LESS-VFL: Communication-Efficient Feature Selection for Vertical Federated Learning

Timothy Castiglia [1] Yi Zhou [2] Shiqiang Wang [2] Swanand Kadhe [2] Nathalie Baracaldo [2] Stacy Patterson [1]

Abstract

We propose LESS-VFL, a communication-efficient feature selection method for distributed systems with vertically partitioned data. We consider a system of a server and several parties with local datasets that share a sample ID space but have different feature sets. The parties wish to collaboratively train a model for a prediction task. As part of the training, the parties wish to remove unimportant features in the system to improve generalization, efficiency, and explainability. In LESS-VFL, after a short pre-training period, the server optimizes its part of the global model to determine the relevant outputs from party models. This information is shared with the parties to then allow local feature selection without communication. We analytically prove that LESS-VFL removes spurious features from model training. We provide extensive empirical evidence that LESS-VFL can achieve high accuracy and remove spurious features at a fraction of the communication cost of other feature selection approaches.

1. Introduction

Federated learning has recently become of interest to the research community, and has shown promise in several application areas, such as healthcare, smart transportation, and predictive energy systems (Sun et al., 2019; Kairouz et al., 2021; Zhou et al., 2021). Federated learning algorithms support distributed model training among parties without the need to directly share local private data.

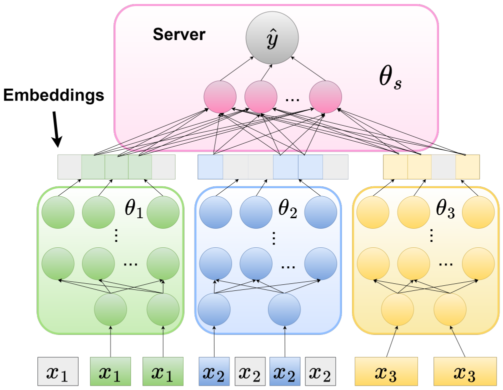


Figure 1. Example VFL model architecture. Non-significant features and embedding components (in gray) are removed after training with LESS-VFL.

Vertical Federated Learning (VFL), an important class of federated learning algorithms, has received a significant amount of attention lately (Yang et al., 2019; Cha et al., 2021; Castiglia et al., 2022). VFL works consider the case where parties store a shared sample ID space, but different private feature sets. For example, a healthcare provider, a wearable technology company, and an insurance company wish to collaboratively train a model for disease prediction. The parties have information on the same individuals (sample ID space), but each party stores different health information (feature space). In VFL, parties typically use local feature extractor models, such as deep neural networks, to produce low-dimensional *embeddings* of local feature sets (Hu et al., 2019; Ceballos et al., 2020). The server takes embeddings as input to a fusion model for predictions. We provide an example VFL model in Figure 1.

Feature selection is an important part of machine learning tasks. Often, datasets contain many spurious features that do not relate to the current prediction task. For example, health care providers may train models using electronic medical records (EMRs), which contain clinical documents, results from routine visits, and many features that may be irrelevant to disease diagnosis (Canino et al., 2016). Failing to remove spurious features can have drastic effects on generalization. In Figure 2, we compare the test accuracy of VFL training with the original dataset against training with the dataset and an additional set of Gaussian noise features. Simply adding in these spurious features causes the test accuracy of VFL to fall drastically. In addition to improving model

[1] Rensselaer Polytechnic Institute [2] IBM Research. Correspondence to: Timothy Castiglia <castigliatim@gmail.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

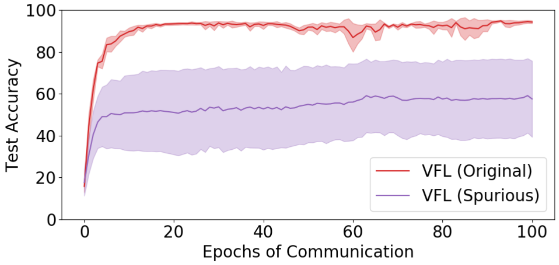


Figure 2. Test accuracy on the Activity dataset (details in Section 5). VFL (Original) denotes the test accuracy of running Algorithm 2 with the original dataset. VFL (Spurious) denotes the test accuracy on the dataset with spurious features added in. The solid line is the average of 5 runs and the shaded region represents the standard deviation.

generalization, feature selection is often used for model explainability (Clinciu & Hastie, 2019).

Most centralized feature selection algorithms cannot be directly translated to VFL because it either requires direct data sharing or is communication inefficient. Parties in VFL may be globally distributed, leading communication to be expensive in time, money, and resources. Thus it is important to design communication-efficient VFL algorithms. Although a few works propose VFL feature selection algorithms (summarized in Table 1), no work formally analyzes the feature selection problem in VFL and creates a method that provably removes spurious features. In this work, we seek to answer the following: *Can we design a communication-efficient VFL feature selection algorithm and formally verify that it removes spurious features and achieves high accuracy?*

Related Work. Feature selection algorithms tend to broadly fit into three categories: filter, wrapper, and embedded methods (Chandrashekar & Sahin, 2014). Filter methods use statistical metrics of the data to determine feature importance a priori to training a model. These methods require direct access to features to calculate the metrics and cannot be directly applied to the VFL setting without sharing raw data. Wrapper methods typically involve retraining a model several times to determine the importance of different feature subsets. This is impractical for the VFL setting where model training requires a large amount of communication between parties and the server. Embedded methods involve training a model while simultaneously determining the importance of all features. These methods may fully train a model before performing feature selection, or gradually remove unimportant features during training. Embedded methods that remove unimportant features during training seem to be a good fit for VFL, however they must be adapted to support distributed training and keep communication overhead low.

Table 1. VFL feature selection algorithms.

VFL Feature Selection Algorithm	Supports neural networks	Features selected during training	Provably removes spurious features
MMVFL (Feng & Yu, 2020)	✗	✓	✗
Fed-EINI (Chen et al., 2021)	✗	✓	✗
Hou et al. (2022)	✗	✓	✗
Zhang et al. (2022b)	✗	✓	✗
Zhang et al. (2022a)	✓	✗	✗
EVFL (Chen et al., 2022)	✓	✗	✗
VFLFS (Feng, 2022)	✓	✓	✗
FedSDG-FS (Li et al., 2023)	✓	✓	✗
LESS-VFL (ours)	✓	✓	✓

There have been a few works that propose embedded VFL feature selection methods (Feng & Yu, 2020; Chen et al., 2021; Hou et al., 2022; Zhang et al., 2022a;b; Chen et al., 2022; Feng, 2022; Li et al., 2023). However, most of these methods lack support for deep neural networks or require a fully trained model to begin feature selection (see Table 1). Critically, none of these works provide theoretical evidence that spurious features are removed with their proposed methods, only providing empirical evidence. An important open problem is how to formalize the feature selection problem in the VFL setting and provide a theoretical framework for proving that unimportant features are removed.

Contributions. In this work, we formalize the VFL feature selection problem and propose Local communication-Efficient group laSSo for Vertical Federated Learning (LESS-VFL), an embedded feature selection method for VFL that provably removes spurious features in a communication-efficient manner. Our method utilizes group lasso regularization (Zhao et al., 2015; Zhang et al., 2020; Wang et al., 2021) in a novel way that reduces the amount of communication between parties. After a short pre-training period, the server determines a set of "significant" embedding components from each party. Using this information, each party performs feature selection utilizing group lasso locally without communication. Although it has been proven that a centralized implementation of group lasso removes spurious features (Dinh & Ho, 2020), it is not obvious that our method can provide similar guarantees in VFL. We prove in our analysis that the parties asymptotically solve the feature selection problem in terms of the sample size given the set of significant embedding components. In our experiments, we compare LESS-VFL to applying group lasso regularization directly to VFL and find that our method can greatly reduce the communication cost of feature selection.

We summarize our contributions:

- We formalize the feature selection problem for VFL in Section 2.
- We propose a three-stage approach, namely LESS-VFL, along with a practical implementation in Section 3.
- We prove analytically that LESS-VFL removes spurious features and achieves high accuracy in Section 4.

- We provide empirical evidence that LESS-VFL achieves high accuracy at a low communication cost in Section 5.

2. Problem Formulation and Background

We consider a system with M parties and a server. Each party m stores d_m features for N training samples. We denote party m's dataset as $\mathbf{X}_m \in \mathbb{R}^{N\times d_m}$. We let the i-th sample in $\mathbf{X}_m$ be denoted as $\mathbf{x}_m^{(i)}$. We assume that each party's dataset is aligned, i.e., $\mathbf{x}_m^{(i)}$ and $\mathbf{x}_j^{(i)}$ for all parties $m \neq j$ are different features for data sample i. We let a combined data sample be denoted as $\mathbf{x}^{(i)} = [\mathbf{x}_1^{(i)}, \dots, \mathbf{x}_M^{(i)}]$ and let $\mathcal{X}$ be the set of all possible values for a sample $\mathbf{x}^{(i)}$. We denote the combined dataset as $\mathbf{X} = [\mathbf{X}_1, \dots, \mathbf{X}_M]$. We assume the server stores the training labels $\mathbf{y}$. Note that any party can play the role of server if it stores the labels.

Each party trains a local model $\mathbf{h}_m(\cdot)$ with parameters $\boldsymbol{\theta}_m$, and the server trains a server model $\mathbf{h}_s(\cdot)$ with parameters $\boldsymbol{\theta}_s$. The output of the party models are called *embeddings*. All party embeddings act as input to the server fusion model $\mathbf{h}_s(\cdot)$. The full VFL model $f(\cdot)$ is defined as follows:

$$f(\boldsymbol{\Theta}; \mathbf{x}^{(i)}) := \mathbf{h}_s(\boldsymbol{\theta}_s, \mathbf{h}_1(\boldsymbol{\theta}_1; \mathbf{x}_1^{(i)}), \dots, \mathbf{h}_M(\boldsymbol{\theta}_M; \mathbf{x}_M^{(i)})).$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_s^\top, \boldsymbol{\theta}_1^\top, \dots, \boldsymbol{\theta}_M^\top]^\top$.

We formalize the feature selection problem. Recall from Section 1 that not all input features may be significant to the current prediction task. We formalize this notion of significance for any given input $\mathbf{w}$ and model $u(\boldsymbol{\theta})$, which extends Definition 2.2 in Dinh & Ho (2020) to VFL systems.

Definition 2.1. For a given model u with parameters $\boldsymbol{\theta}$, let $\mathbf{w}^j$ be the j-th input to the model $u(\boldsymbol{\theta}; \mathbf{w})$, and $g^j(\mathbf{w}, s)$ be a function that replaces $\mathbf{w}^j$ with value s. The input $\mathbf{w}^j$ is *non-significant* in this model $u(\boldsymbol{\theta}; \mathbf{w})$ iff $u(\boldsymbol{\theta}; \mathbf{w}) = u(\boldsymbol{\theta}; g^j(\mathbf{w}, s))$ for all $s \in \mathbb{R}$. Otherwise, $\mathbf{w}^j$ is *significant*.

We want to emphasize that in Definition 2.1, the set of significant inputs is dependent on the model parameters $\boldsymbol{\theta}$. Throughout this paper, we apply the notion of significance in Definition 2.1 to the following two scenarios specifically:

1. When the input is the set of *training features* and the model is the classifier, i.e., $\mathbf{w} = \mathbf{x}$, $u(\cdot) = f(\cdot)$, and $\boldsymbol{\theta} = \boldsymbol{\Theta}$ in Definition 2.1;
2. When the input is the set of *embedding components* generated based on parties' local models and the model is the server model, i.e. $\mathbf{w} = [\mathbf{h}_1(\cdot); \dots; \mathbf{h}_M(\cdot)]$, $u(\cdot) = \mathbf{h}_s(\cdot)$, and $\boldsymbol{\theta} = \boldsymbol{\theta}_s$.

We define $\boldsymbol{\Theta}^\diamond = [(\boldsymbol{\theta}_s^\diamond)^\top, (\boldsymbol{\theta}_1^\diamond)^\top, \dots, (\boldsymbol{\theta}_M^\diamond)^\top]^\top$ as the *generating model* that generated the training labels: $y^{(i)} = f(\boldsymbol{\Theta}^\diamond; \mathbf{x}^{(i)}) + \epsilon^{(i)}$ where $\mathbf{x}^{(i)}$ are drawn from a distribution $\mathcal{P}_{\mathbf{X},y}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ (formal definition in Section 4). Our main goal in feature selection is to determine the set of significant features for $\boldsymbol{\Theta}^\diamond$. We let the set of significant features for a party m's generating model $\boldsymbol{\theta}_m^\diamond$ be $\mathbf{s}_m$ and the set of non-significant features be $\mathbf{z}_m$ for any data sample $\mathbf{x}_m$. We can consider the input layer weights that correspond to the significant and non-significant features:

$$\mathbf{x}_m = (\mathbf{s}_m, \mathbf{z}_m) \text{ and } \pi(\boldsymbol{\theta}_m^\diamond) = (\mathbf{U}_m, \mathbf{V}_m)$$

where $\pi(\boldsymbol{\theta})$ extracts the input weights in a model $\boldsymbol{\theta}$, and $\mathbf{U}_m$ and $\mathbf{V}_m$ are the input layer weights for the significant and non-significant features in $\boldsymbol{\theta}_m^\diamond$, respectively. Note that the separation between $\mathbf{U}_m$ and $\mathbf{V}_m$ is *not known* during training and is simply used for mathematical convenience.

The goal of embedded VFL feature selection is to find a model that simultaneously gives similar predictions to the generating model $\boldsymbol{\Theta}^\diamond$ and sets non-significant feature weights to zero:

$$\min_{\boldsymbol{\Theta}} R(\boldsymbol{\Theta}; \mathcal{P}_{\mathbf{X},\mathbf{y}}) \text{ s.t. } \mathbf{U}_m^k \neq \mathbf{0} \quad \forall m \in [M],\ \forall k \in [d_m^s]$$
$$\mathbf{V}_m^l = \mathbf{0} \quad \forall m \in [M],\ \forall l \in [d_m^z] \quad (1)$$

where $R(\cdot)$ is some generalization risk over the data distribution $\mathcal{P}_{\mathbf{X},\mathbf{y}}$ (e.g., expected squared loss, cross-entropy) and d_m^s and d_m^z are the number of significant and non-significant features at party m, respectively. Setting the input weights on non-significant features to zero removes their influence in the network, thus it essentially removes the features from the model (shown visually in Figure 1). Note $\mathbf{V}_m$ may not necessarily be zero in $\boldsymbol{\Theta}^\diamond$, as the effect of non-significant features can be lost at any layer in the model $f(\boldsymbol{\Theta}^\diamond)$.

A popular centralized method to solve (1) for neural networks is group lasso (Zhao et al., 2015; Zhang et al., 2020; Wang et al., 2021). If we apply group lasso directly to the VFL setting, then we can define the estimator as follows:

$$\bar{\boldsymbol{\Theta}} := \arg\min_{\boldsymbol{\Theta}} R_N(\boldsymbol{\Theta}; \mathbf{X}; \mathbf{y}) + \textstyle\sum_{m=1}^M \lambda_m G(\boldsymbol{\theta}_m) \quad (2)$$

where $R_N(\cdot)$ is an empirical risk that approximates $R(\cdot)$ over N training samples, and $G(\cdot)$ is $L_{2,1}$ regularization:

$$G(\boldsymbol{\theta}_m) := \textstyle\sum_{j=1}^{d_m} \|\pi(\boldsymbol{\theta}_m)^j\|_2$$

where the projection $\pi(\cdot)$ extracts the input layer weights of $\boldsymbol{\theta}_m$ and d_m is the number of input features. The regularizer $G(\cdot)$ sparsifies the input layer weights on each feature, pushing irrelevant feature weights to zero.

Why not standard group lasso? Minimizing the group lasso objective (2) using standard VFL training (Hu et al., 2019; Ceballos et al., 2020) requires the parties and server to exchange embeddings and partial derivatives every iteration of training (see Algorithm 2). Instead of communicating embeddings at every iteration, is it possible to perform feature selection locally at each client given auxiliary information from the server? In the next section, we propose a feature selection method to solve (1) that utilizes local training with minimal communication between the parties and the server.

Algorithm 1 LESS-VFL implemented using P-SGD

```
1: **Input**: pre-trained model parameters $\hat{\boldsymbol{\theta}}_s, \hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_M$
2: **for** $m \leftarrow 1, \dots, M$ in parallel **do**
3:     Send $\mathbf{h}_m(\hat{\boldsymbol{\theta}}_m; \mathbf{X}_m)$ to server
4: **end for**
5: **Initialize**: $\boldsymbol{\theta}_s^0 \leftarrow \hat{\boldsymbol{\theta}}_s$
6: **for** $t \leftarrow 0, \dots, T_1 - 1$ **do**
7:     Randomly sample $\mathcal{B} \subset [N]$
8:     $\hat{\Phi} \leftarrow \{\boldsymbol{\theta}_s^t, \mathbf{h}_1(\hat{\boldsymbol{\theta}}_1; \mathbf{X}_m^{(\mathcal{B})}), \dots, \mathbf{h}_M(\hat{\boldsymbol{\theta}}_M; \mathbf{X}_m^{(\mathcal{B})})\}$
9:     $\boldsymbol{\theta}_s^{t+1} \leftarrow \text{prox}_{\lambda_s, \eta_s^t}\left(\boldsymbol{\theta}_s^t - \eta_s^t \nabla_s R_{\mathcal{B}}(\hat{\Phi}; \mathbf{y}^{(\mathcal{B})})\right)$
10: **end for**
11: **for** $m \leftarrow 1, \dots, M$ in parallel **do**
12:     $\mathcal{K}_m = \{k \mid \|\pi(\boldsymbol{\theta}_s^{T_1})^k\|_2 > 0\}$
13:     **Initialize**: $\boldsymbol{\theta}_m^0 \leftarrow \hat{\boldsymbol{\theta}}_m$
14: **end for**
15: **for** $t \leftarrow 0, \dots, T_2 - 1$ **do**
16:     **for** $m \leftarrow 1, \dots, M$ in parallel **do**
17:         Randomly sample $\mathcal{B} \subset [N]$
18:         $\boldsymbol{\theta}_m^{t+1} = \text{prox}_{\lambda_m, \eta_m^t}\left(\boldsymbol{\theta}_m^t - \eta_m^t \nabla H_{\mathcal{B}}(\boldsymbol{\theta}_m^t; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m)\right)$
19:     **end for**
20: **end for**
21: $\bar{\boldsymbol{\Theta}} \leftarrow [\boldsymbol{\theta}_s^{T_1}, \boldsymbol{\theta}_1^{T_2}, \dots, \boldsymbol{\theta}_M^{T_2}]$
```

3. Algorithm

We present LESS-VFL, a communication-efficient approach to perform feature selection in VFL. We formalize the three stages of LESS-VFL and present a practical implementation.

Stage 1 – Pre-training. The parties and server begin by solving the following empirical risk minimization problem:

$$\hat{\boldsymbol{\Theta}} := \arg\min\nolimits_{\boldsymbol{\Theta}} R_N(\boldsymbol{\Theta}; \mathbf{X}; \mathbf{y}). \tag{3}$$

Standard VFL training, described in Algorithm 2 in Appendix A (Hu et al., 2019; Ceballos et al., 2020), is a practical method to find an approximate solution to (3).

Stage 2 – Embedding Component Selection. In this stage, the server determines the set of significant embedding components. Each party sends its current pre-trained embeddings $\mathbf{h}_m(\hat{\boldsymbol{\theta}}_m; \mathbf{x}^{(i)})$ for each sample i. These embeddings are fixed and used as input to the server model during this stage. With some abuse of notation, we let $R_N(\boldsymbol{\theta}_s, \mathbf{h}_1(\hat{\boldsymbol{\theta}}_1), \dots, \mathbf{h}_M(\hat{\boldsymbol{\theta}}_M))$ be the empirical risk of the server model using pre-trained embeddings as inputs. The server solves the following:

$$\bar{\boldsymbol{\theta}}_s := \arg\min_{\boldsymbol{\theta}_s} R_N(\boldsymbol{\theta}_s, \mathbf{h}_1(\hat{\boldsymbol{\theta}}_1), \dots, \mathbf{h}_M(\hat{\boldsymbol{\theta}}_M)) + \lambda_s G(\boldsymbol{\theta}_s) \tag{4}$$

where $G(\cdot)$ is the $L_{2,1}$ regularizer on the input layer of $\boldsymbol{\theta}_s$ and $\hat{\boldsymbol{\theta}}_m$ are party m's pre-trained parameters after pre-training. Note that the server uses the pre-trained embeddings as input, and does not require communication with the parties to calculate $R_N(\cdot)$. Solving (4) simultaneously minimizes the risk while sparsifying embedding component weights. This is illustrated in Figure 1, where non-significant embedding components (in gray) no longer provide input to the server model.

In Algorithm 1 (lines 1–10) we provide a practical method to find an approximate solution to (4). The parties generate embeddings for all data samples using the pre-trained models and send them to the server (lines 1–4). The server then starts embedding component selection (lines 5–10). The server randomly samples a mini-batch $\mathcal{B}$ of indices, then calculates the partial derivative of the risk with respect to the server model: $\nabla_s R_{\mathcal{B}}(\cdot)$. The server then employs proximal stochastic gradient descent (P-SGD). We let $\text{prox}_{\lambda,\eta}(\boldsymbol{\theta})$ with parameter λ and step size η denote the closed-form solution to the proximal operator for $L_{2,1}$ regularization:

$$\text{prox}_{\lambda,\eta}(\mathbf{P}^j) = \begin{cases} \mathbf{P}^j - \lambda\eta \frac{\mathbf{P}^j}{\|\mathbf{P}^j\|_2} & \|\mathbf{P}^j\|_2 > \lambda\eta \\ \mathbf{0} & \|\mathbf{P}^j\|_2 \leq \lambda\eta \end{cases}$$

where $\mathbf{P}^j := \pi(\boldsymbol{\theta})^j$ is the j-th group of input weights. After training, any embedding components with non-zero input weights are considered significant, and each party m is sent its list of significant embedding components indices $\mathcal{K}_m$ (lines 11–14).

Stage 3 – Feature Selection. In this stage, each party's goal is to match the values of the significant embedding components while removing non-significant features from its model. We denote the squared difference between the party's embedding value and the pre-trained embedding values over the set of significant components $\mathcal{K}_m$:

$$e(\boldsymbol{\theta}_m; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m; \mathbf{x}_m^{(i)}) := \sum\nolimits_{k \in \mathcal{K}_m} (\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m^{(i)})^k - \mathbf{h}_m(\hat{\boldsymbol{\theta}}_m; \mathbf{x}_m^{(i)})^k)^2$$

where $\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m^{(i)})^k$ is the k-th embedding component. Each party minimizes $e(\cdot)$ for each data sample while sparsifying its input layer weights:

$$\bar{\boldsymbol{\theta}}_m := \arg\min_{\boldsymbol{\theta}_m} H_N(\boldsymbol{\theta}_m; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m; \mathbf{X}_m) + \lambda_m G(\boldsymbol{\theta}_m) \tag{5}$$

where,

$$H_N(\boldsymbol{\theta}_m; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m; \mathbf{X}_m) := \frac{1}{N} \sum\nolimits_{i=1}^{N} e(\boldsymbol{\theta}_m; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m; \mathbf{x}_m^{(i)}).$$

A practical method to find an approximate solution to (5) can be seen in Algorithm 1 (lines 15–20). Each party updates its model using the mini-batch gradient $\nabla H_{\mathcal{B}}(\cdot)$ and applying $\text{prox}_{\lambda_m, \eta_m^t}(\boldsymbol{\theta})$ with regularization parameter λ_m.

After minimizing (5), any input feature weights set to zero are considered non-significant and removed from the model.

This is illustrated in Figure 1, where input weights from non-significant features (in gray) are removed. Once feature selection is complete, one can further refine the network with the remaining features using Algorithm 2 if desired.

Algorithm Cost. Stage 1 of LESS-VFL is the same as standard VFL, and thus has the same communication cost per iteration. Stages 2 and 3 only require one round of communication where parties send current embeddings to the server. The number of iterations T_1 and T_2 in Algorithm 1 controls the computation cost at the server and parties respectively, which one can tune.

Privacy. LESS-VFL uses information already shared during VFL training to perform feature selection. Thus it provides the same privacy guarantees as standard VFL. Although no raw data is shared between parties, VFL may be vulnerable to reconstruction attacks and label leakage. There have been techniques applied on top of VFL to protect against such attacks (Qiu et al., 2022; Zou et al., 2022), and these can be similarly applied to LESS-VFL. Our analysis in Section 4 still holds when applying these methods.

Theory vs. Practice. We note that Algorithms 1 and 2 provide approximate solutions to each stage's optimization problem when running a fixed number of iterations. By using P-SGD, input weights are set to zero and features are selected without the need for convergence, even if it is not the optimal set. Our analysis of LESS-VFL in Section 4 considers the ideal case where (3), (4), and (5) are solved exactly. However, we find in our experiments in Section 5 that LESS-VFL can remove spurious features and achieve high accuracy with only a few communication rounds for pre-training and an approximate solution from Algorithm 1.

Hyper-parameter tuning. Determining the best hyper-parameters for LESS-VFL (e.g. regularization parameters, number of pre-training epochs) can be done in an efficient manner. The parties and server can produce several pre-trained models for Stage 1 with different numbers of iterations. In Stage 2, the server can then explore the space of server model regularization parameters λ_s without communication. For Stage 3, the server can share the resulting sets of significant embedding component indices with the parties, and each party then can tune its local regularization parameter λ_m without communication. For choosing the numbers of iterations T_1 in Stage 2, the server can minimize its optimization problem locally until the training loss plateaus. Similarly, for choosing T_2 in Stage 3, each party can minimize its local feature selection problem until its proxy training loss plateaus.

4. Theoretical Analysis

We analyze LESS-VFL and prove under which conditions the algorithm minimizes risk and removes spurious features. We assume each party m's network is structured as follows:

- input layer: $\boldsymbol{\ell}_m^1(\mathbf{x}_m) = \mathbf{P}_m \cdot \mathbf{x}_m + \mathbf{p}_m$
- hidden layers:
  $\boldsymbol{\ell}_m^j(\mathbf{x}_m) = \zeta_m^j(\mathbf{S}_m^j, \boldsymbol{\ell}_m^{j-1}(\mathbf{x}_m), \ldots, \boldsymbol{\ell}_m^1(\mathbf{x}_m))$
- output layer: $\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m) = \mathbf{Q}_m \cdot \boldsymbol{\ell}_m^{L-1}(\mathbf{x}_m) + \mathbf{q}_m$

where d_m^j are the number of neurons in the j-th hidden layer for party m, $\mathbf{P}_m \in \mathbb{R}^{d_m^1 \times d_m}$, $\mathbf{Q}_m \in \mathbb{R}^{d_m^L \times d_m^{L-1}}$, $\mathbf{p}_m \in \mathbb{R}^{d_m^1}$, $\mathbf{q}_m \in \mathbb{R}^{d_m^L}$, and $\zeta_m^j(\cdot)$ are functions of previous layers with parameters $\mathbf{S}_m^j$. We define the server network structure the same, denoted with subscript s. This structure generalizes to several types of neural networks, including feed-forward networks, convolutional networks, and many residual networks (Dinh & Ho, 2020).

We make the following assumptions, standard in model-based feature selection (Huang et al., 2010; Wu & Liu, 2009; Dinh & Ho, 2020):

Assumption 4.1. Training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$ are sampled i.i.d. from distribution $\mathcal{P}_{\mathbf{X},\mathbf{y}}$ such that the input density $p_{\mathbf{X}}$ is positive and continuous on its open domain $\mathcal{X}$ and $y^{(i)} = f(\mathbf{\Theta}^\diamond; \mathbf{x}^{(i)}) + \epsilon^{(i)}$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Assumption 4.1 states that there is a generating model $f(\mathbf{\Theta}^\diamond)$ that generates the training labels with some Gaussian noise. This ensures that feature selection is possible since the learned model $f(\bar{\mathbf{\Theta}})$ matches the structure of the generating model. The assumption on the input density $p_{\mathbf{X}}$ ensures that there are no perfect correlations between input features. Note that since $\mathcal{X}$ is an open domain, we assume that all underlying features are continuous for this analysis.

Assumption 4.2. The hidden layer functions $\zeta_m^j(\cdot)$ in all models are analytic. The empirical risk is mean squared error: $R_N(\mathbf{\Theta}; \mathbf{X}; \mathbf{y}) := \frac{1}{N}\sum_{i=1}^N (f(\mathbf{\Theta}; \mathbf{x}^{(i)}) - y^{(i)})^2$ and the generalization risk function is expected squared error: $R(\mathbf{\Theta}; \mathcal{P}_{\mathbf{X},\mathbf{y}}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}_{\mathbf{X},\mathbf{y}}}[(f(\mathbf{\Theta}; \mathbf{x}) - y)^2]$.

Assumption 4.2 ensures that the risk function is analytic, which allows us to reason about the distance between the learned model and the generating model. Note that under the definition of the generating model in Assumption 4.1, $\mathbf{\Theta}^\diamond$ minimizes the expected squared error $R(\cdot)$.

Next, we formalize our goal to find parameters $\bar{\mathbf{\Theta}}$ that solves (1), i.e. performs the same as the generating model $\mathbf{\Theta}^\diamond$ while removing non-significant features. We define the set $\mathcal{T}^*$ as the parameters that achieve the same risk as the generating model:

$$\mathcal{T}^* := \{\mathbf{\Theta} : R(\mathbf{\Theta}) = R(\mathbf{\Theta}^\diamond)\}.$$

Recall from Section 2 that for $\mathbf{\Theta}^\diamond$, it is not necessarily the case that the input weights on non-significant features are zero. The same holds for any model in $\mathcal{T}^*$. Thus, we define a subset of parameters in $\mathcal{T}^*$ that also have weights on non-significant features set to zero:

$$\mathcal{T}^*_\phi := \{\mathbf{\Theta} : \mathbf{\Theta} \in \mathcal{T}^* \text{ and } \mathbf{V}_m = \mathbf{0}\},$$

where $\mathbf{V}_m$ are the input weights on non-significant features in the generating model $\mathbf{\Theta}^\diamond$ (definition in Section 2). We define the distance of a vector $\boldsymbol{\theta}$ from a set of vectors $\mathcal{S}$ as:

$$d(\boldsymbol{\theta}, \mathcal{S}) = \inf_{\boldsymbol{\theta}' \in \mathcal{S}} \| \boldsymbol{\theta} - \boldsymbol{\theta}' \|_2.$$

The feature selection problem (1) is solved if $d(\bar{\mathbf{\Theta}}, \mathcal{T}^*_\phi) \to 0$ for our learned parameters $\bar{\mathbf{\Theta}}$. Dinh & Ho (2020) proved this can be achieved using group lasso in centralized machine learning problems. Our goal is to show that our three-stage method can achieve the same in VFL settings.

4.1. Main Result

We present our main result below.

Theorem 4.3. *Let $\tilde{\mathbf{\Theta}} = \arg\min_{\mathbf{\Theta} \in \mathcal{T}^*} \|\mathbf{\Theta} - \hat{\mathbf{\Theta}}\|_2$. Let $\mathcal{K}_m$ and $\mathcal{Z}_m$ be the set of party m's significant and non-significant embedding components in the model $\tilde{\mathbf{\Theta}}$, respectively. Let $\bar{\mathbf{\Theta}}$ be the model after solving* (3)*,* (4)*, and* (5) *in succession. Under Assumptions 4.1 and 4.2, we conclude:*

(i) After solving (4) *and obtaining $\bar{\boldsymbol{\theta}}_s$, for all m, $\pi(\bar{\boldsymbol{\theta}}_s)^k \neq \mathbf{0}$ for all $k \in \mathcal{K}_m$ and $\pi(\bar{\boldsymbol{\theta}}_s)^l \to \mathbf{0}$ for all $l \in \mathcal{Z}_m$, where $\pi(\bar{\boldsymbol{\theta}}_s)$ are the embedding layer weights.*

(ii) When the server finds $\mathcal{K}_m$ for all m, then for any $\delta > 0$ and $\nu > 1$, if $\lambda_s \sim N^{-1/4}$ and $\lambda_m \sim N^{-1/4}$ for all m,

$$d(\bar{\mathbf{\Theta}}, \mathcal{T}^*_\phi) = O\left(\sqrt{M}\left(\frac{\log N}{N}\right)^{\frac{1}{4(\nu-1)}}\right) \tag{6}$$

with probability $1 - \delta$.

Theorem 4.3 (i) states that the server finds a set of embedding components that are significant in $\tilde{\mathbf{\Theta}}$, the closest model to the pre-trained model $\hat{\mathbf{\Theta}}$ that matches the risk of the generating model. This result ensures that the list of embedding components given by the server to the parties can serve as an accurate proxy for the loss function.

Theorem 4.3 (ii) states if we run each LESS-VFL stage to convergence, then we approach a model that minimizes the risk and removes non-significant features at a polynomial rate in terms of the number of training samples N. Since parties cannot calculate the risk $R_N(\cdot)$ locally, each uses $H_N(\cdot)$, the distance between the produced embeddings and the significant components of the pre-trained embeddings, as a proxy. It is not immediately evident that feature selection can be performed at each party without access to the server model to calculate the risk. Theorem 4.3 states that regardless of the depth or complexity of the server model, given pre-trained embeddings from solving (3) and the set of significant embedding components $\mathcal{K}_m$ found by solving (4), each party can successfully remove its non-significant features by solving (5). This emphasizes that all stages of LESS-VFL are necessary.

The bound in (6) is similar to that of centralized group lasso (Dinh & Ho, 2020), with the addition of sub-linear error growth depending on the number of parties M. It is common for M to be small in many VFL applications (Kairouz et al., 2021), thus this term has a minor effect on the bound.

4.2. Proof Sketch

For the sake of brevity, we present this proof sketch for the case where $M = 1$ (one party and server), using subscript m to denote the party. We provide the complete proof of Theorem 4.3 for $M > 1$ in the appendix. The proof for $M > 1$ is similar to that of $M = 1$ since the server-side group lasso treats embeddings as input and is agnostic to the number of parties, and party-side group lasso runs in parallel using only its own significant embedding components as a proxy for the loss function. The key challenge in the extension to $M > 1$ comes in the relationship between significant embedding components and significant party features (see Lemma B.9 in Appendix B.5).

We start by providing some definitions and additional notation. We define $H(\cdot)$ as the expected squared difference between two embeddings over the full data distribution $\mathcal{P}_{\mathbf{X},\mathbf{y}}$:

$$H(\boldsymbol{\theta}_m; \boldsymbol{\theta}'_m; \mathcal{K}_m) := \mathbb{E}_{(\mathbf{x},y) \sim \mathcal{P}_{\mathbf{X},\mathbf{y}}}\left[e(\boldsymbol{\theta}_m; \boldsymbol{\theta}'_m; \mathcal{K}_m; \mathbf{x}_m)\right]$$

where $\boldsymbol{\theta}_m, \boldsymbol{\theta}'_m$ are party model parameters, $\mathcal{K}_m$ is a set of embedding components, and $e(\cdot)$ is the square difference between embeddings components in $\mathcal{K}_m$. Recall that the notion of significance as given in Definition 2.1 can be applied to any input and model. We summarize our steps to prove that $d(\mathbf{\Theta}, \mathcal{T}^*_\phi) \to 0$:

(a) Prove that minimizing $H(\cdot)$ also minimizes $R(\cdot)$ if $\mathcal{K}_m$ is the set of significant embedding components.

(b) Prove that $e(\cdot\,; \mathcal{K}_m)$ has the same significant and non-significant features as $f(\cdot)$ if $\mathcal{K}_m$ is the set of significant embedding components.

(c) Prove that (4) finds optimal server parameters and finds the set of significant embedding components.

(d) Prove that given the set of significant embedding components, (5) finds optimal party parameters and removes non-significant features.

We start by proving (a) and (b). In the following proposition, we discuss the relationship between the significance of features in the full network versus the significance of features to embedding components in the party sub-network.

Proposition 4.4. *Consider a model $\mathbf{\Theta} = [\boldsymbol{\theta}_s^\top, \boldsymbol{\theta}_m^\top]^\top$. Let $\boldsymbol{s}$ and $\boldsymbol{z}$ be the sets of significant and non-significant features for $f(\mathbf{\Theta})$, respectively. Let the set of significant embed-*

ding components for $f(\boldsymbol{\theta}_s;\boldsymbol{h}(\boldsymbol{\theta}_m;\boldsymbol{x}))$ be $\boldsymbol{s}_s$. Let $g^j(\boldsymbol{x},r)$ replace input $\boldsymbol{x}^j$ with value r. For each significant embedding component $k \in \boldsymbol{s}_s$, for all $j \in \boldsymbol{z}$, and any $r \in \mathbb{R}$, $\boldsymbol{h}(\boldsymbol{\theta}_m;\boldsymbol{x})^k = \boldsymbol{h}(\boldsymbol{\theta}_m;g^j(\boldsymbol{x},r))^k$.

Informally, Proposition 4.4 states that significant embedding component values are unchanged by non-significant features, and can *only* be changed by significant features.

Proof. Suppose that $\mathbf{h}(\boldsymbol{\theta}_m;\mathbf{x})^k \neq \mathbf{h}(\boldsymbol{\theta}_m;g^j(\mathbf{x},r))^k$ for some significant embedding component $k \in \mathbf{s}_s$, non-significant feature $j \in \mathbf{z}$, and $r \in \mathbb{R}$. By our supposition and since component k is significant, $f(\boldsymbol{\theta}_s;\mathbf{h}(\boldsymbol{\theta}_m;\mathbf{x})) \neq f(\boldsymbol{\theta}_s;\mathbf{h}(\boldsymbol{\theta}_m;g^j(\mathbf{x};r)))$ for some value $r \in \mathbb{R}$. This contradicts the fact that j is a non-significant feature. □

Utilizing Proposition 4.4, we can prove (a) and (b).

Lemma 4.5. *Let $\tilde{\boldsymbol{\Theta}} = [\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_m^\top]^\top \in \mathcal{T}^*$. Let $\mathcal{K}_m$ be the significant embedding components for $f(\tilde{\boldsymbol{\theta}}_s;\boldsymbol{h}(\tilde{\boldsymbol{\theta}}_m))$. Let $\boldsymbol{\theta}_m = \arg\min_{\boldsymbol{\theta}'_m} H(\boldsymbol{\theta}'_m;\tilde{\boldsymbol{\theta}}_m;\mathcal{K}_m)$. Let $\boldsymbol{s}$ and $\boldsymbol{z}$ be the significant and non-significant features for $f(\boldsymbol{\Theta}^\diamond)$. Let $\boldsymbol{s}_h$ and $\boldsymbol{z}_h$ be the significant and non-significant features for $e(\boldsymbol{\theta}_m;\tilde{\boldsymbol{\theta}}_m;\mathcal{K}_m)$ with parameters $\boldsymbol{\theta}_m$. Then:*

$$[\tilde{\boldsymbol{\theta}}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^*, \;\; \boldsymbol{s}_h = \boldsymbol{s}, \;\; \textit{and} \;\; \boldsymbol{z}_h = \boldsymbol{z}\,.$$

The proof of Lemma 4.5 is given in Appendix B.2. Lemma 4.5 proves that $H(\cdot)$ can serve as a proxy for $R(\cdot)$ if the pre-trained model parameters $\hat{\boldsymbol{\Theta}}$ are in the optimal set $\mathcal{T}^*$ and the set of selected embedding components $\mathcal{K}_m$ contains only the significant embedding components.

Next, we consider objective (c), and prove that the server can find the significant embedding components required for (a) and (b). We first define $\tilde{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}\in\mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2$ as the closest model in $\mathcal{T}^*$ to the pre-trained parameters $\hat{\boldsymbol{\Theta}}$. The server's goal is to find the set of significant embedding components in $\tilde{\boldsymbol{\Theta}}$, which the parties can then use to remove their non-significant features. We define the set of server model parameters in $\mathcal{T}^*$ with non-significant embedding component weights set to zero:

$$\mathcal{S}_\phi^* = \{\boldsymbol{\theta}_s : \exists\, \boldsymbol{\theta}_m \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^* \text{ and } \mathbf{V}_s = \mathbf{0}\}$$

where $\mathbf{V}_s$ are the weights on non-significant embedding components in $\tilde{\boldsymbol{\Theta}}$. If $d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}_\phi^*) \to 0$, the server finds the set of significant embedding components, completing objective (c). We bound this distance in the following theorem.

Theorem 4.6. *Given a pre-trained model $\hat{\boldsymbol{\Theta}}$ defined by* (3), *for any $\delta > 0$, there exists $N \geq N_0(\delta)$ such that:*

$$d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}_\phi^*) = O\left(\frac{\log N}{\lambda_s \sqrt{N}} + \left(\frac{\log N}{\sqrt{N}} + \lambda_s^{\nu/(\nu-1)}\right)^{1/\nu}\right)$$

with probability $1 - \delta$.

The proof of Theorem 4.6 can be found in Appendix B.3. The bound in Theorem 4.6 indicates that if $\lambda_s \sim N^{-1/4}$, then non-significant weights will approach zero at a polynomial rate in terms of the number of training samples N; if the regularization parameter is set appropriately, then the set of significant embedding components are found.

Finally, since the server finds the set of significant embedding components allowing $H(\cdot)$ to be a proxy of $R(\cdot)$, we can prove objective (d). We define the set of party models in $\mathcal{T}^*$ with non-significant feature weights set to zero:

$$\mathcal{C}_\phi^* = \{\boldsymbol{\theta}_m : \exists\, \boldsymbol{\theta}_s \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^* \text{ and } \mathbf{V}_m = \mathbf{0}\}$$

where $\mathbf{V}_m$ are the input weights on non-significant features in the generating model $\boldsymbol{\Theta}^\diamond$. If $d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}_\phi^*) \to 0$, then the party removes the non-significant features, completing objective (d). We bound this distance in the following theorem.

Theorem 4.7. *Let $[\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_m^\top]^\top = \arg\min_{\boldsymbol{\Theta}\in\mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2$ where $\hat{\boldsymbol{\Theta}}$ are pre-trained model parameters defined in* (3). *If $\mathcal{K}_m$ in* (5) *is the set of significant embedding components for $f(\tilde{\boldsymbol{\theta}}_s;\boldsymbol{h}(\tilde{\boldsymbol{\theta}}_m))$, then for any $\delta > 0$, there exists some number of samples $N \geq N_0(\delta)$ such that:*

$$d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}_\phi^*) \leq O\left(\frac{\log N}{\lambda_m \sqrt{N}} + \left(\frac{\log N}{\sqrt{N}} + \lambda_m^{\nu/(\nu-1)}\right)^{1/\nu}\right)$$

with probability $1 - \delta$.

The proof of Theorem 4.7 can be found in Appendix B.4. Similar to the server's case, this bound goes to zero at a polynomial rate if $\lambda_m \sim N^{-1/4}$. Note that for Theorem 4.7 to hold, the server must provide the party with the set of significant embeddings. Otherwise, the bound is not guaranteed. In Section 5, we explore the importance of the embedding component selection stage in practice. In Appendix B.5, we extend Theorems 4.6 and 4.7 to the case where $M > 1$, which can be combined to prove Theorem 4.3.

5. Experiments

We implement LESS-VFL by running a fixed number of iterations of Algorithm 2 (standard VFL algorithm, see Appendix A), then running Algorithm 1 to remove non-significant features, then continuing training with Algorithm 2. We evaluate LESS-VFL on several datasets.

- **MIMIC-III** (Johnson et al., 2016; Harutyunyan et al., 2019): Hospital dataset consisting of time-series medical information on anonymized patients. Used to predict in-hospital mortality. Contains 14,681 samples each with 712 features.
- **Activity** (Anguita et al., 2013): Time-series positional data on humans performing various activities. Used for multi-class classification of the current activity (walking,

sitting, running, etc.). Contains 7,352 samples each with 560 features.

- **Phishing** (Dua & Graff, 2017): Dataset that provides relevant features for determining if a website is a phishing website (use of HTTP, TinyURL, forwarding, etc.). Contains 11,055 samples each with 30 features.
- **Gina** (Guyon, 2007): Hand-written two-digit images. Used for binary classification between even and odd numbers, meaning only the first digit is necessary for classification and the rest of the features are distractions. Contains 3,468 samples each with 968 features.
- **Sylva** (Guyon, 2007): Forest cover type information. Used for binary classification (Ponderosa pine vs. everything else). Similar to Gina, half the features are distractions; each sample has two records with relevant information for the target, while the other two are randomly chosen. Contains 14,395 samples each with 216 features.

For each dataset, we add 50% more features that are Gaussian noise. These spurious features act as our non-significant features, allowing us to measure how well LESS-VFL performs feature selection. Note that not all the features in the original dataset are necessarily significant. The only features we know for sure are non-significant are the Gaussian noise features we add to each dataset. Thus, the final test accuracy is our indicator that we have correctly selected significant features in the dataset and trained a model that generalizes well.

We compare LESS-VFL with the following VFL baselines.

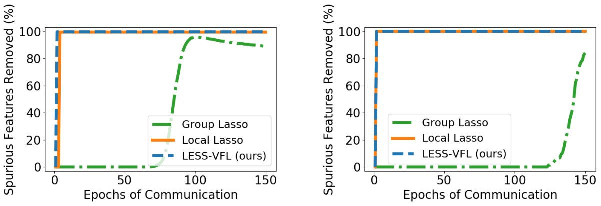
- **VFL (Original)**: VFL as described in Algorithm 2 *without* spurious features in the datasets.
- **VFL (Spurious)**: Algorithm 2 *with* spurious features in the datasets.
- **Group Lasso**: Applies group lasso directly to the VFL model by approximately solving (2) using P-SGD.
- **Local Lasso**: This algorithm is the same as LESS-VFL with stage 2, embedding component selection, removed.

We restrict our evaluations to methods that do not require a fully trained VFL model as input, which excludes Zhang et al. (2022a) and Chen et al. (2022) from our comparison. The feature selection portion of VFLFS (Feng, 2022) employs group lasso, which we include in our evaluations.

Training Details. For each dataset, we split both the original and Gaussian noise features evenly among a set of parties (three parties for Phishing, four parties otherwise). Each party's model is a 3-layer dense neural network, and the server trains a linear model that takes the concatenation of party embeddings as input. We run a grid search to determine regularization parameters for LESS-VFL, local lasso, and group lasso, and the number of pre-training epochs for LESS-VFL and local lasso. We chose parameters that achieved the highest training accuracy and removed at least 80% of spurious features. We use the ADAM optimizer with a learning rate of 0.01 when employing Algorithm 2 in VFL (Original and Spurious) and pre-training and post feature selection in local lasso and LESS-VFL. We run 150 epochs of P-SGD for embedding component selection in LESS-VFL and feature selection in LESS-VFL and local lasso, which we found to be a sufficient amount of iterations for the training loss to plateau.

Table 2. Communication cost to achieve 90% of baseline test accuracy and remove at least 80% of the spurious features. The value shown is the average of 5 runs ± the standard deviation.

Dataset	Communication Cost (MB)		
	Group Lasso	**Local Lasso**	**LESS-VFL (ours)**
MIMIC-III	57.35 ± 0.00	30.47 ± 1.79	**7.17 ± 0.00**
Activity	322.73 ± 61.32	26.56 ± 5.83	**21.17 ± 0.00**
Phishing	95.22 ± 1.89	8.10 ± 3.40	**3.99 ± 0.75**
Gina	13.55 ± 0.00	1.90 ± 0.27	**1.48 ± 0.26**
Sylva	22.49 ± 0.00	**5.62 ± 0.00**	**5.62 ± 0.00**

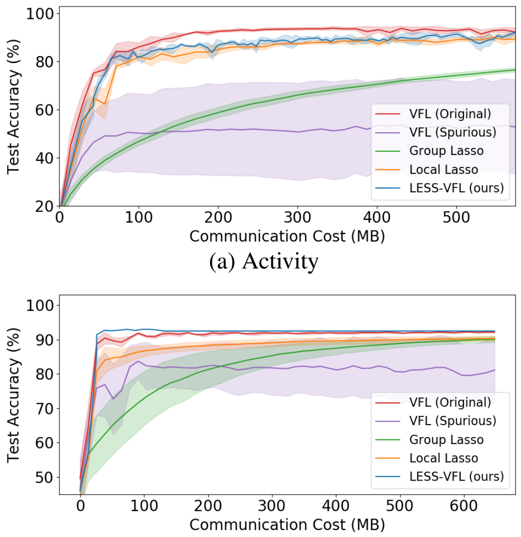


(a) Activity (b) Phishing

Figure 3. Communication rounds to remove spurious features. The values shown is the average of 5 runs. LESS-VFL and local lasso remove a similar percentage of spurious features after pre-training, though local lasso takes longer to reach high accuracy (see Table 5). Group Lasso gradually removes features while local lasso and LESS-VFL remove features with only a few rounds of communication after pre-training.

Communication cost. In Table 2, we compare the communication cost of reaching a target test accuracy while removing at least 80% of the spurious features. We choose a target accuracy of 90% of the maximum accuracy reached by VFL (Original). In all cases, LESS-VFL meets these conditions with the lowest communication cost, reducing the communication cost when compared to group lasso. In the case of the Phishing dataset, LESS-VFL has ∼20× lower communication cost than group lasso. LESS-VFL greatly reduces the cost of feature selection over group lasso by only communicating during pre-training, and once at the start of feature selection. LESS-VFL also always achieves the same or lower communication cost than local lasso. Local lasso forgoes embedding component selection, and in most datasets, this led to higher communication cost. We explore this more in our next set of experiments.

In the remaining experiments, we seek to illustrate how



(a) Activity

(b) Phishing

Figure 4. Test accuracy plotted by communication cost. VFL (Original) is trained *without* spurious features, while all other methods are trained *with* spurious features. The solid lines are the average of 5 runs and the shaded region represents the standard deviation.

LESS-VFL performs over the course of training. We focus on two representative datasets (Activity and Phishing). We provide results for all datasets in Appendix C.

Feature removal. In Figure 3, we compare the percentage of spurious features removed using group lasso, local lasso, and LESS-VFL over the communication epochs. Group lasso gradually removes features over the course of training, while local lasso and LESS-VFL remove features after a few rounds of communication for pre-training. LESS-VFL benefits greatly from using local training without communication to perform its feature selection. Local lasso removes a similar percentage of features as LESS-VFL in about the same communication epochs. However, we see in the next experiment that local lasso can require more communication to both reach high accuracy and remove spurious features.

Accuracy. In Figure 4, we plot the test accuracy against communication cost. The test accuracy of VFL (Spurious) in both datasets indicates that VFL training without removing spurious features can have a drastic effect on the generalization. In both cases, LESS-VFL achieves high accuracy faster than group lasso, and achieves similar accuracy to the baseline VFL algorithm without spurious features. In fact, in the Phishing dataset, LESS-VFL performs better than the VFL (Original) baseline. This is due to LESS-VFL removing non-significant embedding components which reduce post feature selection communication cost. Local lasso performs similarly to LESS-VFL in the Activity dataset, but local lasso requires a much higher communication cost to achieve high accuracy in the Phishing dataset. In Figure 4b, we can see local lasso has a lower model accuracy than LESS-VFL after feature selection (at ∼25 MB). This reinforces that embedding component selection can improve model accuracy by both minimizing risk to refine server model parameters, and providing parties with important information for local feature selection.

Table 3. Experimental results with heterogeneous feature partitions. Communication cost to achieve 90% of baseline test accuracy and remove at least 80% of the spurious features. The value shown is the average of 5 runs ± the standard deviation.

Dataset	**Communication Cost (MB)**		
	Group Lasso	**Local Lasso**	**LESS-VFL (ours)**
MIMIC-III	57.35 ± 0.00	**7.17 ± 0.00**	**7.17 ± 0.00**
Activity	187.39 ± 52.61	24.77 ± 6.75	**15.76 ± 3.09**
Phishing	94.57 ± 1.94	5.51 ± 0.79	**3.98 ± 0.74**
Gina	13.55 ± 0.00	1.63 ± 0.33	**1.35 ± 0.00**
Sylva	21.93 ± 1.12	**5.62 ± 0.00**	**5.62 ± 0.00**

Uneven Features. For the previous experiments, we considered a case where all parties have the same percentage of Gaussian noise features. We now consider a case where parties have an uneven distribution of Gaussian noise features: One party with 80% additional Gaussian noise features, one with 25%, one with 10%, and one with no Gaussian noise features. Table 3 shows the communication cost of group lasso, local lasso, and LESS-VFL to reach 90% of baseline VFL (Original) test accuracy while removing 80% of the total spurious features. We find that, in this heterogeneous setting, LESS-VFL still achieves high accuracy while removing spurious features, and does so with low communication cost.

6. Conclusion

In this work, we proposed LESS-VFL, a communication-efficient method for feature selection in vertical federated learning. We analytically proved that LESS-VFL removes spurious features. We experimentally showed that LESS-VFL can achieve comparable accuracy and percentage of spurious features removed at reduced communication cost. In the future, we seek to extend our analysis to non-analytic neural networks and adaptive group lasso.

Acknowledgment

This material is based upon work supported in part by the National Science Foundation under Grant CNS-1553340, and by the Rensselaer-IBM AI Research Collaboration (`http://airc.rpi.edu`), part of the IBM AI Horizons Network.

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. A public domain dataset for human activity recognition using smartphones. In *European Symp. Artif. Neural Net.*, 2013.

Canino, G., Suo, Q., Guzzi, P. H., Tradigo, G., Zhang, A., and Veltri, P. Feature selection model for diagnosis, electronic medical records and geographical data correlation. In *Proc. ACM Int. Conf. Bioinfo. Comp. Bio. Health Info.*, pp. 616–621, 2016.

Castiglia, T., Das, A., Wang, S., and Patterson, S. Compressed-VFL: Communication-efficient learning with vertically partitioned data. In *Proc. 39th Int. Conf. on Machine Learn.*, pp. 2738–2766, 2022.

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

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

Chandrashekar, G. and Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.*, 40(1):16–28, 2014.

Chen, P., Du, X., Lu, Z., Wu, J., and Hung, P. C. K. EVFL: an explainable vertical federated learning for data-oriented artificial intelligence systems. *J. Syst. Archit.*, 126:102474, 2022.

Chen, X., Zhou, S., Guan, B., Yang, K., Fao, H., Wang, H., and Wang, Y. Fed-EINI: An efficient and interpretable inference framework for decision tree ensembles in vertical federated learning. In *IEEE Int. Conf. Big Data*, pp. 1242–1248, 2021.

Clinciu, M. and Hastie, H. A survey of explainable ai terminology. In *Proc. Workshop Interactive Natural Lang. Tech. Explainable AI*, pp. 8–13, 2019.

Dinh, V. C. and Ho, L. S. T. Consistent feature selection for analytic deep neural networks. In *Adv. Neural Inf. Process. Syst.*, 2020.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Feng, S. Vertical federated learning-based feature selection with non-overlapping sample utilization. *Expert Syst. Appl.*, 208:118097, 2022.

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

Guyon, I. Agnostic learning vs. prior knowledge. IJCNN Workshop on Agnostic Learning vs. Prior Knowledge, 2007. URL `http://www.agnostic.inf.ethz.ch/datasets.php`. Accessed Apr. 27 2023.

Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1): 96, 2019.

Hou, J., Su, M., Fu, A., and Yu, Y. Verifiable privacy-preserving scheme based on vertical federated random forest. *IEEE Internet Things*, 9(22):22158–22172, 2022.

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

Huang, J., Horowitz, J. L., and Wei, F. Variable selection in nonparametric additive models. *The Annals of Statistics*, 38(4):2282–2313, 2010.

Ji, S., Kollár, J., and Shiffman, B. A global łojasiewicz inequality for algebraic varieties. *Trans. American Math. Soc.*, 329(2):813–818, 1992.

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

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

Li, A., Peng, H., Zhang, L., Huang, J., Guo, Q., Yu, H., and Liu, Y. FedSDG-FS: Efficient and secure feature selection for vertical federated learning. In *IEEE Int. Conf. Comp. Comm.*, 2023.

Qiu, P., Zhang, X., Ji, S., Pu, Y., and Wang, T. All you need is hashing: Defending against data reconstruction attack in vertical federated learning. *arXiv:2212.00325*, 2022.

Sun, C., Ippel, L., van Soest, J., Wouters, B., Malic, A., Adekunle, O., van den Berg, B., Mussmann, O., Koster, A., van der Kallen, C., van Oppen, C., Townend, D., Dekker, A., and Dumontier, M. A Privacy-Preserving infrastructure for analyzing personal health data in a vertically partitioned scenario. 264:373–377, 2019.

Wang, J., Zhang, H., Wang, J., Pu, Y., and Pal, N. R. Feature selection using a neural network with group lasso regularization and controlled redundancy. *IEEE Trans. Neural Networks Learn. Syst.*, 32(3):1110–1123, 2021.

Wu, Y. and Liu, Y. Variable selection in quantile regression. *Statistica Sinica*, pp. 801–817, 2009.

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

Zhang, H., Wang, J., Sun, Z., Zurada, J. M., and Pal, N. R. Feature selection for neural networks using group lasso regularization. *IEEE Trans. Knowl. Data Eng.*, 32(4): 659–673, 2020.

Zhang, R., Li, H., Hao, M., Chen, H., and Zhang, Y. Secure feature selection for vertical federated learning in ehealth systems. In *IEEE Int. Conf. Comm.*, pp. 1257–1262. IEEE, 2022a.

Zhang, Y., Hu, Y., Gao, X., Gong, D., Guo, Y., Gao, K., and Zhang, W. An embedded vertical-federated feature selection algorithm based on particle swarm optimisation. *CAAI Trans. Intel. Techn.*, 2022b.

Zhao, L., Hu, Q., and Wang, W. Heterogeneous feature selection with multi-modal deep neural networks and sparse group LASSO. *IEEE Trans. Multim.*, 17(11):1936–1948, 2015.

Zhou, J., Zhang, S., Lu, Q., Dai, W., Chen, M., Liu, X., Pirttikangas, S., Shi, Y., Zhang, W., and Herrera-Viedma, E. A survey on federated learning and its applications for accelerating industrial internet of things. *arXiv:2104.10501*, 2021.

Zou, T., Liu, Y., Kang, Y., Liu, W., He, Y., Yi, Z., Yang, Q., and Zhang, Y.-Q. Defending batch-level label inference and replacement attacks in vertical federated learning. *IEEE Trans. Big Data*, 2022.

A. Vertical Federated Learning Algorithm

In Algorithm 2, we present pseudocode for standard VFL training with neural networks (Hu et al., 2019; Ceballos et al., 2020).

Algorithm 2 Vertical Federated Learning

```
**Initialize:** $\boldsymbol{\theta}_m^0$ for all parties $m$ and server model $\boldsymbol{\theta}_s^0$
**for** $t \leftarrow 0, \dots, T_1 - 1$ **do**
    Randomly sample $\mathcal{B} \subset [N]$
    **for** $m \leftarrow 1, \dots, M$ in parallel **do**
        Send $\mathbf{h}_m(\boldsymbol{\theta}_m^t; \mathbf{X}_m^{(\mathcal{B})})$ to server
    **end for**
    $\Phi \leftarrow \{\boldsymbol{\theta}_s^t, \mathbf{h}_1(\boldsymbol{\theta}_1^t; \mathbf{X}_m^{(\mathcal{B})}), \dots, \mathbf{h}_M(\boldsymbol{\theta}_M^t; \mathbf{X}_m^{(\mathcal{B})})\}$
    $\boldsymbol{\theta}_s^{t+1} \leftarrow U(\boldsymbol{\theta}_s^t, \nabla_s R_{\mathcal{B}}(\Phi^t; \mathbf{y}^{\mathcal{B}^t}))$
    Server sends $\nabla_{\mathbf{h}_m(\boldsymbol{\theta}_m^t)} R_{\mathcal{B}}(\Phi^t; \mathbf{y}^{(\mathcal{B})})$ to each party
    **for** $m \leftarrow 1, \dots, M$ in parallel **do**
        $\nabla_m R_{\mathcal{B}}(\Phi^t) = \nabla_{\theta_m} \mathbf{h}_m(\boldsymbol{\theta}_m^t)^\top \nabla_{\mathbf{h}_m(\boldsymbol{\theta}_m^t)} R_{\mathcal{B}}(\Phi^t)$
        $\boldsymbol{\theta}_m^{t+1} = U(\boldsymbol{\theta}_m^t, \nabla_m R_{\mathcal{B}}(\Phi^t))$
    **end for**
**end for**
```

The parties start by agreeing upon a mini-batch samples $\mathcal{B}$, then sending their current embeddings for the given mini-batch to the server. We let $\mathbf{X}^{(\mathcal{B})}$ and $\mathbf{y}^{(\mathcal{B})}$ denote the training samples and labels in the mini-batch, respectively. The server updates its model using the mini-batch partial derivative with respect to $\boldsymbol{\theta}_s$, denoted by $\nabla_s R_{\mathcal{B}}(\cdot)$, and some optimizer update rule $U(\cdot)$ (e.g. SGD, Adam, etc.). The server then sends the partial derivatives with respect to the party's embeddings. Each party m then updates its model using its mini-batch partial derivative, denoted by $\nabla_m R_{\mathcal{B}}(\cdot)$.

B. Proof of Theorem 4.3

In this section, we start by proving Theorems 4.7 and 4.6 for the case when $M = 1$, extend the results to $M > 1$ case, and finally prove Theorem 4.3. We provide a summary of the notation used in this section in Table 4.

B.1. Additional Notation for $M = 1$

We start by providing additional notation for proving Theorems 4.7 and 4.6. We define the set of party and server model parameters that are in the optimal parameter set $\mathcal{T}^*$:

$$\mathcal{C}^* = \{\boldsymbol{\theta}_m : \exists\, \boldsymbol{\theta}_s \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^*\}$$

and

$$\mathcal{S}^* = \{\boldsymbol{\theta}_s : \exists\, \boldsymbol{\theta}_m \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^*\}.$$

We use the following lemmas proven by Dinh & Ho (2020):

Lemma B.1. *Let* $\boldsymbol{U}$ *and* $\boldsymbol{V}$ *be the significant and non-significant input layer weights in a generating model* $\boldsymbol{\Theta}^\diamond$*. Let* $\phi(\boldsymbol{\Theta})$ *be the parameters* $\boldsymbol{\Theta}$ *with all non-significant input layer weights* $\boldsymbol{V}$ *set to zero. Under Assumption 4.1,*

- *There exists* $c_0 > 0$ *such that* $\| \boldsymbol{U}^k \| \geq c_0$ *for all* $\boldsymbol{\Theta} \in \mathcal{T}^*$ *and* $k = 1, \dots, d^s$ *(where* d^s *is the number of significant features).*
- *If* $\boldsymbol{\Theta} \in \mathcal{T}^*$*, then parameters* $\phi(\boldsymbol{\Theta})$ *also belongs in* $\mathcal{T}^*$*.*

Lemma B.2. *There exist* $c_2, \nu > 0$ *such that:*

$$c_2 d(\boldsymbol{\Theta}, \mathcal{T}^*)^\nu \leq R(\boldsymbol{\Theta}) - R(\boldsymbol{\Theta}^\diamond)$$

for all $\boldsymbol{\Theta} \in \mathcal{T}$*.*

Table 4. Summary of notation.

Notation	**Definitions**
N	Number of training samples.
M	Number of parties.
λ_s, λ_m	The server's and party's regularization coefficients, respectively.
$f(\cdot)$	VFL model label prediction.
$\mathbf{h}(\cdot)$	Party's local embedding function.
$e(\cdot)$	Mean squared error between two embeddings.
$R(\cdot)$	Risk function: MSE with labels for all possible samples.
$R_N(\cdot)$	Empirical risk function: MSE with labels for all training samples.
$G(\cdot)$	Group lasso $L_{2,1}$ regularization term.
$H(\cdot)$	Expected mean-squared difference between two embedding functions.
$H_N(\cdot)$	Empirical mean-squared difference between two embedding functions.
$d(\cdot)$	Distance between a vector and a set of vectors.
$\boldsymbol{\Theta}^{\diamond}$	The generating model parameters defined in Assumption 4.1.
$\hat{\boldsymbol{\Theta}}$	Pre-trained model parameters from minimizing empirical risk.
$\tilde{\boldsymbol{\Theta}}$	Model with the same risk as the generating model closest to pre-trained model.
$\bar{\boldsymbol{\Theta}}$	Learned model parameters after running LESS-VFL.
$\mathbf{U}_m, \mathbf{V}_m$	Input weights on significant and non-significant features in party m's generating model.
$\mathcal{T}^*$	Set of models that have the same risk as the generating model.
$\mathcal{S}^*$	Set of server models that have the same risk as the generating server model.
$\mathcal{C}^*$	Set of client models that have the same risk as the generating client model.
$\mathcal{T}^*_\phi$	Subset of $\mathcal{T}^*$ with non-significant feature weights set to zero.
$\mathcal{S}^*_\phi$	Server models in $\mathcal{T}^*$ with non-significant embedding weights set to zero.
$\mathcal{C}^*_\phi$	Party models in $\mathcal{T}^*$ with non-significant feature weights set to zero.

Lemma B.3. *For any $\delta > 0$, there exists $c_1(\delta) > 0$ such that for all $\boldsymbol{\Theta} \in \mathcal{T}$:*

$$|R_N(\boldsymbol{\Theta}) - R(\boldsymbol{\Theta})| \leq c_1 \frac{\log N}{\sqrt{N}}$$

with probability $1 - \delta$.

We also prove the following lemma:

Lemma B.4. *Given a model $\hat{\boldsymbol{\Theta}}$ defined by* (3)*, for any $\delta > 0$, there exists $C_\delta(\delta) > 0$ and $N \geq N_0(\delta)$ such that:*

$$d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) \leq C_\delta \frac{\log N}{\sqrt{N}} \tag{7}$$

with probability $1 - \delta$.

Proof. Let $[\hat{\boldsymbol{\theta}}_m^\top, \hat{\boldsymbol{\theta}}_s^\top]^\top = \hat{\boldsymbol{\Theta}}$ be the party and server model parameters after the pre-training step. We define $\tilde{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta} \in \mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2 = [\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_m^\top]^\top$ as the optimal model closest to the pre-trained model. By Lemmas B.2 and B.3, we have the following:

$$\begin{aligned}
c_2 d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*)^\mu &= c_2 \|\tilde{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Theta}}\|_2^\mu && (8)\\
&\leq R(\hat{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) && (9)\\
&\leq 2c_1 \frac{\log N}{\sqrt{N}} + R_N(\hat{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R_N(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) && (10)
\end{aligned}$$

Note that $R_N(\hat{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) \leq R_N(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m))$, thus:

$$c_2 d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*)^\mu \leq 2c_1 \frac{\log N}{\sqrt{N}} \tag{11}$$

$$d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) \leq \left(\frac{2c_1}{c_2} \frac{\log N}{\sqrt{N}} \right)^{1/\mu} \tag{12}$$

We let $\mu = 1$. This completes the proof of Lemma B.4. □

B.2. Proof of Lemma 4.5

Proof. Note that the minimization of $H(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ causes $\mathbf{h}(\boldsymbol{\theta}_m; \mathbf{x})^k = \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k$ for all significant embedding components $k \in \mathcal{K}_m$ and any input $\mathbf{x}$. By the definition of $\mathcal{T}^*$ and Definition 2.1, this means that $R(\boldsymbol{\theta}_s^\diamond; \mathbf{h}(\boldsymbol{\theta}_m^\diamond)) = R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) = R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\boldsymbol{\theta}_m))$. Thus, $[\tilde{\boldsymbol{\theta}}_s, \boldsymbol{\theta}_m] \in \mathcal{T}^*$.

By Lemma 3.1 in (Dinh & Ho, 2020), because $\tilde{\boldsymbol{\Theta}} \in \mathcal{T}^*$, $f(\tilde{\boldsymbol{\Theta}}; \mathbf{x}; y) = f(\boldsymbol{\Theta}^\diamond; \mathbf{x}; y)$ for all inputs $\mathbf{x}$. This means that the significant and non-significant features for $f(\boldsymbol{\Theta}^\diamond)$ must be the same for $f(\tilde{\boldsymbol{\Theta}})$. Let $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{z}}$ be the significant and non-significant features for $f(\tilde{\boldsymbol{\Theta}})$. It must be the case that $\tilde{\mathbf{s}} = \mathbf{s}$ and $\tilde{\mathbf{z}} = \mathbf{z}$.

Let $j \in \tilde{\mathbf{z}}$ be a non-significant feature in $f(\tilde{\boldsymbol{\Theta}})$ and let $r \in \mathbb{R}$. Let $g^j(\mathbf{x}, s)$ be a function that replaces $\mathbf{x}^j$ with value s. We know that for all $k \in \mathcal{K}_m$:

$$\mathbf{h}(\boldsymbol{\theta}_m; g^j(\mathbf{x}, r))^k = \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; g^j(\mathbf{x}, r))^k = \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k.$$

In fact, by Proposition 4.4, all embedding components in $\mathcal{K}_m$ only depend on $\tilde{\mathbf{s}}$. Since $\mathbf{h}(\boldsymbol{\theta}_m; \cdot)^k$ for all $k \in \mathcal{K}_m$ is unaffected by features in $\tilde{\mathbf{z}}$, this means the set of non-significant features for $e(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ contains the set of non-significant features for $f(\tilde{\boldsymbol{\Theta}})$: $\mathbf{z}_h \supseteq \tilde{\mathbf{z}}$.

Similarly, let k be a significant feature for $f(\tilde{\boldsymbol{\Theta}})$. By Proposition 4.4, for all $k \in \mathcal{K}_m$:

$$\mathbf{h}(\boldsymbol{\theta}_m; g^k(\mathbf{x}, r))^k = \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; g^k(\mathbf{x}, r))^k \neq \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k$$

for some $r \in \mathbb{R}$. This means that:

$$\sum_{k \in \mathcal{K}_m} (\mathbf{h}(\boldsymbol{\theta}_m; g^k(\mathbf{x}, r))^k - \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k)^2 \neq \sum_{k \in \mathcal{K}_m} (\mathbf{h}(\boldsymbol{\theta}_m; \mathbf{x})^k - \mathbf{h}(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k)^2$$

and we can say that significant features for $e(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_s)$ contains of the set of significant features for $f(\tilde{\boldsymbol{\Theta}})$: $\mathbf{s}_h \supseteq \tilde{\mathbf{s}}$. Therefore, $\mathbf{s}_h = \tilde{\mathbf{s}} = \mathbf{s}$ and $\mathbf{z}_h = \tilde{\mathbf{z}} = \mathbf{z}$. □

B.3. Proof of Theorem 4.6

Next, we prove that the server solving (4) finds an optimal solution that also sets the non-significant embedding component weights to zero. We define $\bar{\boldsymbol{\theta}}_s$ as the server model parameters that solves (4). We start by proving the following lemma:

Lemma B.5. *Let L be the Lipschitz constant for $f(\cdot)$. Given a pre-trained model $\hat{\boldsymbol{\Theta}} = [\hat{\boldsymbol{\theta}}_m^\top, \hat{\boldsymbol{\theta}}_s^\top]^\top$ defined by* (3)*, let $\tilde{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta} \in \mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2 = [\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_m^\top]^\top$ be the optimal model closest to the pre-trained model. For any $\delta > 0$, there exists $C_1(\delta), C_2(\delta), C_3(\delta), C_4(\delta), C_5 > 0$ and $N \geq N_0(\delta)$ such that:*

$$d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}^*) \leq d(\bar{\boldsymbol{\theta}}_s, \{\tilde{\boldsymbol{\theta}}_s\}) \leq \left(C_1 \frac{\log N}{\sqrt{N}} + C_2 \lambda_s^{\nu/(\nu-1)} + C_3 L d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) \right)^{1/\nu} \tag{13}$$

and the sum over the non-significant embedding component weights is

$$\sum_l \| \boldsymbol{V}_s^l \|_2 \leq C_4 \frac{\log N}{\lambda_s \sqrt{N}} + \frac{2L}{\lambda_s} d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + C_5 d(\bar{\boldsymbol{\theta}}_s, \{\tilde{\boldsymbol{\theta}}_s\}). \tag{14}$$

with probability $1 - \delta$.

Proof. Note that $\{\tilde{\boldsymbol{\theta}}_s\}$ is the zero-level set of the analytic function $E(\boldsymbol{\theta}_s) = R(\boldsymbol{\theta}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) - R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m))$. We can apply the Łojasiewicz inequality (Ji et al., 1992) as follows:

$$\begin{aligned} c_2 d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}^*)^\nu &\le c_2 d(\bar{\boldsymbol{\theta}}_s, \{\tilde{\boldsymbol{\theta}}_s\})^\nu && (15) \\ &= c_2 \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2^\nu && (16) \\ &\le R(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) - R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) && (17) \end{aligned}$$

Since $f(\cdot)$ is analytic, we know that the risk function is smooth. Let L be the Lipschitz constant for $R(\cdot)$. For any $\boldsymbol{\theta}_s$ we have:

$$\begin{aligned} \Big| R(\boldsymbol{\theta}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) - R(\boldsymbol{\theta}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) \Big| &\le L \|[\boldsymbol{\theta}_s, \tilde{\boldsymbol{\theta}}_m] - [\boldsymbol{\theta}_s, \hat{\boldsymbol{\theta}}_m]\|_2 && (18) \\ &\le L \|\tilde{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Theta}}\|_2 && (19) \\ &= L d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*). && (20) \end{aligned}$$

Applying (20) and Lemma B.3 to (17) we have:

$$\begin{aligned} c_2 d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}^*)^\nu &\le 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + R(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) && (21) \\ &\le 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + R_N(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R_N(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) && (22) \end{aligned}$$

By the definition of $\bar{\boldsymbol{\theta}}_s$ in (4) we have:

$$\begin{aligned} R_N(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) + \lambda_s G(\bar{\boldsymbol{\theta}}_s) &\le R_N(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) + \lambda_s G(\tilde{\boldsymbol{\theta}}_s) && (23) \\ R_N(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R_N(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) &\le \lambda_s G(\tilde{\boldsymbol{\theta}}_s) - \lambda_s G(\bar{\boldsymbol{\theta}}_s) && (24) \end{aligned}$$

Plugging (24) into (22), and noting that regularizer $G(\cdot)$ is smooth, we have:

$$\begin{aligned} c_2 \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2^\nu &\le 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + \lambda_s (G(\tilde{\boldsymbol{\theta}}_s) - G(\bar{\boldsymbol{\theta}}_s)) && (25) \\ &\le 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + \lambda_s C \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 && (26) \end{aligned}$$

where C is the Lipschitz constant for $G(\cdot)$.

By Young's inequality, we have:

$$\begin{aligned} \lambda_s C \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 &\le \frac{1}{\nu} \left(\frac{(c_2\nu)^{1/\nu}}{2} \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 \right)^\nu + \frac{\nu - 1}{\nu} \left(\frac{2C}{(2c_2)^{1/\nu}} \lambda_s \right)^{\nu/(\nu-1)} && (27) \\ &= \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 + \frac{2(\nu-1)C^{\nu/(\nu-1)}}{\nu (c_2\nu)^{1/(\nu-1)}} \lambda_s^{\nu/(\nu-1)}. && (28) \end{aligned}$$

Let $C_0 = \frac{2(\nu-1)C^{\nu/(\nu-1)}}{\nu(c_2\nu)^{1/(\nu-1)}}$. Plugging (28) into (26) we have:

$$\begin{aligned} c_2 \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2^\nu &\le 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + C_0 \lambda_s^{\nu/(\nu-1)} + \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2^\nu && (29) \\ \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2^\nu &\le 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + C_0 \lambda_s^{\nu/(\nu-1)} && (30) \\ \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 &\le \left(\frac{4c_1}{c_2} \frac{\log N}{\sqrt{N}} + \frac{2C_0}{c_2} \lambda_s^{\nu/(\nu-1)} + \frac{4L}{c_2} d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) \right)^{1/\nu} && (31) \end{aligned}$$

Note that $G(\cdot)$ can be rewritten as $G(\cdot) = \sum_k \| \mathbf{U}_s^k \|_2 + \sum_l \| \mathbf{V}_s^l \|_2$. Let $K(\cdot) = \sum_k \| \mathbf{U}_s^k \|_2$ be the sum of significant embedding component weights in the regularizer $G(\cdot)$. Let $\phi(\boldsymbol{\theta}_s)$ be the parameters $\boldsymbol{\theta}_s$ with all non-significant embedding component weights $\mathbf{V}_s$ set to zero. Note that $R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) \leq R(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m))$ since $[\tilde{\boldsymbol{\theta}}_s, \tilde{\boldsymbol{\theta}}_m] \in \mathcal{T}^*$. Using the definition of $\bar{\boldsymbol{\theta}}_s$ and the smoothness of $K(\cdot)$, we have the following:

$$\lambda_s \sum_l \| \mathbf{V}_s^l \|_2 \leq R_N(\phi(\tilde{\boldsymbol{\theta}}_s); \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) - R_N(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\hat{\boldsymbol{\theta}}_m)) + \lambda_s(K(\phi(\tilde{\boldsymbol{\theta}}_s)) - K(\bar{\boldsymbol{\theta}}_s)) \tag{32}$$

$$\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) - R(\bar{\boldsymbol{\theta}}_s; \mathbf{h}(\tilde{\boldsymbol{\theta}}_m)) + \lambda_s(K(\tilde{\boldsymbol{\theta}}_s) - K(\bar{\boldsymbol{\theta}}_s)) \tag{33}$$

$$\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2Ld(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + \lambda_s C \|\tilde{\boldsymbol{\theta}}_s - \bar{\boldsymbol{\theta}}_s\|_2 \tag{34}$$

$$\sum_l \| \mathbf{V}_s^l \|_2 \leq 2c_1 \frac{\log N}{\lambda_s \sqrt{N}} + \frac{2L}{\lambda_s} d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*) + Cd(\bar{\boldsymbol{\theta}}_s, \{\tilde{\boldsymbol{\theta}}_s\}). \tag{35}$$

This completes the proof of Lemma B.5. □

To complete the proof of Theorem 4.6, we look at the distance of $\bar{\boldsymbol{\theta}}_s$ from the set of parameters $\mathcal{S}_\phi^*$. Note that by Lemma B.1, $\phi(\tilde{\boldsymbol{\theta}}_s) \in \mathcal{S}_\phi^*$. Let $\mathbf{V}_{\boldsymbol{\theta}_s} = \sum_l \| \mathbf{V}_s^l \|_2$ be the sum over non-significant embedding component weights in a model $\boldsymbol{\theta}_s$.

$$d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}_\phi^*) \leq \|\bar{\boldsymbol{\theta}}_s - \phi(\tilde{\boldsymbol{\theta}}_s)\|_2 \tag{36}$$

$$\leq \|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 + \|\phi(\tilde{\boldsymbol{\theta}}_s) - \tilde{\boldsymbol{\theta}}_s\|_2 \tag{37}$$

$$\leq \|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_s} \|_2 \tag{38}$$

$$\leq \|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_s} + \mathbf{V}_{\bar{\boldsymbol{\theta}}_s} - \mathbf{V}_{\bar{\boldsymbol{\theta}}_s} \|_2 \tag{39}$$

$$\leq \|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 + \| \mathbf{V}_{\bar{\boldsymbol{\theta}}_s} \|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_s} - \mathbf{V}_{\bar{\boldsymbol{\theta}}_s} \|_2 \tag{40}$$

$$\leq \|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 + \| \mathbf{V}_{\bar{\boldsymbol{\theta}}_s} \|_2 + C\|\bar{\boldsymbol{\theta}}_s - \tilde{\boldsymbol{\theta}}_s\|_2 \tag{41}$$

The proof of Theorem 4.6 is completed by combining Lemma B.5, Lemma B.4, and (41).

B.4. Proof of Theorem 4.7

Next we prove that the party solving (5) finds the optimal solution and sets all non-significant input layer weights to zero. Following the same proof of Lemma B.3 given by Dinh & Ho (2020), we can prove the following lemma:

Lemma B.6. *For any $\delta > 0$, there exist $c_1(\delta) > 0$ such that for all $\boldsymbol{\theta}, \boldsymbol{\theta}'$ and sets $\mathcal{K}$:*

$$|H_N(\boldsymbol{\theta}; \boldsymbol{\theta}'; \mathcal{K}) - H(\boldsymbol{\theta}; \boldsymbol{\theta}'; \mathcal{K})| \leq c_1 \frac{\log N}{\sqrt{N}}$$

with probability $1 - \delta$.

Let $\bar{\boldsymbol{\theta}}_m$ be the parameters that solve (5). We prove the following lemma:

Lemma B.7. *Let B be the Lipschitz constant for $H(\cdot)$. Let $[\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_m^\top]^\top = \arg\min_{\boldsymbol{\Theta} \in \mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2$ where $\hat{\boldsymbol{\Theta}}$ is the pre-trained model defined in (3). If $\mathcal{K}_m$ in (5) is the set of significant embedding components for $f(\tilde{\boldsymbol{\theta}}_s; \boldsymbol{h}(\tilde{\boldsymbol{\theta}}_m))$, for any $\delta > 0$, there exists $C_1(\delta), C_2(\delta), C_3(\delta), C_4(\delta), C_5 > 0$ and $N \geq N_0(\delta)$ such that:*

$$d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}^*) \leq d(\bar{\boldsymbol{\theta}}_m, \{\tilde{\boldsymbol{\theta}}_m\}) \leq \left(C_1 \frac{\log N}{\sqrt{N}} + C_2 B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + C_3 (\lambda_m)^{\nu/(\nu-1)} \right)^{1/\nu} \tag{42}$$

and the sum over the non-significant input layer weights is

$$\sum_l \| \boldsymbol{V}_m^l \|_2 \leq C_4 \frac{\log N}{\lambda_m \sqrt{N}} + \frac{2B}{\lambda_m} d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + C_5 d(\bar{\boldsymbol{\theta}}_m, \{\tilde{\boldsymbol{\theta}}_m\}) \tag{43}$$

with probability $1 - \delta$.

Proof. Note that $\{\tilde{\boldsymbol{\theta}}_m\}$ is the zero-level set of $H(\bar{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$. Since $H(\cdot)$ is analytic, we can apply the Łojasiewicz inequality as follows:

$$\begin{aligned} c_2 d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}^*)^\nu &\leq c_2 d(\bar{\boldsymbol{\theta}}_m, \{\tilde{\boldsymbol{\theta}}_m\})^\nu && (44) \\ &\leq H(\bar{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m) && (45) \\ &= H(\bar{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m) - H(\tilde{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m) && (46) \end{aligned}$$

where (46) follows from that fact that $H(\boldsymbol{\theta}_m; \boldsymbol{\theta}_m) = 0$ for any $\boldsymbol{\theta}_m$.

Since $H(\cdot)$ is analytic, we know $H(\cdot)$ is smooth. Let B be the Lipschitz constant for $H(\cdot)$. For any $\boldsymbol{\theta}_m$ we have:

$$\begin{aligned} |H(\boldsymbol{\theta}_m; \hat{\boldsymbol{\theta}}_m) - H(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m)| &\leq B \|\hat{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 && (47) \\ &\leq B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) && (48) \end{aligned}$$

Applying (48) and Lemma B.6 to (46):

$$\begin{aligned} c_2 d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}^*)^\nu &\leq 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + H(\bar{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) - H(\tilde{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) && (49) \\ &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + H_N(\bar{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) - H_N(\tilde{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) && (50) \end{aligned}$$

By the definition of $\bar{\boldsymbol{\theta}}_m$ in (5):

$$\begin{aligned} H_N(\bar{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) + \lambda_m G(\bar{\boldsymbol{\theta}}_m) &\leq H_N(\tilde{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) + \lambda_m G(\tilde{\boldsymbol{\theta}}_m) && (51) \\ H_N(\bar{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) - H_N(\tilde{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m) &\leq \lambda_m (G(\tilde{\boldsymbol{\theta}}_m) - G(\bar{\boldsymbol{\theta}}_m)) && (52) \end{aligned}$$

Plugging (52) into (50):

$$\begin{aligned} c_2 d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}^*)^\nu &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + \lambda_m (G(\tilde{\boldsymbol{\theta}}_m) - G(\bar{\boldsymbol{\theta}}_m)) && (53) \\ &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + \lambda_m C \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2 && (54) \end{aligned}$$

where C is the Lipschitz constant for $G(\cdot)$.

By Young's inequality:

$$\begin{aligned} \lambda_m C \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2 &\leq \frac{1}{\nu} \left(\frac{(c_2 \nu)^{1/\nu}}{2} \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2 \right)^\nu + \frac{\nu - 1}{\nu} \left(\frac{2C}{(2c_2)^{1/\nu}} \lambda_s \right)^{\nu/(\nu-1)} && (55) \\ &= \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2 + \frac{2(\nu - 1) C^{\nu/(\nu-1)}}{\nu (c_2 \nu)^{1/(\nu-1)}} \lambda_m^{\nu/(\nu-1)}. && (56) \end{aligned}$$

Let $C_0 = \frac{2(\nu-1)C^{\nu/(\nu-1)}}{\nu(c_2\nu)^{1/(\nu-1)}}$. Applying (56) to (54) we have:

$$\begin{aligned} c_2 \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2^\nu &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + C_0 (\lambda_m)^{\nu/(\nu-1)} + \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2^\nu && (57) \\ \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2^\nu &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + C_0 (\lambda_m)^{\nu/(\nu-1)} && (58) \\ \frac{c_2}{2} \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2^\nu &\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2B d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + C_0 (\lambda_m)^{\nu/(\nu-1)} && (59) \\ d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}^*) \leq d(\bar{\boldsymbol{\theta}}_m, \{\tilde{\boldsymbol{\theta}}_m\}) &\leq \left(\frac{4c_1}{c_2} \frac{\log N}{\sqrt{N}} + \frac{4B}{c_2} d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + \frac{2C_0}{c_2} (\lambda_m)^{\nu/(\nu-1)} \right)^{1/\nu} && (60) \end{aligned}$$

Note that $G(\cdot)$ can be rewritten as $G(\cdot) = \sum_k \| \mathbf{U}_m^k \|_2 + \sum_l \| \mathbf{V}_m^l \|_2$. Let $K(\cdot) = \sum_k \| \mathbf{U}_m^k \|_2$ be the sum of significant input layer weights in the regularizer $G(\cdot)$. Let $\phi(\boldsymbol{\theta}_m)$ be the parameters $\boldsymbol{\theta}_m$ with all non-significant input layer weights $\mathbf{V}_m$ set to zero. Note that under our assumption that $\mathcal{K}_m$ only contains significant embedding components and Proposition 4.4, $H(\phi(\tilde{\boldsymbol{\theta}}_m); \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m) = H(\tilde{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m) = 0$, because non-significant features have no effect on significant embedding components. By the definition of $\bar{\boldsymbol{\theta}}_m$:

$$\begin{aligned}
\lambda_m \sum_l \| \mathbf{V}_m^l \|_2 &\leq H_N(\phi(\tilde{\boldsymbol{\theta}}_m); \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m) - H_N(\bar{\boldsymbol{\theta}}_m; \hat{\boldsymbol{\theta}}_m; \mathcal{K}_m) + \lambda_m (K(\phi(\tilde{\boldsymbol{\theta}}_m)) - K(\bar{\boldsymbol{\theta}}_m)) && (61) \\
&\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2Bd(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + H(\phi(\tilde{\boldsymbol{\theta}}_m); \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m) - H(\bar{\boldsymbol{\theta}}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m) + \lambda_m (K(\tilde{\boldsymbol{\theta}}_m) - K(\bar{\boldsymbol{\theta}}_m)) && (62) \\
&\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2Bd(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + \lambda_m (K(\tilde{\boldsymbol{\theta}}_m) - K(\bar{\boldsymbol{\theta}}_m)) && (63) \\
&\leq 2c_1 \frac{\log N}{\sqrt{N}} + 2Bd(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + \lambda_m C \|\tilde{\boldsymbol{\theta}}_m - \bar{\boldsymbol{\theta}}_m\|_2 && (64) \\
\sum_l \| \mathbf{V}_m^l \|_2 &\leq 2c_1 \frac{\log N}{\lambda_m \sqrt{N}} + \frac{2B}{\lambda_m} d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) + Cd(\bar{\boldsymbol{\theta}}_m, \{\tilde{\boldsymbol{\theta}}_m\}) && (65)
\end{aligned}$$

This completes the proof of Lemma B.5. □

To complete the proof of Theorem 4.7, we look at the distance of $\bar{\boldsymbol{\theta}}_m$ from the set of parameters $\mathcal{C}_\phi^*$. Note that by Lemma B.1, $\phi(\tilde{\boldsymbol{\theta}}_m) \in \mathcal{C}_\phi^*$. Let $\mathbf{V}_{\boldsymbol{\theta}_m} = \sum_l \| \mathbf{V}_m^l \|_2$ be the sum over non-significant feature weights in a model $\boldsymbol{\theta}_m$.

$$\begin{aligned}
d(\bar{\boldsymbol{\theta}}_m, \mathcal{C}_\phi^*) &\leq \|\bar{\boldsymbol{\theta}}_m - \phi(\boldsymbol{\theta}_s')\|_2 && (66) \\
&\leq \|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 + \|\phi(\tilde{\boldsymbol{\theta}}_m) - \tilde{\boldsymbol{\theta}}_m\|_2 && (67) \\
&\leq \|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_m} \|_2 && (68) \\
&\leq \|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_m} + \mathbf{V}_{\bar{\boldsymbol{\theta}}_m} - \mathbf{V}_{\bar{\boldsymbol{\theta}}_m} \|_2 && (69) \\
&\leq \|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 + \| \mathbf{V}_{\bar{\boldsymbol{\theta}}_m} \|_2 + \| \mathbf{V}_{\tilde{\boldsymbol{\theta}}_m} - \mathbf{V}_{\bar{\boldsymbol{\theta}}_m} \|_2 && (70) \\
&\leq \|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2 + \| \mathbf{V}_{\bar{\boldsymbol{\theta}}_m} \|_2 + C\|\bar{\boldsymbol{\theta}}_m - \tilde{\boldsymbol{\theta}}_m\|_2. && (71)
\end{aligned}$$

Note that:

$$d(\hat{\boldsymbol{\theta}}_m, \mathcal{C}^*) = \|\tilde{\boldsymbol{\theta}}_m - \hat{\boldsymbol{\theta}}_m\|_2 \leq \|\tilde{\boldsymbol{\Theta}} - \hat{\boldsymbol{\Theta}}\|_2 = d(\hat{\boldsymbol{\Theta}}, \mathcal{T}^*). \qquad (72)$$

The proof of Theorem 4.7 is completed by combining Lemma B.7, (71), (72), and Lemma B.4.

B.5. Extension to $M > 1$ Parties

Proposition B.8. *Consider a model $\boldsymbol{\Theta} = [\boldsymbol{\theta}_s^\top, \boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_M^\top]^\top$. Let $\boldsymbol{s}$ and $\boldsymbol{z}$ be the sets of significant and non-significant features for $f(\boldsymbol{\Theta})$, respectively. Let the set of significant embedding components for $f(\boldsymbol{\theta}_s; \boldsymbol{h}_1(\boldsymbol{\theta}_1^\diamond); \ldots; \boldsymbol{h}_M(\boldsymbol{\theta}_M^\diamond))$ be $\boldsymbol{s}_s$. Let $g^j(\boldsymbol{x}_m, r)$ replace input $\boldsymbol{x}_m^j$ with value r. For each significant embedding component $k \in \boldsymbol{s}_s$, for all $j \in \boldsymbol{z}$ and $m \in [M]$, and any $r \in \mathbb{R}$, $\boldsymbol{h}_m(\boldsymbol{\theta}_m; \boldsymbol{x}_m)^k = \boldsymbol{h}_m(\boldsymbol{\theta}_m; g^j(\boldsymbol{x}_m, r))^k$.*

Proof. Suppose that for a party m, $\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m)^k \neq \mathbf{h}_m(\boldsymbol{\theta}_m; g^j(\mathbf{x}_m, r))^k$ for some significant embedding component $k \in \mathbf{s}_s$, non-significant feature $j \in \mathbf{z}$, and $r \in \mathbb{R}$. By our supposition and since component k is significant,

$$f(\boldsymbol{\theta}_s; \mathbf{h}_1(\boldsymbol{\theta}_1; \mathbf{x}_1); \ldots; \mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m); \ldots; \mathbf{h}_M(\boldsymbol{\theta}_M; \mathbf{x}_M)) \neq f(\boldsymbol{\theta}_s; \mathbf{h}_1(\boldsymbol{\theta}_1; \mathbf{x}_1); \ldots; \mathbf{h}_m(\boldsymbol{\theta}_m; g^j(\mathbf{x}_m; r)); \ldots; \mathbf{h}_M(\boldsymbol{\theta}_M; \mathbf{x}_M))$$

for some value $r \in \mathbb{R}$. This contradicts the fact that j is a non-significant feature. □

Lemma B.9. *Let $\tilde{\boldsymbol{\Theta}} = [\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_1^\top, \dots, \tilde{\boldsymbol{\theta}}_M^\top]^\top \in \mathcal{T}^*$. Let $\boldsymbol{s}$ and $\boldsymbol{z}$ be the significant and non-significant features for $f(\boldsymbol{\Theta}^\diamond)$. Let $\mathcal{K}_m$ be the subset of significant embedding components for $f(\tilde{\boldsymbol{\theta}}_s; \boldsymbol{h}_1(\tilde{\boldsymbol{\theta}}_1); \dots; \boldsymbol{h}_M(\tilde{\boldsymbol{\theta}}_M))$ in the embedding vector $\boldsymbol{h}_m(\tilde{\boldsymbol{\theta}}_m)$. Let $\boldsymbol{\theta}_m = \arg\min_{\boldsymbol{\theta}'_m} H(\boldsymbol{\theta}'_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ for all parties m. Let $\boldsymbol{s}_{h_m}$ and $\boldsymbol{z}_{h_m}$ be the significant and non-significant features at each party m for $e(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ with parameters $\boldsymbol{\theta}_m$. Then:*

$$[\tilde{\boldsymbol{\theta}}_s, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M] \in \mathcal{T}^*, \ \bigcup_{m=1}^{M} \boldsymbol{s}_{h_m} = \boldsymbol{s}, \ \textit{and} \ \bigcup_{m=1}^{M} \boldsymbol{z}_{h_m} = \boldsymbol{z}\,.$$

Proof. Note that the minimization of $H(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ causes $\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m)_i = \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; \mathbf{x}_m)_i$ for all significant embedding components $i \in \mathcal{K}_m$ and any input $\mathbf{x}_m$. By the definition of $\mathcal{T}^*$ and Definition 2.1, this means that

$$R(\boldsymbol{\theta}_s^\diamond; \mathbf{h}_1(\boldsymbol{\theta}_1^\diamond); \dots; \mathbf{h}_M(\boldsymbol{\theta}_M^\diamond)) = R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}_1(\tilde{\boldsymbol{\theta}}_1); \dots; \mathbf{h}_M(\tilde{\boldsymbol{\theta}}_M)) = R(\tilde{\boldsymbol{\theta}}_s; \mathbf{h}_1(\boldsymbol{\theta}_1); \dots; \mathbf{h}_M(\boldsymbol{\theta}_M)).$$

Thus, $[\tilde{\boldsymbol{\theta}}_s, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M] \in \mathcal{T}^*$.

By Lemma 3.1 in (Dinh & Ho, 2020), because $\tilde{\boldsymbol{\Theta}} \in \mathcal{T}^*$, $f(\tilde{\boldsymbol{\Theta}}; \mathbf{x}; y) = f(\boldsymbol{\Theta}^\diamond; \mathbf{x}; y)$ for all inputs $\mathbf{x}$. This means that the significant and non-significant features for $f(\boldsymbol{\Theta}^\diamond)$ must be the same for $f(\tilde{\boldsymbol{\Theta}})$. Let $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{z}}$ be the significant and non-significant features for $f(\tilde{\boldsymbol{\Theta}})$. It must be the case that $\tilde{\mathbf{s}} = \mathbf{s}$ and $\tilde{\mathbf{z}} = \mathbf{z}$.

Let $j \in \tilde{\mathbf{z}}_m$ be a non-significant feature for some party m in $f(\tilde{\boldsymbol{\Theta}})$ and let $r \in \mathbb{R}$. Let $g^j(\cdot)$ be defined the same as in Definition 2.1. We know that for all $k \in \mathcal{K}_m$:

$$\mathbf{h}_m(\boldsymbol{\theta}_m; g^j(\mathbf{x}_m, r))^k = \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; g^j(\mathbf{x}_m, r))^k = \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; \mathbf{x}_m)^k$$

because by Proposition 4.4, all embedding components in $\mathcal{K}_m$ only depend on $\tilde{\mathbf{s}}_m$. Since $\mathbf{h}_m(\boldsymbol{\theta}_m; \cdot)^k$ is unaffected by features in $\tilde{\mathbf{z}}_m$, this means the set of non-significant features for $e(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ contains the set of non-significant features for $f(\tilde{\boldsymbol{\Theta}})$ at party m: $\mathbf{z}_{h_m} \supseteq \tilde{\mathbf{z}}_m$.

Similarly, let $k \in \tilde{\mathbf{s}}_m$ be a significant feature for some party m in $f(\tilde{\boldsymbol{\Theta}})$. By Proposition B.8, for some $k \in \mathcal{K}_m$:

$$\mathbf{h}_m(\boldsymbol{\theta}_m; g^k(\mathbf{x}_m, r))^k = \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; g^k(\mathbf{x}_m, r))^k \neq \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; \mathbf{x})^k$$

for some $r \in \mathbb{R}$. This means that:

$$\sum_{k \in \mathcal{K}_m} (\mathbf{h}_m(\boldsymbol{\theta}_m; g^k(\mathbf{x}_m, r))^k - \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; \mathbf{x}_m)^k)^2 \neq \sum_{k \in \mathcal{K}_m} (\mathbf{h}_m(\boldsymbol{\theta}_m; \mathbf{x}_m)^k - \mathbf{h}_m(\tilde{\boldsymbol{\theta}}_m; \mathbf{x}_m)^k)^2$$

and we can say that significant features for $e(\boldsymbol{\theta}_m; \tilde{\boldsymbol{\theta}}_m; \mathcal{K}_m)$ contains the set of significant features for $f(\tilde{\boldsymbol{\Theta}})$ at party m: $\mathbf{s}_{h_m} \supseteq \tilde{\mathbf{s}}_m$.

Since for each party m, $\mathbf{s}_{h_m} \cap \mathbf{z}_{h_m} = \emptyset$, $\mathbf{s}_{h_m} = \tilde{\mathbf{s}}_m$ and $\mathbf{z}_{h_m} = \tilde{\mathbf{z}}_m$. Since for parties $m \neq j$, $\tilde{\mathbf{s}}_m \cap \mathbf{s}'_j = \emptyset$ and $\tilde{\mathbf{z}}_m \cap \mathbf{z}'_j = \emptyset$, $\bigcup\limits_{m=1}^{M} \mathbf{s}_{h_m} = \tilde{\mathbf{s}} = \mathbf{s}$ and $\bigcup\limits_{m=1}^{M} \mathbf{z}_{h_m} = \tilde{\mathbf{z}} = \mathbf{z}$. This completes the proof of Lemma B.9. □

We redefine $\mathcal{S}_\phi^*$ for $M > 1$:

$$\mathcal{S}_\phi^* = \{\boldsymbol{\theta}_s : \exists\, \boldsymbol{\theta}_m \ \forall m = 1, \dots, M \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M] \in \mathcal{T}^* \text{ and } \mathbf{V}_s = \mathbf{0}\}.$$

We bound the distance $d(\boldsymbol{\theta}_s, \mathcal{S}_\phi^*)$ in the following theorem.

Theorem B.10. *Let L be the Lipschitz constant for $f(\cdot)$. Given a pre-trained model $\hat{\boldsymbol{\Theta}}$ defined by* (3)*, for any $\delta > 0$, there exists $C_N, C_\delta(\delta) > 0$ and $N \geq N_0(\delta)$ such that:*

$$d(\boldsymbol{\theta}_s, \mathcal{S}_\phi^*) \leq L C_N \frac{\log N}{\lambda_s \sqrt{N}} + L C_\delta \left(\frac{\log N}{\sqrt{N}} + \lambda_s^{\nu/(\nu-1)} \right)^{1/\nu} \tag{73}$$

with probability $1 - \delta$. If $\lambda_s \sim N^{-1/4}$, then with probability $1 - \delta$ there exists $C(\delta) > 0$ such that:

$$d(\boldsymbol{\theta}_s, \mathcal{S}_\phi^*) \leq L C \left(\frac{\log N}{N} \right)^{\frac{1}{4(\nu-1)}} \tag{74}$$

Proof. The proof of Theorem B.10 is the same as the proof of Theorem 4.6 in Appendix B.3 when replacing $R(\boldsymbol{\theta}_s; \mathbf{h}(\boldsymbol{\theta}_m))$ with $R(\boldsymbol{\theta}_s; \mathbf{h}_1(\boldsymbol{\theta}_1); \ldots \mathbf{h}_M(\boldsymbol{\theta}_M))$. □

We define the set of party m parameters in $\mathcal{T}^*$ that have the weights on local non-significant features set to zero as:

$$\mathcal{C}_m^* = \{\boldsymbol{\theta}_m : \exists\, \boldsymbol{\theta}_s \text{ and } \boldsymbol{\theta}_j \ \forall j \neq m \text{ s.t. } [\boldsymbol{\theta}_s, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m, \ldots, \boldsymbol{\theta}_M] \in \mathcal{T}^* \text{ and } \mathbf{V}_m = \mathbf{0}\}.$$

We bound the distance $d(\boldsymbol{\theta}_m, \mathcal{C}_m^*) \to 0$ in the following theorem.

Theorem B.11. *Let B be the Lipschitz constant for $H(\cdot)$. Let $[\tilde{\boldsymbol{\theta}}_s^\top, \tilde{\boldsymbol{\theta}}_1^\top, \ldots, \tilde{\boldsymbol{\theta}}_M^\top]^\top = \arg\min_{\boldsymbol{\Theta} \in \mathcal{T}^*} \|\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\|_2$ where $\hat{\boldsymbol{\Theta}}$ is the pre-trained model defined in* (3)*. If $\mathcal{K}_m$ in* (5) *is the subset of significant embedding components for $f(\tilde{\boldsymbol{\theta}}_s; \boldsymbol{h}_1(\tilde{\boldsymbol{\theta}}_1); \ldots; \boldsymbol{h}_M(\tilde{\boldsymbol{\theta}}_M))$ in $\boldsymbol{h}_m(\tilde{\boldsymbol{\theta}}_m)$, then for each party m, for any $\delta > 0$, there exists $C_N, C_\delta(\delta) > 0$ and $N \geq N_0(\delta)$ such that:*

$$d(\boldsymbol{\theta}_m, \mathcal{C}_m^*) \leq BC_N \frac{\log N}{\lambda_m \sqrt{N}} + C_\delta \left(B \frac{\log N}{\sqrt{N}} + (\lambda_m)^{\nu/(\nu-1)} \right)^{1/\nu} \tag{75}$$

with probability $1 - \delta$. If $\lambda_m \sim N^{-1/4}$, then with probability $1 - \delta$ there exists $C_m(\delta) > 0$ such that:

$$d(\boldsymbol{\theta}_m, \mathcal{C}_m^*) \leq BC_m \left(\frac{\log N}{N} \right)^{\frac{1}{4(\nu-1)}} \tag{76}$$

Theorem B.11 follows from applying the proof of Theorem 4.7 to each party m, replacing $\mathcal{C}_\phi^*$ with $\mathcal{C}_m^*$.

B.6. Proof of Theorem 4.3

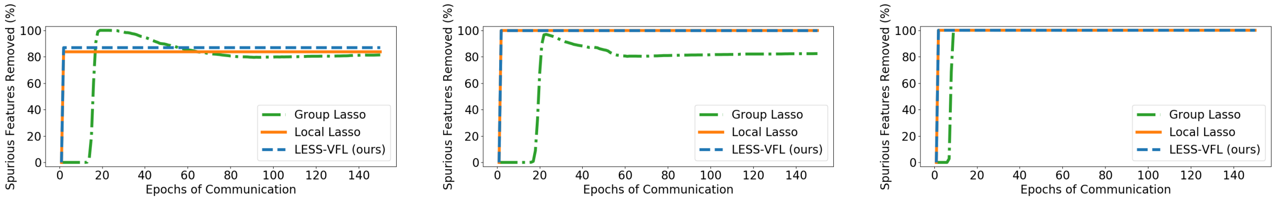
Let constant C be defined as in Theorem B.10 and let constant C_m be defined the same as in Theorem B.11 for all parties m. Let B_m be the Lipschitz constant of $H(\cdot)$ at party m. Then by Theorems B.10 and B.11, with probability $1 - \delta$:

$$\begin{aligned}
d(\bar{\boldsymbol{\Theta}}, \mathcal{T}_\phi^*) &= \sqrt{d(\bar{\boldsymbol{\theta}}_s, \mathcal{S}_\phi^*)^2 + d(\bar{\boldsymbol{\theta}}_1, \mathcal{C}_1^\diamond)^2 + \ldots + d(\bar{\boldsymbol{\theta}}_M, \mathcal{C}_m^*)^2} &(77)\\
&\leq \sqrt{L^2 C^2 \left(\frac{\log N}{N} \right)^{\frac{1}{2(\nu-1)}} + \left(\frac{\log N}{N} \right)^{\frac{1}{2(\nu-1)}} \sum_{m=1}^{M} B_m^2 C_m^2} &(78)\\
&\leq \sqrt{\left(L^2 C^2 + \sum_{m=1}^{M} B_m^2 C_m^2 \right)} \left(\frac{\log N}{N} \right)^{\frac{1}{4(\nu-1)}} &(79)\\
&= O\left(\sqrt{M} \left(\frac{\log N}{N} \right)^{\frac{1}{4(\nu-1)}} \right). &(80)
\end{aligned}$$

C. Additional Experimental Results

We now provide additional experimental results. We use the same experimental setup as described in Section 5, and provide results for the datasets that were not included previously (MIMIC-III, Gina, Sylva). We also include a complete results from the grid search, showing the percentage of spurious feature removed and final training accuracy of group lasso, local lasso, and LESS-VFL with different regularization parameters.

In Figure 5, we plot the percentage of spurious features removed over 150 communication epochs of training in the MIMIC-III, Gina, and Sylva datasets. For MIMIC-III and Sylva, we can see that all method perform similarly in terms of removing spurious features quickly, though group lasso lags behind the other methods by a few communication rounds. In the case of Gina, group lasso takes about 20 additional communication epochs to start removing spurious features, and oscillates before settling at a percentage lower than the other methods. Reinforcing the takeaways from the main paper, by allowing feature selection to take place with minimal upfront communication, spurious features can be removed in fewer communication rounds compared to group lasso.

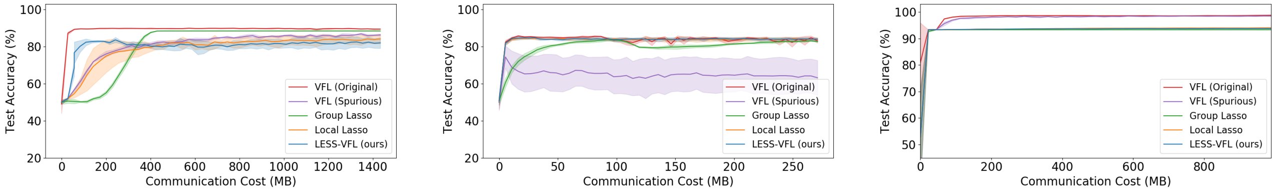


(a) MIMIC-III

(b) Gina

(c) Sylva

Figure 5. Percentage of spurious features removed over 150 communication epochs. The values shown is the average of 5 runs. Group Lasso gradually removes features while local lasso and LESS-VFL remove features with one round of communication after pre-training.



(a) MIMIC-III

(b) Gina

(c) Sylva

Figure 6. Test accuracy over the first 50 communication epochs. The solid lines are the average of 5 runs and the shaded region represents the standard deviation.

In Figure 6, we plot the test accuracy against communication cost for all baselines. For both MIMIC-III and Sylva, the inclusion of spurious features does not have a large detrimental effect on the VFL test accuracy. In this case, it is important that applying the feature selection methods do not lead to model performance becoming worse than if we had not removed any spurious features. In the case of MIMIC-III, all methods achieve similar test accuracy. However, for the Sylva dataset, group lasso is unable to achieve the same accuracy as the other methods in the first 50 communication epochs. For the Gina dataset, all feature selection methods achieve test accuracy similar to the VFL baseline without spurious features, although group lasso takes more communication rounds to converge.

In Table 5, we provide the communication cost to reach 90% of the baseline VFL (original) test accuracy and remove 80% of spurious features for different amount of pre-training epochs. We show the communication cost taken during pre-training and post feature selection (Post-FS) as well as the total communication cost. The values shown are the average of five runs, $\pm$ the standard deviation. We can see that in many cases, LESS-VFL has zero cost for post feature selection. This indicates that LESS-VFL removed spurious features and achieves high accuracy during feature selection itself. We can see that LESS-VFL always achieves the same or lower communication cost than local lasso. Additionally, we see that in the Phishing dataset, local lasso requires more pre-training epochs in order to achieve its lowest communication cost to reach the thresholds. In the Activity dataset, LESS-VFL always costs less communication than local lasso between 1 and 5 pre-training epochs. Local lasso's lowest communication cost is 26.56 MB, while LESS-VFL's highest communication cost is 21.88 MB.

In Tables 6, 7, and 8, we provide the results of our grid search, used to determine the best regularization parameters for each method. We provide the final training accuracy and percentage of spurious features removed for group lasso, local lasso, and LESS-VFL using different regularization values: (λ_m, λ_s). Note that the server regularization parameter λ_s only applies to LESS-VFL. The values shown are the average of five runs, $\pm$ the standard deviation.

Table 5. Communication cost to reach 90% of baseline VFL (original) test accuracy and remove 80% of spurious features for different amount of pre-training epochs. All values are the average of 5 runs. Bold values are the lowest communication cost achieved by that method on the dataset. 'Pretrain' is the communication cost during pre-training, 'Post-FS' is the communication cost during training after feature selection is complete, and 'Total' is the sum of the previous.

Dataset	**Pre-training Epochs**	**Communication Cost (MB)**					
		Local Lasso			LESS-VFL (ours)		
		Pretrain	Post-FS	Total	Pretrain	Post-FS	Total
Activity	1	3.59	30.15	33.74	3.59	15.97	19.56
	2	5.38	26.56	31.95	5.38	16.49	21.88
	3	7.18	19.39	**26.56**	7.18	10.90	18.08
	4	8.97	21.54	30.51	8.97	7.70	**16.68**
	5	10.77	18.85	29.62	10.77	10.51	21.28
Phishing	1	3.24	4.86	8.10	3.24	0.75	**3.99**
	2	4.86	1.62	**6.48**	4.86	0.38	5.23
	3	6.48	1.62	8.10	6.48	0.00	6.48
	4	8.10	1.62	9.72	8.10	0.00	8.10
	5	9.72	1.62	11.34	9.72	0.00	9.72
MIMIC-III	1	7.17	0.00	**7.17**	7.17	0.00	**7.17**
	2	10.75	0.00	10.75	10.75	0.00	10.75
	3	14.34	0.00	14.34	14.34	0.00	14.34
	4	17.92	0.00	17.92	17.92	0.00	17.92
	5	21.51	0.00	21.51	21.51	0.00	21.51
Gina	1	1.35	0.54	**1.90**	1.35	0.13	**1.48**
	2	2.03	0.00	2.03	2.03	0.00	2.03
	3	2.71	0.00	2.71	2.71	0.00	2.71
	4	3.39	0.00	3.39	3.39	0.00	3.39
	5	4.06	0.00	4.06	4.06	0.00	4.06
Sylva	1	5.62	0.00	**5.62**	5.62	0.00	**5.62**
	2	8.43	0.00	8.43	8.43	0.00	8.43
	3	11.25	0.00	11.25	11.25	0.00	11.25
	4	14.06	0.00	14.06	14.06	0.00	14.06
	5	16.87	0.00	16.87	16.87	0.00	16.87

Table 6. Training accuracy and percentage of spurious features removed for the Activity and Phishing datasets.

Dataset	**Regularizer Coefficients** (λ_m, λ_s)	**Group Lasso**		**Local Lasso**		**LESS-VFL (ours)**	
		Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed
Activity	$(2.0, 0.5)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	47.22 ± 2.49	100.00 ± 0.00
	$(2.0, 0.25)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	69.38 ± 1.75	100.00 ± 0.00
	$(2.0, 0.1)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	73.53 ± 0.71	100.00 ± 0.00
	$(2.0, 0.05)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	74.38 ± 0.61	100.00 ± 0.00
	$(1.0, 0.5)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	50.00 ± 0.29	100.00 ± 0.00
	$(1.0, 0.25)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	69.38 ± 1.75	100.00 ± 0.00
	$(1.0, 0.1)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	73.53 ± 0.71	100.00 ± 0.00
	$(1.0, 0.05)$	18.22 ± 0.00	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	74.38 ± 0.61	100.00 ± 0.00
	$(0.5, 0.5)$	25.37 ± 8.79	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	45.95 ± 8.65	99.46 ± 0.54
	$(0.5, 0.25)$	25.37 ± 8.79	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	72.96 ± 2.34	100.00 ± 0.00
	$(0.5, 0.1)$	25.37 ± 8.79	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	73.95 ± 1.02	100.00 ± 0.00
	$(0.5, 0.05)$	25.37 ± 8.79	100.00 ± 0.00	74.18 ± 0.77	100.00 ± 0.00	74.38 ± 0.61	100.00 ± 0.00
	$(0.25, 0.5)$	57.10 ± 1.74	100.00 ± 0.00	73.72 ± 4.78	100.00 ± 0.00	45.54 ± 7.09	91.90 ± 3.76
	$(0.25, 0.25)$	57.10 ± 1.74	100.00 ± 0.00	73.72 ± 4.78	100.00 ± 0.00	78.66 ± 3.40	100.00 ± 0.00
	$(0.25, 0.1)$	57.10 ± 1.74	100.00 ± 0.00	73.72 ± 4.78	100.00 ± 0.00	74.02 ± 7.81	100.00 ± 0.00
	$(0.25, 0.05)$	57.10 ± 1.74	100.00 ± 0.00	73.72 ± 4.78	100.00 ± 0.00	73.73 ± 5.01	100.00 ± 0.00
	$(0.1, 0.5)$	75.47 ± 1.93	88.93 ± 3.06	86.75 ± 2.04	100.00 ± 0.00	49.68 ± 4.03	59.05 ± 7.24
	$(0.1, 0.25)$	75.47 ± 1.93	88.93 ± 3.06	86.75 ± 2.04	100.00 ± 0.00	86.70 ± 3.13	87.93 ± 8.91
	$(0.1, 0.1)$	75.47 ± 1.93	88.93 ± 3.06	86.75 ± 2.04	100.00 ± 0.00	88.31 ± 0.74	99.64 ± 0.71
	$(0.1, 0.05)$	75.47 ± 1.93	88.93 ± 3.06	86.75 ± 2.04	100.00 ± 0.00	87.14 ± 1.86	99.93 ± 0.14
	$(0.05, 0.5)$	89.98 ± 2.60	1.71 ± 0.61	90.17 ± 2.02	0.64 ± 0.35	43.91 ± 5.10	0.48 ± 0.34
	$(0.05, 0.25)$	89.98 ± 2.60	1.71 ± 0.61	90.17 ± 2.02	0.64 ± 0.35	89.13 ± 2.36	0.50 ± 0.36
	$(0.05, 0.1)$	89.98 ± 2.60	1.71 ± 0.61	90.17 ± 2.02	0.64 ± 0.35	87.75 ± 2.18	0.57 ± 0.43
	$(0.05, 0.05)$	89.98 ± 2.60	1.71 ± 0.61	90.17 ± 2.02	0.64 ± 0.35	91.11 ± 1.58	0.64 ± 0.35
	$(0.01, 0.5)$	89.05 ± 2.24	0.00 ± 0.00	90.45 ± 1.12	0.00 ± 0.00	43.91 ± 5.10	0.00 ± 0.00
	$(0.01, 0.25)$	89.05 ± 2.24	0.00 ± 0.00	90.45 ± 1.12	0.00 ± 0.00	90.15 ± 1.86	0.00 ± 0.00
	$(0.01, 0.1)$	89.05 ± 2.24	0.00 ± 0.00	90.45 ± 1.12	0.00 ± 0.00	89.39 ± 1.61	0.00 ± 0.00
	$(0.01, 0.05)$	89.05 ± 2.24	0.00 ± 0.00	90.45 ± 1.12	0.00 ± 0.00	87.97 ± 2.30	0.00 ± 0.00
Phishing	$(2.0, 0.01)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(2.0, 0.005)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(1.0, 0.01)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(1.0, 0.005)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(0.5, 0.01)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(0.5, 0.005)$	55.63 ± 0.00	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(0.25, 0.01)$	89.26 ± 1.96	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	53.38 ± 4.50	100.00 ± 0.00
	$(0.25, 0.005)$	89.26 ± 1.96	100.00 ± 0.00	90.27 ± 0.67	100.00 ± 0.00	51.13 ± 5.52	100.00 ± 0.00
	$(0.1, 0.01)$	91.71 ± 0.20	84.00 ± 5.33	78.96 ± 15.00	92.00 ± 9.80	78.98 ± 15.01	90.67 ± 9.04
	$(0.1, 0.005)$	91.71 ± 0.20	84.00 ± 5.33	78.96 ± 15.00	92.00 ± 9.80	92.45 ± 0.00	93.33 ± 0.00
	$(0.05, 0.01)$	91.85 ± 0.43	0.00 ± 0.00	92.06 ± 0.11	0.00 ± 0.00	91.92 ± 0.11	0.00 ± 0.00
	$(0.05, 0.005)$	91.85 ± 0.43	0.00 ± 0.00	92.06 ± 0.11	0.00 ± 0.00	92.06 ± 0.12	0.00 ± 0.00
	$(0.01, 0.01)$	91.86 ± 0.21	0.00 ± 0.00	92.00 ± 0.11	0.00 ± 0.00	92.00 ± 0.12	0.00 ± 0.00
	$(0.01, 0.005)$	91.86 ± 0.21	0.00 ± 0.00	92.00 ± 0.11	0.00 ± 0.00	91.88 ± 0.26	0.00 ± 0.00

Table 7. Training accuracy and percentage of spurious features removed in MIMIC-III dataset. A '–' means that the experiments with this regularization parameter choice was not run.

Dataset	**Regularizer Coefficients** (λ_m, λ_s)	**Group Lasso**		**Local Lasso**		**LESS-VFL (ours)**	
		Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed
MIMIC-III	$(40.0, 0.5)$	–	–	81.97 ± 2.64	100.00 ± 0.00	81.12 ± 2.08	99.89 ± 0.22
	$(40.0, 0.25)$	–	–	81.97 ± 2.64	100.00 ± 0.00	80.60 ± 1.59	100.00 ± 0.00
	$(40.0, 0.1)$	–	–	81.97 ± 2.64	100.00 ± 0.00	80.61 ± 2.62	100.00 ± 0.00
	$(40.0, 0.05)$	–	–	81.97 ± 2.64	100.00 ± 0.00	80.82 ± 1.44	99.21 ± 1.57
	$(35.0, 0.5)$	–	–	80.94 ± 3.45	98.60 ± 2.81	81.28 ± 1.89	96.07 ± 5.68
	$(35.0, 0.25)$	–	–	80.94 ± 3.45	98.60 ± 2.81	80.96 ± 1.92	98.60 ± 2.81
	$(35.0, 0.1)$	–	–	80.94 ± 3.45	98.60 ± 2.81	81.79 ± 1.03	98.43 ± 3.15
	$(35.0, 0.05)$	–	–	80.94 ± 3.45	98.60 ± 2.81	80.23 ± 2.09	98.71 ± 2.58
	$(32.5, 0.1)$	–	–	83.45 ± 2.47	87.53 ± 10.81	81.85 ± 2.66	89.94 ± 10.14
	$(32.5, 0.05)$	–	–	83.45 ± 2.47	87.53 ± 10.81	82.56 ± 1.37	86.85 ± 10.21
	$(30.0, 0.5)$	–	–	84.45 ± 2.08	76.24 ± 6.35	84.39 ± 2.84	66.15 ± 13.21
	$(30.0, 0.25)$	–	–	84.45 ± 2.08	76.24 ± 6.35	84.26 ± 2.26	76.12 ± 4.97
	$(30.0, 0.1)$	–	–	84.45 ± 2.08	76.24 ± 6.35	85.21 ± 1.62	74.38 ± 5.37
	$(30.0, 0.05)$	–	–	84.45 ± 2.08	76.24 ± 6.35	83.61 ± 2.54	78.30 ± 5.57
	$(25.0, 0.5)$	–	–	85.62 ± 1.24	55.22 ± 5.03	87.53 ± 1.19	53.43 ± 3.23
	$(25.0, 0.25)$	–	–	85.62 ± 1.24	55.22 ± 5.03	87.82 ± 0.65	53.54 ± 1.39
	$(25.0, 0.1)$	–	–	85.62 ± 1.24	55.22 ± 5.03	87.21 ± 1.45	56.07 ± 5.01
	$(25.0, 0.05)$	–	–	85.62 ± 1.24	55.22 ± 5.03	85.73 ± 1.13	55.11 ± 5.03
	$(20.0, 0.5)$	–	–	85.51 ± 1.34	44.61 ± 2.23	87.58 ± 0.53	46.40 ± 1.90
	$(20.0, 0.25)$	–	–	85.51 ± 1.34	44.61 ± 2.23	88.01 ± 0.40	44.94 ± 1.51
	$(20.0, 0.1)$	–	–	85.51 ± 1.34	44.61 ± 2.23	87.78 ± 0.66	43.71 ± 1.59
	$(20.0, 0.05)$	–	–	85.51 ± 1.34	44.61 ± 2.23	87.50 ± 0.54	45.51 ± 1.48
	$(15.0, 0.5)$	–	–	85.19 ± 0.86	32.25 ± 5.01	87.39 ± 0.79	29.72 ± 3.10
	$(15.0, 0.25)$	–	–	85.19 ± 0.86	32.25 ± 5.01	86.92 ± 0.76	31.40 ± 4.34
	$(15.0, 0.1)$	–	–	85.19 ± 0.86	32.25 ± 5.01	86.67 ± 0.75	31.63 ± 3.76
	$(15.0, 0.05)$	–	–	85.19 ± 0.86	32.25 ± 5.01	86.48 ± 1.32	30.73 ± 2.98
	$(10.0, 0.5)$	–	–	83.92 ± 1.70	8.99 ± 2.20	86.37 ± 2.26	9.89 ± 1.66
	$(10.0, 0.25)$	–	–	83.92 ± 1.70	8.99 ± 2.20	87.50 ± 0.83	8.26 ± 1.52
	$(10.0, 0.1)$	–	–	83.92 ± 1.70	8.99 ± 2.20	84.82 ± 2.11	9.72 ± 3.01
	$(10.0, 0.05)$	–	–	83.92 ± 1.70	8.99 ± 2.20	85.65 ± 0.86	8.88 ± 1.99
	$(2.0, 0.5)$	80.21 ± 1.25	100.00 ± 0.00	–	–	–	–
	$(2.0, 0.25)$	80.21 ± 1.25	100.00 ± 0.00	–	–	–	–
	$(2.0, 0.1)$	80.21 ± 1.25	100.00 ± 0.00	–	–	–	–
	$(2.0, 0.05)$	80.21 ± 1.25	100.00 ± 0.00	–	–	–	–
	$(1.0, 0.5)$	83.88 ± 2.24	98.93 ± 0.63	–	–	–	–
	$(1.0, 0.25)$	83.88 ± 2.24	98.93 ± 0.63	–	–	–	–
	$(1.0, 0.1)$	83.88 ± 2.24	98.93 ± 0.63	–	–	–	–
	$(1.0, 0.05)$	83.88 ± 2.24	98.93 ± 0.63	–	–	–	–
	$(0.5, 0.5)$	87.55 ± 0.74	93.37 ± 0.98	–	–	–	–
	$(0.5, 0.25)$	87.55 ± 0.74	93.37 ± 0.98	–	–	–	–
	$(0.5, 0.1)$	87.55 ± 0.74	93.37 ± 0.98	–	–	–	–
	$(0.5, 0.05)$	87.55 ± 0.74	93.37 ± 0.98	–	–	–	–
	$(0.25, 0.5)$	87.64 ± 0.92	81.35 ± 3.42	–	–	–	–
	$(0.25, 0.25)$	87.64 ± 0.92	81.35 ± 3.42	–	–	–	–
	$(0.25, 0.1)$	87.64 ± 0.92	81.35 ± 3.42	–	–	–	–
	$(0.25, 0.05)$	87.64 ± 0.92	81.35 ± 3.42	–	–	–	–
	$(0.1, 0.5)$	87.63 ± 0.47	60.84 ± 4.70	–	–	–	–
	$(0.1, 0.25)$	87.63 ± 0.47	60.84 ± 4.70	–	–	–	–
	$(0.1, 0.1)$	87.63 ± 0.47	60.84 ± 4.70	–	–	–	–
	$(0.1, 0.05)$	87.63 ± 0.47	60.84 ± 4.70	–	–	–	–
	$(0.05, 0.5)$	86.01 ± 1.41	43.99 ± 7.25	–	–	–	–
	$(0.05, 0.25)$	86.01 ± 1.41	43.99 ± 7.25	–	–	–	–
	$(0.05, 0.1)$	86.01 ± 1.41	43.99 ± 7.25	–	–	–	–
	$(0.05, 0.05)$	86.01 ± 1.41	43.99 ± 7.25	–	–	–	–
	$(0.01, 0.5)$	85.02 ± 0.84	0.00 ± 0.00	–	–	–	–
	$(0.01, 0.25)$	85.02 ± 0.84	0.00 ± 0.00	–	–	–	–
	$(0.01, 0.1)$	85.02 ± 0.84	0.00 ± 0.00	–	–	–	–
	$(0.01, 0.05)$	85.02 ± 0.84	0.00 ± 0.00	–	–	–	–

Table 8. Training accuracy and percentage of spurious features removed for the Gina and Sylva datasets. A '–' means that the experiments with this regularization parameter choice was not run.

Dataset	Regularizer Coefficients (λ_m, λ_s)	Group Lasso		Local Lasso		LESS-VFL (ours)	
		Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed	Final Accuracy	Spurious Features Removed
Gina	$(2.0, 0.1)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	80.84 ± 0.00	100.00 ± 0.00
	$(2.0, 0.05)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.00 ± 1.15	100.00 ± 0.00
	$(2.0, 0.01)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.46 ± 1.25	100.00 ± 0.00
	$(2.0, 0.005)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.37 ± 1.34	100.00 ± 0.00
	$(1.0, 0.1)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	80.98 ± 0.00	100.00 ± 0.00
	$(1.0, 0.05)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.00 ± 1.15	100.00 ± 0.00
	$(1.0, 0.01)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.46 ± 1.25	100.00 ± 0.00
	$(1.0, 0.005)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.37 ± 1.34	100.00 ± 0.00
	$(0.5, 0.1)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	80.98 ± 0.00	100.00 ± 0.00
	$(0.5, 0.05)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.00 ± 1.15	100.00 ± 0.00
	$(0.5, 0.01)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.46 ± 1.25	100.00 ± 0.00
	$(0.5, 0.005)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.37 ± 1.34	100.00 ± 0.00
	$(0.25, 0.1)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	80.98 ± 0.00	99.59 ± 0.00
	$(0.25, 0.05)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.00 ± 1.15	99.79 ± 0.21
	$(0.25, 0.01)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.46 ± 1.25	100.00 ± 0.00
	$(0.25, 0.005)$	50.43 ± 0.00	100.00 ± 0.00	83.46 ± 1.10	100.00 ± 0.00	83.37 ± 1.34	100.00 ± 0.00
	$(0.1, 0.1)$	81.16 ± 1.12	82.33 ± 0.47	83.46 ± 1.10	100.00 ± 0.00	81.99 ± 0.00	81.20 ± 0.00
	$(0.1, 0.05)$	81.16 ± 1.12	82.33 ± 0.47	83.46 ± 1.10	100.00 ± 0.00	83.57 ± 0.86	99.38 ± 0.62
	$(0.1, 0.01)$	81.16 ± 1.12	82.33 ± 0.47	83.46 ± 1.10	100.00 ± 0.00	83.46 ± 1.25	100.00 ± 0.00
	$(0.1, 0.005)$	81.16 ± 1.12	82.33 ± 0.47	83.46 ± 1.10	100.00 ± 0.00	83.37 ± 1.34	100.00 ± 0.00
	$(0.05, 0.1)$	82.71 ± 0.24	54.75 ± 3.12	83.29 ± 0.84	100.00 ± 0.00	81.99 ± 0.00	43.18 ± 0.00
	$(0.05, 0.05)$	82.71 ± 0.24	54.75 ± 3.12	83.29 ± 0.84	100.00 ± 0.00	83.93 ± 1.22	88.22 ± 5.17
	$(0.05, 0.01)$	82.71 ± 0.24	54.75 ± 3.12	83.29 ± 0.84	100.00 ± 0.00	83.54 ± 1.37	100.00 ± 0.00
	$(0.05, 0.005)$	82.71 ± 0.24	54.75 ± 3.12	83.29 ± 0.84	100.00 ± 0.00	83.34 ± 0.87	100.00 ± 0.00
	$(0.025, 0.1)$	–	–	84.03 ± 0.67	99.83 ± 0.15	81.99 ± 0.00	30.37 ± 0.00
	$(0.025, 0.05)$	–	–	84.03 ± 0.67	99.83 ± 0.15	83.57 ± 0.00	71.90 ± 0.00
	$(0.025, 0.01)$	–	–	84.03 ± 0.67	99.83 ± 0.15	80.40 ± 0.62	99.71 ± 0.31
	$(0.025, 0.005)$	–	–	84.03 ± 0.67	99.83 ± 0.15	84.15 ± 0.92	99.75 ± 0.15
	$(0.01, 0.1)$	80.40 ± 0.75	0.00 ± 0.00	81.12 ± 1.01	0.00 ± 0.00	81.99 ± 0.00	0.00 ± 0.00
	$(0.01, 0.05)$	80.40 ± 0.75	0.00 ± 0.00	81.12 ± 1.01	0.00 ± 0.00	79.47 ± 0.94	0.00 ± 0.00
	$(0.01, 0.01)$	80.40 ± 0.75	0.00 ± 0.00	81.12 ± 1.01	0.00 ± 0.00	81.38 ± 1.61	0.00 ± 0.00
	$(0.01, 0.005)$	80.40 ± 0.75	0.00 ± 0.00	81.12 ± 1.01	0.00 ± 0.00	80.03 ± 1.89	0.00 ± 0.00
Sylva	$(2.0, 0.01)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.55 ± 0.18	100.00 ± 0.00
	$(2.0, 0.005)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.58 ± 0.23	100.00 ± 0.00
	$(1.0, 0.01)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.55 ± 0.18	100.00 ± 0.00
	$(1.0, 0.005)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.58 ± 0.23	100.00 ± 0.00
	$(0.5, 0.01)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.55 ± 0.18	100.00 ± 0.00
	$(0.5, 0.005)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.58 ± 0.23	100.00 ± 0.00
	$(0.25, 0.01)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.55 ± 0.18	100.00 ± 0.00
	$(0.25, 0.005)$	93.30 ± 0.00	100.00 ± 0.00	97.60 ± 0.28	100.00 ± 0.00	97.58 ± 0.23	100.00 ± 0.00
	$(0.1, 0.01)$	93.30 ± 0.00	100.00 ± 0.00	98.54 ± 0.06	100.00 ± 0.00	98.61 ± 0.06	100.00 ± 0.00
	$(0.1, 0.005)$	93.30 ± 0.00	100.00 ± 0.00	98.54 ± 0.06	100.00 ± 0.00	98.55 ± 0.04	100.00 ± 0.00
	$(0.05, 0.01)$	98.52 ± 0.08	23.89 ± 4.32	98.56 ± 0.07	11.11 ± 2.03	98.53 ± 0.09	10.00 ± 1.08
	$(0.05, 0.005)$	98.52 ± 0.08	23.89 ± 4.32	98.56 ± 0.07	11.11 ± 2.03	98.59 ± 0.11	10.56 ± 1.81
	$(0.01, 0.01)$	98.46 ± 0.10	0.00 ± 0.00	98.37 ± 0.11	0.00 ± 0.00	98.49 ± 0.09	0.00 ± 0.00
	$(0.01, 0.005)$	98.46 ± 0.10	0.00 ± 0.00	98.37 ± 0.11	0.00 ± 0.00	98.53 ± 0.07	0.00 ± 0.00