.8 ± 0.5	76.4 ± 1.1
`Combinatorial`	91.3 ± 0.3	87.2 ± 1.7	83.6 ± 0.4	91.0 ± 0.3	86.7 ± 1.4	83.4 ± 0.4	80.5 ± 1.2	80.6 ± 1.7	78.9 ± 0.5
`PlugVFL`	91.2 ± 0.5	89.0 ± 0.8	79.6 ± 2.8	90.6 ± 0.2	88.8 ± 0.5	79.8 ± 2.6	86.5 ± 0.7	85.3 ± 0.9	77.9 ± 2.8
`LASER-VFL` (ours)	**92.0** ± 0.2	**91.2** ± 0.3	**85.5** ± 1.0	**91.1** ± 0.5	**90.2** ± 0.3	**84.7** ± 0.9	**87.7** ± 0.2	**86.9** ± 0.2	**82.0** ± 1.0
CIFAR-10 (accuracy, %)									
`Local`	72.4 ± 0.1	72.5 ± 0.2	72.5 ± 0.9	71.8 ± 0.2	72.0 ± 0.3	72.0 ± 0.7	67.6 ± 0.3	67.7 ± 0.4	68.0 ± 1.1
`Standard VFL`	87.8 ± 0.2	59.5 ± 12.0	11.6 ± 3.5	85.6 ± 0.8	58.0 ± 11.2	11.5 ± 3.4	52.3 ± 10.2	37.0 ± 9.0	10.9 ± 2.1
`Ensemble`	83.8 ± 0.2	82.5 ± 0.6	74.9 ± 0.5	83.2 ± 0.3	82.0 ± 0.8	74.3 ± 0.7	79.5 ± 0.5	78.1 ± 0.5	70.3 ± 0.3
`Combinatorial`	88.0 ± 0.3	87.2 ± 0.5	81.2 ± 1.0	85.6 ± 0.7	85.3 ± 0.6	80.1 ± 1.0	51.9 ± 10.1	56.3 ± 7.7	65.7 ± 3.7
`PlugVFL`	88.5 ± 0.2	87.6 ± 0.4	**81.7** ± 1.8	87.3 ± 0.4	86.6 ± 0.5	81.0 ± 1.6	78.2 ± 1.9	77.7 ± 1.9	73.9 ± 2.1
`LASER-VFL` (ours)	**90.3** ± 0.2	**89.3** ± 0.3	81.6 ± 1.2	**89.7** ± 0.2	**88.6** ± 0.3	**81.1** ± 1.1	**85.5** ± 0.3	**84.6** ± 0.4	**77.0** ± 1.1
CIFAR-100 (accuracy, %)									
`Local`	46.4 ± 0.2	46.5 ± 0.1	46.5 ± 0.1	45.5 ± 0.3	45.4 ± 0.4	45.3 ± 0.8	37.4 ± 0.6	37.7 ± 0.5	37.7 ± 0.9
`Standard VFL`	62.1 ± 0.3	40.0 ± 9.3	2.3 ± 2.8	55.6 ± 1.8	35.6 ± 7.6	2.3 ± 2.7	15.6 ± 6.5	10.2 ± 4.3	1.4 ± 0.9
`Ensemble`	58.5 ± 0.5	56.3 ± 0.9	48.6 ± 0.3	57.5 ± 0.6	55.3 ± 0.7	47.4 ± 0.7	48.2 ± 0.9	46.3 ± 0.7	40.1 ± 0.7
`Combinatorial`	62.1 ± 0.4	61.6 ± 0.4	56.4 ± 0.9	55.7 ± 1.9	56.3 ± 1.9	53.4 ± 1.0	13.0 ± 4.5	16.4 ± 4.0	27.8 ± 3.6
`PlugVFL`	64.9 ± 0.4	63.5 ± 0.6	53.5 ± 2.2	62.8 ± 0.7	61.5 ± 0.7	53.3 ± 1.4	44.7 ± 2.8	44.4 ± 2.6	41.2 ± 3.1
`LASER-VFL` (ours)	**69.7** ± 0.4	**68.2** ± 0.8	**58.4** ± 1.5	**68.3** ± 0.2	**66.8** ± 0.5	**57.0** ± 1.5	**58.5** ± 0.9	**57.3** ± 0.7	**48.7** ± 0.8

- `Local`: as in (2); the local approach leverages its block for training and inference whenever it is observed, but ignores the remaining blocks.
- `Standard VFL` (Liu et al., 2022): as in (1); the standard VFL model can only be trained on and used for inference for fully-observed samples. When the model is unavailable for prediction, a random prediction is made.
- `VFedTrans` (Huang et al., 2023): VFedTrans introduces a multi-stage VFL training pipeline to collaboratively train local predictors. (We give more details about VFedTrans below.)
- `Ensemble`: in the ensemble approach, we train local predictors as in `Local`. However, instead of performing decoupled inference, the clients share their predictions and select a joint prediction by majority vote. Ties are broken at random.
- `Combinatorial`: as in (3); during training, each batch is used by all the models corresponding to subsets of the observed blocks. During inference, we use the predictor corresponding to the set of observed blocks. We go over the scalability problems of this approach below.
- `PlugVFL` Sun et al. (2024): we extend `PlugVFL` to deal with missing features during training by replacing the missing representations with zeros.

Our experiments focus on the following tasks:

- **HAPT** (Reyes-Ortiz et al., 2016)**:** a human activity recognition dataset.
- **Credit** (Yeh & Lien, 2009)**:** a dataset for predicting credit card payment defaults.
- **MIMIC-IV** (Johnson et al., 2020)**:** a time series dataset concerning healthcare data. We focus on the task of predicting in-hospital mortality from ICU data corresponding to patients admitted with chronic kidney disease. We resort to the data processing pipeline of Gupta et al. (2022).
- **CIFAR-10 & CIFAR-100** (Krizhevsky et al., 2009)**:** are two popular image datasets, the former containing 10 classes and the latter containing 100.

For all datasets, we split the samples into $K = 4$ feature blocks. For example, for the CIFAR datasets, each image is partitioned into four quadrants, each assigned to a different client. Given our interest in ensuring that all clients perform well at inference time, we average the metrics across clients. In Appendix D, we provide detailed descriptions of how we compute accuracy and F_1-score.

We do not run VFedTrans for MIMIC-IV, CIFAR-10, and CIFAR-100 due to its greater complexity. In particular, the training pipeline of VFedTrans includes three stages: federated representation learning, a local representation distillation, and training a local model. The first two stages are performed per pair of clients and are thus repeated $K-1$ times. This means that experiments with VFedTrans take significantly longer to run (and to tune the hyperparameters of each stage of the training pipeline), as seen in Appendix D. Thus, given how VFedTrans performs similarly to Local in the first two experiments, we limit our experiments with VFedTrans to these simpler tasks.

Performance for different missing data probabilities. In Table 1, we see that the methods with local predictors (Local and VFedTrans) are robust to missing blocks during both training and inference, yet they fall behind when all the blocks are observed. In contrast, Standard VFL performs well when all the blocks are observed, yet its performance degrades very quickly in the presence of missing blocks. The Ensemble method improves upon Standard VFL significantly in the presence of missing blocks, as it leverages the robustness its local predictors. Yet, when all the blocks are observed, although Ensemble improves over the local predictors significantly, it still cannot match the performance of Standard VFL. We see that the Combinatorial approach outperforms all baselines when there is little to no missing training data ($p_{\text{miss}} \in \{0.0, 0.1\}$). Yet, for a larger amount of missing training data ($p_{\text{miss}} = 0.5$), its performance degrades significantly. This is because each mini-batch is only used to train predictors that use observed feature blocks, harming the generalization of predictors that use a larger subset of the features. This leads to an interesting phenomenon: when $p_{\text{miss}} = 0.5$ for training data, Combinatorial performs better for higher probabilities of missing testing data. This is because missing test data dictates the use of predictors trained on smaller, more frequently observed feature subsets. PlugVFL outperforms most baselines, though not consistently. LASER-VFL consistently outperforms the baselines across the different probabilities of missing blocks during training and inference. (LASER-VFL is outperformed by more than the standard deviation in only once out of 45 settings.) In particular, even when the samples are fully-observed, LASER-VFL outperforms Standard VFL and Combinatorial. We believe this is due to the regularization effect of the task-sampling mechanism in our method, which effectively behaves as a form of dropout.

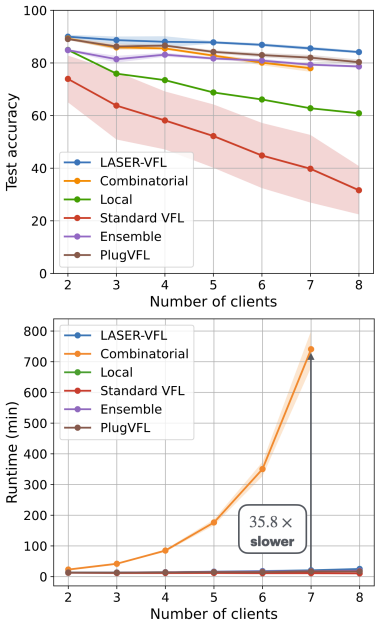


Figure 2: Performance and runtime across different numbers of clients (CIFAR-10, $p_{\text{miss}} = 0.1$ for training and inference). We did not run Combinatorial for 8 clients due to resource constraints.

Scalability. In Figure 2, we study the scalability of the different methods. Namely, we see that LASER-VFL performs the best across the different numbers of clients. After that, Combinatorial and Ensemble perform the best, with Combinatorial doing better for fewer clients, as each exact combination of feature blocks is observed more times, while Ensemble does better for a larger number of clients since, like LASER-VFL, it trains each representation model for *any* batch for which its feature block is observed (regardless of the availability of the remaining blocks). We can also observe in Figure 2 the expected scalability issues of Combinatorial.

6 Conclusions

In this work, we introduced LASER-VFL, a novel method for efficient training and inference in vertical federated learning that is robust to missing blocks of features without wasting data. This is achieved by carefully sharing model parameters and by employing a task-sampling mechanism that allows us to efficiently estimate a loss whose computation would otherwise lead to an exponential complexity. We provide convergence guarantees for LASER-VFL and present numerical experiments demonstrating the improved performance of LASER-VFL, not only in the presence of missing features but, remarkably, even when all samples are fully observed. For future work, it would be interesting to explore applications of our method to multi-task learning, seeing how our method behaves when different clients tackle different tasks, as this fits very naturally into our setup.

REFERENCES

Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

Iker Ceballos, Vivek Sharma, Eduardo Mugica, Abhishek Singh, Alberto Roman, Praneeth Vepakomma, and Ramesh Raskar. SplitNN-driven vertical partitioning. *arXiv:2008.04137*, 2020.

Di Chai, Leye Wang, Junxue Zhang, Liu Yang, Shuowei Cai, Kai Chen, and Qiang Yang. Practical lossless federated singular vector decomposition over billion-scale data, 2022. URL https://arxiv.org/abs/2105.08925.

Siwei Feng. Vertical federated learning-based feature selection with non-overlapping sample utilization. *Expert Systems with Applications*, 208:118097, 2022.

Siwei Feng and Han Yu. Multi-participant multi-class vertical federated learning. *arXiv preprint arXiv:2001.11154*, 2020.

Siwei Feng, Boyang Li, Han Yu, Yang Liu, and Qiang Yang. Semi-supervised federated heterogeneous transfer learning. *Knowledge-Based Systems*, 252:109384, 2022.

Surojit Ganguli, Zeyu Zhou, Christopher G. Brinton, and David I. Inouye. Fault tolerant serverless vfl over dynamic device environment, 2024. URL https://arxiv.org/abs/2312.16638.

Dashan Gao, Sheng Wan, Lixin Fan, Xin Yao, and Qiang Yang. Complementary knowledge distillation for robust and privacy-preserving model serving in vertical federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19832–19839, 2024.

Mehak Gupta, Brennan Gallamoza, Nicolas Cutrona, Pranjal Dhakal, Raphael Poulain, and Rahmatollah Beheshti. An Extensive Data Processing Pipeline for MIMIC-IV. In *Proceedings of the 2nd Machine Learning for Health symposium*, volume 193 of *Proceedings of Machine Learning Research*, pp. 311–325. PMLR, 28 Nov 2022. URL https://proceedings.mlr.press/v193/gupta22a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Yuanqin He, Yan Kang, Xinyuan Zhao, Jiahuan Luo, Lixin Fan, Yuxing Han, and Qiang Yang. A hybrid self-supervised learning framework for vertical federated learning. *IEEE Transactions on Big Data*, 2024.

S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

Chung-ju Huang, Leye Wang, and Xiao Han. Vertical federated knowledge transfer via representation distillation for healthcare collaboration networks. In *Proceedings of the ACM Web Conference 2023*, pp. 4188–4199, 2023.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. URL https://arxiv.org/abs/1502.03167.

Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv. *PhysioNet. Available online at: https://physionet. org/content/mimiciv/1.0/(accessed August 23, 2021)*, pp. 49–55, 2020.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021.

Yan Kang, Yuanqin He, Jiahuan Luo, Tao Fan, Yang Liu, and Qiang Yang. Privacy-preserving federated adversarial domain adaptation over feature groups for interpretability. *IEEE Transactions on Big Data*, 2022a.

Yan Kang, Yang Liu, and Xinle Liang. Fedcvt: Semi-supervised vertical federated learning with cross-view training. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4): 1–16, 2022b.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and Pawan K Mudigonda. In defense of the unitary scalarization for deep multi-task learning. *Advances in Neural Information Processing Systems*, 35:12169–12183, 2022.

Wenjie Li, Qiaolin Xia, Hao Cheng, Kouyin Xue, and Shu-Tao Xia. Vertical semi-federated learning for efficient online advertising. In *Proceedings of the International Workshop on Trustworthy Federated Learning in Conjunction with IJCAI 2023 (FL-IJCAI'23)*, 2023a. URL https://arxiv.org/abs/2209.15635.

Wenjie Li, Qiaolin Xia, Junfeng Deng, Hao Cheng, Jiangming Liu, Kouying Xue, Yong Cheng, and Shu-Tao Xia. Vfed-ssd: Towards practical vertical federated advertising, 2023b. URL https://arxiv.org/abs/2205.15987.

Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 35(4):70–82, July 2020. ISSN 1941-1294. doi: 10.1109/mis.2020.2988525. URL http://dx.doi.org/10.1109/MIS.2020.2988525.

Yang Liu, Xinwei Zhang, Yan Kang, Liping Li, Tianjian Chen, Mingyi Hong, and Qiang Yang. FedBCD: A communication-efficient collaborative learning framework for distributed features. *IEEE Transactions on Signal Processing*, 70:4277–4290, 2022.

Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Boris T Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

Zhenghang Ren, Liu Yang, and Kai Chen. Improving availability of vertical federated learning: Relaxing inference on non-overlapping data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–20, 2022.

Jorge-L Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, 2016.

Jingwei Sun, Ziyue Xu, Dong Yang, Vishwesh Nath, Wenqi Li, Can Zhao, Daguang Xu, Yiran Chen, and Holger R Roth. Communication-efficient vertical federated learning with limited overlapping samples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5203–5212, 2023.

Jingwei Sun, Zhixu Du, Anna Dai, Saleh Baghersalimi, Alireza Amirshahi, Qilin Zheng, David Atienza, and Yiran Chen. PlugVFL: Robust and IP-protecting vertical federated learning against unexpected quitting of parties, 2024. URL https://openreview.net/forum?id=TzcuXQq0aR.

Pedro Valdeira, João Xavier, Cláudia Soares, and Yuejie Chi. Communication-efficient vertical federated learning via compressed error feedback. *arXiv preprint arXiv:2406.14420*, 2024.

Yunpeng Xiao, Xufeng Li, Tun Li, Rong Wang, Yucai Pang, and Guoyin Wang. A distributed generative adversarial network for data augmentation under vertical federated learning. *IEEE Transactions on Big Data*, 2024.

Yitao Yang, Xiucai Ye, and Tetsuya Sakurai. Multi-view federated learning with data collaboration. In *Proceedings of the 2022 14th International Conference on Machine Learning and Computing*, pp. 178–183, 2022.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2): 2473–2480, 2009.

Lei Yu, Meng Han, Yiming Li, Changting Lin, Yao Zhang, Mingyang Zhang, Yan Liu, Haiqin Weng, Yuseok Jeon, Ka-Ho Chow, et al. A survey of privacy threats and defense in vertical federated learning: From model life cycle perspective. *arXiv preprint arXiv:2402.03688*, 2024.

Youran Zhou, Sunil Aryal, and Mohamed Reda Bouadjenek. Review for handling missing data with special missing mechanism. *arXiv preprint arXiv:2404.04905*, 2024.

A A MORE DETAILED COMPARISON WITH CLOSEST RELATED WORKS

As mentioned in Section 1, a few recent works have addressed the problem of missing features during VFL training. In particular, a line of work uses all samples that are not fully observed locally to train the representation models via self-supervised learning (Feng, 2022; He et al., 2024). This is followed by collaborative training of a joint predictor for the whole federation using the samples without missing features. Another line of work utilizes a similar framework but resorting to semi-supervised learning instead of self-supervised learning in order to take advantage of nonaligned samples (Kang et al., 2022b; Yang et al., 2022; Sun et al., 2023). In both cases, these works are lacking in some areas that are improved by our method: **(1)** they require the existence of fully observed samples; **(2)** they do not allow for missing features at inference time; and **(3)** they do not leverage collaborative training when more than one, but less than the total number of clients observed their feature blocks.

Another group of works deal with the leveraging of information from the entire federation to train local predictors, allowing for inference in the presence of missing features. Feng & Yu (2020); Kang et al. (2022a) propose following standard, collaborative vertical FL with a transfer learning method allowing each client to have its own predictor. These works assume that the training samples are fully observed. Liu et al. (2020); Feng et al. (2022) follow a similar direction but they also consider the presence of missing features. Instead of utilizing transfer learning to train local predictors, Ren et al. (2022); Li et al. (2023b) use knowledge distillation under the assumption of fully observed training samples, while Li et al. (2023a); Huang et al. (2023) also employ knowledge distillation, but on top of this they further consider missing features during training. Finally, Xiao et al. (2024) recently proposed using a distributed generative adversarial network for collaborative training on nonoverlapping data, leveraging synthetic data generation. In all of these works, the output of training are local predictors which, despite being able to perform inference when other clients in the federation do not observe their blocks, are also not able to leverage other feature blocks when they *are* observed. This failure to utilize valuable information during inference leads to reduced predictive power.

Only a few, very recent works allow for inference from a varying number of feature blocks in vertical FL. `PlugVFL` (Sun et al., 2024) employs party-wise dropout during training to mitigate performance drops from missing feature blocks at inference; Gao et al. (2024) introduce a multi-stage approach with complementary knowledge distillation, enabling inference with different client subsets; and Ganguli et al. (2024) deal with the related task of handling communication failure during inference in cross-device VFL. These methods make progress towards inference in vertical FL in the presence of missing feature blocks, however, they do not consider missing features in the training data. Further, none of these methods provides convergence guarantees.

In contrast to the prior art, our `LASER-VFL` method can handle any missingness pattern in the feature blocks during both training and inference simultaneously *without wasting any data*. Further, `LASER-VFL` achieves this without requiring a complex pipeline with additional stages and hyperparameters, requiring only minor modifications to the standard VFL approach.

B LASER-VFL DIAGRAM AND TABLE OF NOTATION

Table 2: Table of Notation

Symbol	Description
K	Number of clients
N	Number of samples
$\boldsymbol{x}^n$	Sample n
$\boldsymbol{x}_k^n$	Block k of sample n
$\mathcal{D}$	Global dataset
$\mathcal{D}_k$	Local dataset
$\boldsymbol{\theta}$	Parameters of our model
$\boldsymbol{\theta}_{\mathcal{K}}$	Parameters of the standard VFL model
$\boldsymbol{\theta}_{\boldsymbol{f}_k}$	Parameters of the representation model at client k
$\boldsymbol{\theta}_{\boldsymbol{g}_k}$	Parameters of the fusion model at client k
ℓ	Loss function
$\mathcal{L}$	Objective function
h	Predictor of the standard VFL model
$h_{(k,\mathcal{J})}$	Predictor at client k using feature blocks $\mathcal{J}$
$\mathscr{P}(\mathcal{S})$	Power set of $\mathcal{S}$
$\mathscr{P}_j(\mathcal{S})$	Subsets in the power set of $\mathcal{S}$ containing j
$\mathscr{P}_j^i(\mathcal{S})$	Subsets in the power set of $\mathcal{S}$ of size i containing j
$\boldsymbol{f}_k$	Representation model at client k
g_k	Fusion model at client k
h_k	Predictor at client k
g	Standard VFL fusion model
h	Standard VFL predictor
$\mathcal{K}_o^n$	Set of observed features blocks of sample n
$\boldsymbol{x}'$	Test sample

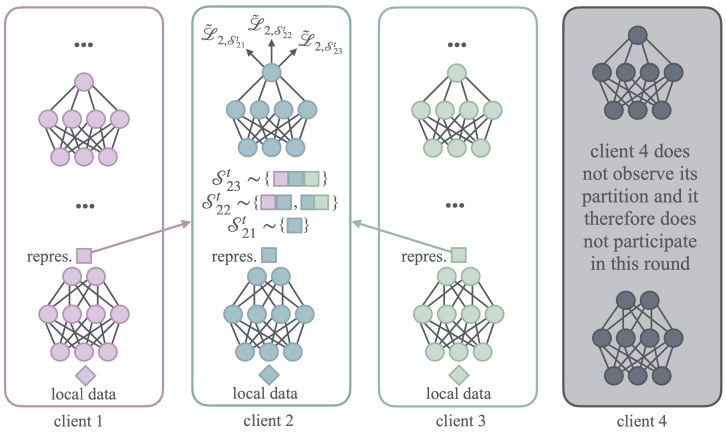


Figure 3: Diagram illustrating a forward pass of LASER-VFL.

C PROOFS

C.1 PRELIMINARIES

The following quadratic upper bound follows from the L-smoothness of $\tilde{\mathcal{L}}$ (A1):

$$\forall\, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \boldsymbol{\Theta}\colon \quad \tilde{\mathcal{L}}(\boldsymbol{\theta}_1) \leq \tilde{\mathcal{L}}(\boldsymbol{\theta}_2) + \nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}_2)^\top(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2) + \frac{L}{2}\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|^2. \tag{10}$$

Let X denote a random variable and let F be a convex function, Jensen's inequality states that

$$F(\mathbb{E}(X)) \leq \mathbb{E}(F(X)). \tag{11}$$

C.2 PROOF OF LEMMA 1

We define the following shorthand notation for the conditional expectations

$$\mathbb{E}_t\left[\cdot\right] \coloneqq \mathbb{E}_{\mathcal{B}^t\mathcal{S}^t}\left[\cdot \mid \sigma(\boldsymbol{I}, \mathcal{F}^t)\right]$$

and

$$\mathbb{E}_{t^+}\left[\cdot\right] \coloneqq \mathbb{E}_{\mathcal{B}^t\mathcal{S}^t}\left[\cdot \mid \sigma(\boldsymbol{I}, \mathcal{F}^t, \mathcal{B}^t)\right].$$

From the definition of our update vector $\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)$, in (7), we have that

$$\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] = \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} a_j^n \cdot \nabla\tilde{\mathcal{L}}_{k,\mathcal{S}_{kj}^t}(\boldsymbol{\theta}^t; n)\right]$$

and, using the law of total expectation, we arrive at

$$\begin{aligned}\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] &= \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\mathbb{E}_{t^+}\left[\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} a_j^n \cdot \nabla\tilde{\mathcal{L}}_{k,\mathcal{S}_{kj}^t}(\boldsymbol{\theta}^t; n)\right]\right] \\ &= \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} a_j^n \cdot \mathbb{E}_{t^+}\left[\nabla\tilde{\mathcal{L}}_{k,\mathcal{S}_{kj}^t}(\boldsymbol{\theta}^t; n)\right]\right].\end{aligned}$$

Now, it follows from the fact that $\mathcal{S}_{kj}^t \sim \mathcal{U}(\mathscr{P}_k^j(\mathcal{K}_o^n))$ that

$$\begin{aligned}\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] &= \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} a_j^n \cdot \sum_{\mathcal{J}\in\mathscr{P}_k^j(\mathcal{K}_o^n)} \mathbb{P}(\mathcal{S}_{kj}^n = \mathcal{J}) \cdot \nabla\tilde{\mathcal{L}}_{k,\mathcal{J}}(\boldsymbol{\theta}^t; n)\right] \\ &= \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} \frac{a_j^n}{|\mathscr{P}_k^j(\mathcal{K}_o^n)|} \cdot \sum_{\mathcal{J}\in\mathscr{P}_k^j(\mathcal{K}_o^n)} \nabla\tilde{\mathcal{L}}_{k,\mathcal{J}}(\boldsymbol{\theta}^t; n)\right] \\ &= \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} \frac{a_j^n}{\binom{|\mathcal{K}_o^n|-1}{j-1}} \cdot \sum_{\mathcal{J}\in\mathscr{P}_k^j(\mathcal{K}_o^n)} \nabla\tilde{\mathcal{L}}_{k,\mathcal{J}}(\boldsymbol{\theta}^t; n)\right].\end{aligned}$$

Now, from the definition of a_j^n, we have that

$$\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] = \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{j=1}^{|\mathcal{K}_o^n|} \frac{1}{j} \cdot \sum_{\mathcal{J}\in\mathscr{P}_k^j(\mathcal{K}_o^n)} \nabla\tilde{\mathcal{L}}_{k,\mathcal{J}}(\boldsymbol{\theta}^t; n)\right],$$

which we can rewrite as

$$\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] = \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\sum_{k\in\mathcal{K}_o^n}\sum_{\mathcal{J}\in\mathscr{P}_k(\mathcal{K}_o^n)} \frac{1}{|\mathcal{J}|} \cdot \nabla\tilde{\mathcal{L}}_{k,\mathcal{J}}(\boldsymbol{\theta}^t; n)\right].$$

Now, from the definition of $\nabla\tilde{\mathcal{L}}_n(\boldsymbol{\theta})$, in (5), we have that

$$\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] = \mathbb{E}_t\left[\frac{1}{|\mathcal{B}^t|}\sum_{n\in\mathcal{B}^t}\nabla\tilde{\mathcal{L}}_n(\boldsymbol{\theta})\right].$$

Finally, from (A2), we have that:

$$\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] = \nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t),$$

arriving at the result that we set out to prove.

C.3 Proof of Theorem 1

Let us show the convergence of our method. It follows from the quadratic upper bound in (10) which ensues from the L-smoothness assumption (A1), that

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1}) - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \leq \nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)^\top(\boldsymbol{\theta}^{t+1}-\boldsymbol{\theta}^t) + \frac{L}{2}\left\|\boldsymbol{\theta}^{t+1}-\boldsymbol{\theta}^t\right\|^2.$$

Using the fact that $\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \eta\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)$, we get that:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1}) - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \leq -\eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)^\top\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t) + \frac{\eta^2 L}{2}\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right\|^2.$$

Now, we take the conditional expectation $\mathbb{E}_t[\cdot] = \mathbb{E}_{\mathcal{B}^t\mathcal{S}^t}[\cdot \mid \sigma(\boldsymbol{I}, \mathcal{F}^t)]$, defined in Appendix C.2, arriving that:

$$\mathbb{E}_t\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1})\right] - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \leq -\eta\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)^\top\mathbb{E}_t\left[\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right] + \frac{\eta^2 L}{2}\mathbb{E}_t\left[\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right\|^2\right].$$

We can now use the result from Lemma 1 to arrive at the following inequality:

$$\mathbb{E}_t\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1})\right] - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \leq -\eta\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\eta^2 L}{2}\mathbb{E}_t\left[\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right\|^2\right]. \tag{12}$$

Now, since it follows from Lemma 1 that

$$\mathbb{E}_t\left[\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t) - \nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2\right] = \mathbb{E}_t\left[\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right\|^2\right] - \left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2,$$

Thus, we have from the bounded variance assumption (A3) that

$$\mathbb{E}_t\left[\left\|\nabla\tilde{\mathcal{L}}_{\mathcal{B}^t\mathcal{S}^t}(\boldsymbol{\theta}^t)\right\|^2\right] \leq \left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\sigma^2}{B}.$$

Using the inequality above in (12), we arrive at

$$\begin{aligned}\mathbb{E}_t\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1})\right] - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) &\leq -\eta\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\eta^2 L}{2}\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}\\ &= -\eta\left(1-\frac{\eta L}{2}\right)\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.\end{aligned}$$

Therefore, for a stepsize $\eta \in (0, 1/L]$, we have that

$$\mathbb{E}_t\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1})\right] - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \leq -\frac{\eta}{2}\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.$$

Taking the conditional expectation $\mathbb{E}[\cdot \mid \sigma(\boldsymbol{I})]$, we get from the inequality above that have:

$$\mathbb{E}\left[\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t+1}) - \tilde{\mathcal{L}}(\boldsymbol{\theta}^t) \mid \sigma(\boldsymbol{I})\right] \leq -\frac{\eta}{2}\cdot\mathbb{E}\left[\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 \mid \sigma(\boldsymbol{I})\right] + \frac{\eta^2 L\sigma^2}{2B}.$$

Now, taking the unconditional expectation (over $\boldsymbol{I}$) and using the law of total expectation and the fact that $\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{I}}[\tilde{\mathcal{L}}(\boldsymbol{\theta})]$, we get that

$$\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^t) \leq -\frac{\eta}{2}\cdot\mathbb{E}\left[\mathbb{E}\left[\left\|\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^t)\right\|^2 \mid \sigma(\boldsymbol{I})\right]\right] + \frac{\eta^2 L\sigma^2}{2B}.$$

Therefore, we have from Jensen's inequality (11) that

$$\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t}) \leq -\frac{\eta}{2} \cdot \mathbb{E}\left\|\mathbb{E}\left[\nabla\tilde{\mathcal{L}}(\boldsymbol{\theta}^{t}) \mid \sigma(\boldsymbol{I})\right]\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.$$

Using the dominated convergence theorem, we can interchange the expectation and the gradient operators, arriving at the following descent lemma in expectation:

$$\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t}) \leq -\frac{\eta}{2} \cdot \mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}. \tag{13}$$

Taking the average of the inequality above for $t \in \{0, 1, \ldots, T-1\}$, we get that:

$$\frac{\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{T}) - \mathcal{L}(\boldsymbol{\theta}^{0})}{T} \leq -\frac{\eta}{2T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 + \frac{\eta^2 L\sigma^2}{2}.$$

Lastly, rearranging the terms and using the existence and definition of $\mathcal{L}^\star$ (A1), we have that

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 \leq \frac{2(\mathcal{L}(\boldsymbol{\theta}^{0}) - \mathcal{L}^\star)}{\eta T} + \frac{\eta L\sigma^2}{B},$$

thus arriving at the result in (9).

C.4 Proof of Theorem 2

Following the same steps as in the proof of Theorem 1, we can arrive at the same descent lemma in expectation (13),

$$\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t}) \leq -\frac{\eta}{2} \cdot \mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.$$

Rearranging the terms and subtracting the infimum on both sides, we get that

$$\mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t+1}) - \mathcal{L}^\star \leq \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t}) - \mathcal{L}^\star - \frac{\eta}{2} \cdot \mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.$$

Now, using the definition of our Lyapunov function, $\delta^t = \mathbb{E}\mathcal{L}(\boldsymbol{\theta}^{t}) - \mathcal{L}^\star$, we arrive at the following inequality:

$$\delta^{t+1} \leq \delta^{t} - \frac{\eta}{2} \cdot \mathbb{E}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^{t})\right\|^2 + \frac{\eta^2 L\sigma^2}{2B}.$$

Now, from the PL inequality (A4), we have that:

$$\delta^{t+1} \leq \delta^{t} - \mu\eta \cdot \mathbb{E}\left[\mathcal{L}(\boldsymbol{\theta}^{t}) - \mathcal{L}^\star\right] + \frac{\eta^2 L\sigma^2}{2B}.$$

Therefore, again using the definition of our Lyapunov function, we arrive at

$$\delta^{t+1} \leq (1 - \mu\eta) \cdot \delta^{t} + \frac{\eta^2 L\sigma^2}{2B}.$$

Recursing the inequality above, we get that

$$\delta^{T} \leq (1 - \mu\eta)^T \delta^{0} + \frac{\eta^2 L\sigma^2}{2B}\sum_{t=0}^{T-1}(1 - \mu\eta)^t.$$

Finally, using the sum of a geometric series, we arrive at

$$\delta^{T} \leq (1 - \mu\eta)^T \delta^{0} + \frac{\eta L\sigma^2}{2\mu B},$$

which corresponds to the result that we set out to prove.

D EXPERIMENT DETAILS

Notes on `VFedTrans`. For the `VFedTrans` (Huang et al., 2023) pipeline, we use FedSVD (Chai et al., 2022) for federated representation learning and an autoencoder for local representation learning, as these are the best performing methos in Huang et al. (2023). For the local predictor, we use a multilayer perceptron, as in Huang et al. (2023). As seen in Table 1, the performance of `VFedTrans` is nearly identical to that of `Local`, however, given that requires a multi-stage pipeline with operations for each pair of clients, its runtime is much longer. In particular, for the Credit dataset, for $p_{\text{miss}} = 0.0$ for training and inference, running `VFedTrans` takes 2843.0 ± 50.6s, while running `Local` takes only 127.8 ± 3.0s.

Notes on `PlugVFL`. We also note that the original `PlugVFL` paper (Sun et al., 2024) **1)** does not address missing training data and **2)** only considers settings with two clients. In our experiments, we extended `PlugVFL` to handle more generic scenarios. To address missing training data fairly, we replaced representations for missing feature blocks with zeros at the fusion model, similarly to the dropout mechanisms in the original paper, but now, for missing data, these representations are zeroed throughout training rather than per iteration. To generalize `PlugVFL` for more than two clients, we assigned a dropout probability to each passive party (clients not holding the fusion model).

Notes on MIMIC-IV. We use ICU data of MIMIC-IV v1.0 (Johnson et al., 2020) and follow the data processing pipeline in Gupta et al. (2022). We focus on the task of predicting in-hospital mortality for patients admitted with chronic kidney disease. We do feature selection with diagnosis data and the selected features of chart events (labs and vitals) used in the "Proof of concept experiments" Gupta et al. (2022) as input features.

Computing metrics. As mentioned in the main text of the paper, we want our metrics to capture that fact that we want all clients to perform well at inference time. Therefore we consider the average metrics across the clients. More precisely, we compute the metrics for Table 1 as follows:

- For the test accuracy, let $\hat{y}^n(k)$ denote the prediction of client k for sample n, we compute:

$$\text{accuracy} = 100 \times \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{|\mathcal{K}_o^n|} \sum_{k \in \mathcal{K}_o^n} \mathbf{1}(\hat{y}^n(k) = y^n) \right).$$

- For the F_1-score ($\times 100$), we compute:

$$F_1\text{-score} = 100 \times \frac{1}{K} \sum_{k=1}^{K} \frac{P_k \cdot R_k}{P_k + R_k},$$

  where P_k and R_k are the precision and recall of client k, with respect to its predictor and (only) the samples it observed.

Models used. For the HAPT and Credit datasets, we trained simple multilayer perceptrons; for the MIMIC-IV dataset, we trained an LSTM (Hochreiter, 1997); and, for the CIFAR-10 and CIFAR-100 experiments, we use ResNets18 (He et al., 2016) models.

Reproducibility. To allow for reproducibility, we have made our code publicly available in an anonymous GitHub repository.