# RVFR: Robust Vertical Federated Learning via Feature Subspace Recovery

**Jing Liu**
UIUC
jil292@illinois.edu

**Chulin Xie**
UIUC
chulinx2@illinois.edu

**Krishnaram Kenthapadi**
Amazon AWS AI/ML
kenthk@amazon.com

**Oluwasanmi Koyejo**
UIUC
sanmi@illinois.edu

**Bo Li**
UIUC
lbo@illinois.edu

## Abstract

Vertical Federated Learning (VFL) is a distributed learning paradigm that allows multiple agents to jointly train a global model when each agent holds a different subset of features for the same sample(s). VFL is known to be vulnerable to backdoor attacks during training, and also vulnerable to test-time attacks. However, unlike the standard horizontal federated learning, improving the robustness of VFL remains challenging. To this end, we propose RVFR, a novel robust VFL training and inference framework. The key to our approach is to ensure that with a low-rank feature subspace, a small number of attacked samples, and other mild assumptions, RVFR recovers the underlying uncorrupted features with guarantees, thus sanitizes the model against a vast range of backdoor attacks. Further, RVFR also defends against inference-time adversarial feature attack. Our empirical studies further corroborate the robustness of the proposed framework.

## 1 Introduction

Federated Learning (FL) [13, 16, 5] has achieved great progresses recently, where a central server coordinates with multiple agents to collaboratively train a machine learning (ML) model and each agent keeps its own training data due to privacy concerns. Two broad categories [24] of FL are Horizontal FL (HFL) and Vertical FL (VFL). In HFL, every agent has a different training set; while in VFL [14, 7, 15], different agents hold different parts of features for the same set of training data.

Given the distributed nature, FL raises new security concerns as the server has less control on the training data. A variety of attacks have been conducted on HFL and VFL. There have been considerable studies to robustify HFL, where each agent sends the model updates based on local data to the server and server aggregates these updates. Robust aggregation protocols have been proposed as defenses in HFL against malicious attacks [25, 11, 4, 9, 17, 10, 8, 22]. However, it is challenging to defend against malicious attacks in VFL, as there is no clear redundancy among the agents. In fact, there are few studies on robustness exploration for VFL.

In this paper, we propose a robust VFL training and inference framework via features subspace recovery (RVFR), which is able to defend against many types of attacks during both training and inference. In particular, during **training**, we propose to train each agent's feature extractor separately based on feedback from the server (Quarantine training stage); the server then performs robust feature subspace recovery given the embedded features provided by different agents (Robust feature subspace recovery stage); and the server further purifies the features based on the assumption that the fraction of malicious agents is relatively small (Feature purifying stage). Finally, the server trains its global

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

model based on the purified features (Server training stage). An overview of the training process is illustrated in Fig. 1. During the **inference** time, the server first purifies the embedded features provided by the agents and then feeds it to the trained global model for prediction. Build upon our framework, we aim to answer the following questions: *Is it possible to train a clean model given poisoned features? Are we able to make accurate predictions based on adversarially corrupted features? If yes, how many malicious agents and poisoned instances can we tolerate?*

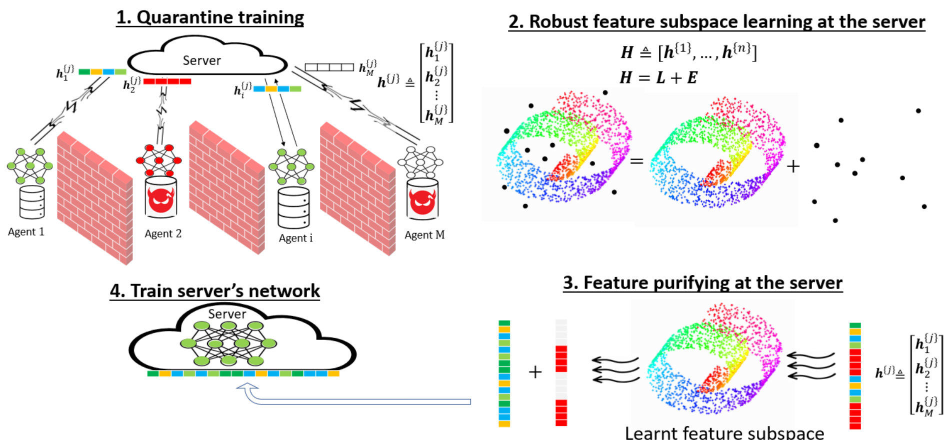
1. Quarantine training
Server
$h_1^{(j)}$
$h_2^{(j)}$
$h_i^{(j)}$
$h_M^{(j)}$
$h^{(j)} \triangleq \begin{bmatrix} h_1^{(j)} \\ h_2^{(j)} \\ \vdots \\ h_M^{(j)} \end{bmatrix}$
Agent 1
Agent 2
Agent i
Agent M
2. Robust feature subspace learning at the server
$H \triangleq [h^{(1)}, \dots, h^{(n)}]$
$H = L + E$
=
+
4. Train server's network
Server
3. Feature purifying at the server
+
Learnt feature subspace
$h^{(j)} \triangleq \begin{bmatrix} h_1^{(j)} \\ h_2^{(j)} \\ \vdots \\ h_M^{(j)} \end{bmatrix}$

Figure 1: Overview of RVFR training procedure.

In the **robust feature subspace learning stage**, we propose a variant of the Robust AutoEncoder [26] to recover the feature subspace. While the existing Robust AutoEncoder has no theoretical justifications, we provide theoretical support for the proposed one by showing that when the underlying feature subspace is linear (and using linear activation functions), it can exactly recover the linear feature subspace in the presence of corruption/poisoning. In the **feature purifying stage** (in both training and inference), we propose a novel AutoEncoder-based robust decomposition method that decomposes the potentially corrupted feature vector into two parts: one that lies on the learnt feature subspace (can be non-linear); and the other contains a block-sparse structure. We theoretically show that under certain conditions, the proposed method can recover the underlying feature vector *exactly* despite corruption from malicious agents.

We take a first step towards providing certifiable robust VFL with following key **contributions:**

- We propose a novel robust training procedure to defend against backdoor attacks in VFL, and prove that under mild assumptions it can exactly recover the underlying uncorrupted features. To the best of our knowledge, this is the first defense for VFL against training attacks with theoretical guarantees.
- We provide the first theoretical justification for the Robust AutoEncoder, which may be of independent interest.
- We propose a robust VFL inference procedure to defend against adversarial attacks. To our knowledge, this is the first defense for VFL against inference-time attacks with guarantees.

## 2 Related Work

In this section, we briefly review some prior works on backdoor attacks and defenses in FL.

**Backdoor attack and defense in Horizontal Federated Learning.** Many recent works have demonstrated the vulnerability of Horizontal FL to backdoor attacks [3, 2, 19, 21]. For example, [3, 2] show that training strong poisoned local models and submitting malicious model updates to the central server can mislead the global model. While [21] exploits the distributed nature of HFL and propose a distributed backdoor attack.

To defend against backdoor attacks in HFL, several robust federated protocols are proposed to mitigate the attacks during training. For example, [18] shows that clipping the norm of model updates and adding Gaussian noise can mitigate backdoor attacks that are based on the model replacement paradigm. [1] incorporates an additional validation phase to each round of FL to detect backdoor.

However, none of these provides certifiable robustness guarantees. Though there are also many robust aggregation methods for HFL [25, 11, 4, 9, 17, 10, 8, 22], the robustness guarantees provided are not for backdoor attack (*i.e.*, they can not guarantee the predicted label for a testing sample is not affected by malicious agents). Recently, [6] propose Ensemble FL to defend against backdoor attacks with certifiable robustness. However, the proposed majority voting strategy requires training hundreds of FL models. Finally, a recent work [20] which exploits model clipping and smoothing in HFL is able to provide certifiable robustness guarantee to backdoor attacks with limited magnitude.

**Backdoor attack and defense in Vertical Federated Learning.** Backdoor attack in VFL is challenging since in the typical setting [7] the agent does not have the label information, which means it does not even know which instance belongs to the target class. The authors in [15] assumes the malicious agent knows at least one training instance belongs to the target class of the backdoor attack. Under this assumption, they use the gradient-replacement method to perform the backdoor attack, *i.e.*, set the intermediate gradients of the poisoned instances to be that of target class's instance. They want the malicious agent to map the features with backdoor trigger to the same space as that of the features from target class. They empirically demonstrate the effectiveness of the proposed backdoor attack. In the end, they also propose empirical ways to defend against such backdoor attack, *e.g.*, by sparsifying the intermediate gradient before sending to the agent or adding noise to it, all without guarantees.

## 3 Preliminaries

**Vertical Federated Learning.** We first describe the basic setup of a typical VFL framework [7, 15]. There are $M$ agents and a server collaboratively training a ML model based on a set of $n$ training data $\mathcal{D} \triangleq \{\boldsymbol{x}_1^{\{j\}}, ..., \boldsymbol{x}_M^{\{j\}}, y^{\{j\}}\}_{j=1}^n$. The server (e.g., a third party) holds the label $y^{\{j\}}$, while agent $i$ holds partial feature $\boldsymbol{x}_i^{\{j\}}$ of $\boldsymbol{x}^{\{j\}} \triangleq [\boldsymbol{x}_1^{\{j\}}; ...; \boldsymbol{x}_M^{\{j\}}]$. Due to privacy concerns, the raw agent data $\boldsymbol{x}_i^{\{j\}}$ are not shared with other agents and the server. Instead, each agent $i$ learns a local embedding $g_i$ parameterized by $\theta_i$ that maps the original feature vector $\boldsymbol{x}_i^{\{j\}}$ to an embedded feature vector $\boldsymbol{h}_i^{\{j\}} \triangleq g_i(\boldsymbol{x}_i^{\{j\}}; \theta_i)$. The dimension of $\boldsymbol{h}_i^{\{j\}}$ is usually less than $\boldsymbol{x}_i^{\{j\}}$. The server only has access to the embedded features from the agents. The learning objective of VFL is to minimize the following:

$$L(\mathcal{D}; \theta_0, \theta_1, ..., \theta_M) = \frac{1}{n}\sum_{j=1}^{n} \ell(\hat{y}^{\{j\}}, y^{\{j\}}) + \sum_{i=1}^{M} \gamma(\theta_i) \tag{1}$$

where $\hat{y}^{\{j\}} = f_{\theta_0}(\boldsymbol{h}_1^{\{j\}}, ..., \boldsymbol{h}_M^{\{j\}}) \triangleq f_{\theta_0}\left(g_1(\boldsymbol{x}_1^{\{j\}}; \theta_1), ..., g_M(\boldsymbol{x}_M^{\{j\}}; \theta_M)\right)$, and $f_{\theta_0}$ is the global model kept at and learnt by the server. $\ell$ is the loss function and $\gamma$ is the regularizer that confines the complexity of, or encodes the prior knowledge about the local model parameters.

**Robust Subspace Recovery.** Robust subspace recovery aims to robustly recover the underlying subspace of the data despite that some data points are arbitrarily corrupted (*i.e.*, outliers). Mathematically, one observes a data matrix $\boldsymbol{H}$, where each column corresponds to a data point. We can decompose $\boldsymbol{H}$ as $\boldsymbol{L} + \boldsymbol{E}$, where $\boldsymbol{L}$ is the underlying uncorrupted data matrix, and the matrix $\boldsymbol{E}$ models the outlier corruptions. Since the fraction of the outliers is usually small, most columns of $\boldsymbol{E}$ are zero, *i.e.*, $\boldsymbol{E}$ is column-sparse.

Assume the underlying uncorrupted data points lie on a low-dimensional linear subspace, one of the notable Robust PCA work [23] showed that it is possible to recover the column-space of $\boldsymbol{L}$ exactly by solving the following convex program:

$$\min_{\boldsymbol{L}, \boldsymbol{E}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{E}^T\|_{2,1} \quad s.t.\ \boldsymbol{H} = \boldsymbol{L} + \boldsymbol{E} \tag{2}$$

Later we will see that in our proposed RVFR framework, $\boldsymbol{H} = \left[\boldsymbol{h}^{\{1\}}, ..., \boldsymbol{h}^{\{n\}}\right]$, where the $j$-th column of $\boldsymbol{H}$ corresponds to the whole embedded features for the $j$-th instance. In the backdoor attack, a few columns of $\boldsymbol{H}$ are poisoned. In general the low-dimensional manifold that the embedded features $\boldsymbol{L}$ lie on is not linear, and we use Robust AutoEncoder instead of Robust PCA to capture this non-linearity. We will show that Robust PCA can be viewed as a linearized version of the proposed Robust AutoEncoder, and we will provide theoretical justification for it.

We first introduce some definitions which are needed to formally state the above Robust PCA result.

**Definition 1.** **(Incoherence parameter [23])** A matrix $\boldsymbol{L} \in \mathbb{R}^{d\times n}$ with thin SVD $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, and $(1-\beta)n$ of whose columns are non-zero, is said to be column-incoherent with parameter $\mu$ if

$$\max_i \|V^T \boldsymbol{e}_i\|_2^2 \leq \frac{\mu r}{(1-\beta)n}, \tag{3}$$

where $\{\boldsymbol{e}_i\}$ are the coordinate unit vectors, $r$ is the rank of matrix $\boldsymbol{L}$.

**Remark 1.** A small incoherence parameter implies the right singular vectors of $\boldsymbol{L}$ are not 'spiky'. As mentioned in [23], if the left hand side of Eq. 3 is as big as 1, it essentially means that one of the directions of the column space which we wish to recover, is defined by only a single observation. Given the regime of a constant fraction of arbitrarily chosen and arbitrarily corrupted points, such a setting is highly undesirable.

With these definitions, we introduce the results from [23]:

**Theorem 1.** **(Exact subspace recovery [23])** Suppose we observe $\boldsymbol{H} = \boldsymbol{L}^* + \boldsymbol{E}^*$, where $\boldsymbol{L}^*$ has rank $r$ and incoherence parameter $\mu$. Suppose further that $\boldsymbol{E}^*$ is supported on at most $\beta n$ columns. Any output of Eq. 2 recovers the column space of $\boldsymbol{L}^*$ exactly, as long as the fraction of corrupted columns, $\beta$, satisfies $\frac{\beta}{1-\beta} \leq \frac{c_1}{\mu r}$, where $c_1 = 9/121$. This can be achieved by setting the parameter $\lambda$ in Eq. 2 to be $\frac{3}{7\sqrt{\beta n}}$ (in fact it holds for any $\lambda$ in a certain range).

Inspired by the Robust PCA properties, we propose a Robust AutoEncoder, which can capture the more general **non-linear** manifold. So far there is no theoretical guarantees for Robust AutoEncoders. The proposed Robust AutoEncoder is slightly different from an existing one [26], and we theoretically justify the proposed one by showing that when the underlying feature subspace is linear, after transformation based on the proposed Robust AutoEncoder, it exactly recovers the linear feature subspace as Robust PCA. Note that our established connection between the proposed Robust AutoEncoder and Robust PCA is non-trivial, especially due to the proposed $\ell_{2,1}$-norm regularizer on the latent layer.

Note that the exact recovery of the feature subspace is not enough for our goal of robust VFL, and we need to recover the underlying $\boldsymbol{L}^*$. We further propose a robust decomposition technique to recover $\boldsymbol{L}^*$ exactly by utilizing our learnt feature subspace based on the assumption that the fraction of malicious agents is small. More details can be found in Theorem 3.

## 4 Robust VFL via Feature Subspace Recovery (RVFR)

We first describe the threat models during VFL training and inference. Then we present both the training and inference procedures of the proposed robust VFL framework.

### 4.1 Threat Model

There are $M$ agents which hold different features of the same instances, and the label $y$ is held by the server. The attacker knows the feature information held by every agent and the label information on the server, but it can only choose and control at most $\alpha M$ agents for a malicious attack.

- **Training time threat model:** There are some small $\alpha$ fraction of agents that are malicious and perform backdoor attack during training, *i.e.*, their goal is to achieve high accuracy on both the original main task and the targeted backdoor task, where the targeted backdoor task is to assign an attacker-chosen target label to input data with a specific pattern (*i.e.*, trigger). Their trained feature extractors behave normally on non-poisoned instances , so as to maintain the main task accuracy. However, on the instances with backdoor trigger, their backdoor trained feature extractors map to a poisoned embedded feature (*e.g.*, the embedded feature space of the instances belonging to the target class). In the training set, the fraction $\beta$ of the instances with a backdoor trigger is small in each malicious agent.
- **Inference time threat model:** At inference time, the goal of the $\alpha M$ malicious agents is to output the adversarial embedded features $\boldsymbol{h}_{adv}$ for the instances with backdoor trigger, such that the final predicted label is close to the adversarial target label $y^A$:

$$\min_{\boldsymbol{h}_{adv}} \ell(f_{\theta_0}(\{\boldsymbol{h}_{adv}, \boldsymbol{h}_{benign}\}), y^A), \tag{4}$$

where $\boldsymbol{h}_{benign}$ denotes the embedded features provided by the $(1-\alpha)M$ benign agents, and $\boldsymbol{h}_{adv}$ denotes the embedded features provided by the malicious agents.

## 4.2 Training Procedure of RVFR

The proposed training procedure has four stages: Quarantine training, Robust feature subspace learning, Feature purifying, and Server training.

Stage 1 (**Quarantined training**): In VFL, the server trains the combined ML model, and each agent also trains its local feature extractor. A strong adversary could interfere with the server training and further propagate errors to benign agents. So we propose that the server connects to each agent separately to train the corresponding feature extractor $g_i$ parameterized by $\theta_i, i = 1, ..., M$. The training can be based on gradient descent, and is a special case of [7, Algorithm 1] with only one agent.

In quarantine training, the agent $i$ sends $\boldsymbol{h}_i^{\{j\}} \triangleq g_i(\boldsymbol{x}_i^{\{j\}}; \theta_i)$ to the server. The server uses it to train/update a temporary global model (whose input dimension equals that of $\boldsymbol{h}_i^{\{j\}}$) on the server side and also calculates $\frac{\partial \ell}{\partial \boldsymbol{h}_i^{\{j\}}}$ and sends it back to the agent $i$ for gradient updating of $\theta_i$ at agent side. After a number of such iterations between the server and the agent $i$, the quarantine training with agent $i$ finishes and the server records the final embedded feature $\boldsymbol{h}_i^{\{j\}}$ for every instance $\boldsymbol{x}_i^{\{j\}}, j = 1, ..., n$.

After the quarantine training with every agent, for each data point (instance) $\boldsymbol{x}^{\{j\}}$, the server has recorded $\{\boldsymbol{h}_1^{\{j\}}, \boldsymbol{h}_2^{\{j\}}, ..., \boldsymbol{h}_M^{\{j\}}\}$ from all the $M$ agents, concatenates them into a long column vector, denoted as $\boldsymbol{h}^{\{j\}}$. Let $\boldsymbol{H} = \left[\boldsymbol{h}^{\{1\}}, ..., \boldsymbol{h}^{\{n\}}\right]$, where the $j$-th column of $\boldsymbol{H}$ corresponds to the concatenated embeded features for the $j$-th instance.

Stage 2 (**Robust feature subspace learning**): We still need to protect the server's model against malicious agents. Fortunately, in our considered training time backdoor attack threat model, malicious agents' feature extractors behave normally on non-backdoored instances (to maintain the main task accuracy). The fraction $\beta$ of the backdoored training instances is small. This provides us the redundancy across the instances for robustifying the server's model. In this stage, the server disconnects from all the agents, and uses $\boldsymbol{H}$ to train a Robust AutoEncoder by minimizing the following objective:

$$\min_{\boldsymbol{L}, \boldsymbol{E}, \phi, \psi} \|\mathcal{E}_\psi(\boldsymbol{L})\|_{2,1} + \lambda \|\boldsymbol{E}^T\|_{2,1} \quad s.t.\ \boldsymbol{H} = \boldsymbol{L} + \boldsymbol{E}, \quad \boldsymbol{L} = \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{L})) \tag{5}$$

where $\boldsymbol{L}$ models the underlying true embeded features, and $\boldsymbol{E}$ models the outlier corruptions due to the backdoor attack (each column of $\boldsymbol{E}$ corresponds to an instance). In Eq. 5, we only enforce $\boldsymbol{E}$ to be column-sparse. $\mathcal{D}_\phi$ is the decoder parameterized by $\phi$, $\mathcal{E}_\psi$ is the encoder parameterized by $\psi$.

Note that Eq. 5 is equivalent to following:

$$\min_{\boldsymbol{L}, \phi, \psi} \|\mathcal{E}_\psi(\boldsymbol{L})\|_{2,1} + \lambda \|(\boldsymbol{H} - \boldsymbol{L})^T\|_{2,1} \quad s.t.\ \boldsymbol{L} = \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{L})) \tag{6}$$

In practice, we further relax the constraint $\boldsymbol{L} = \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{L}))$ and solve the following 'noisy' version instead:

$$\min_{\boldsymbol{L}, \phi, \psi} \|\mathcal{E}_\psi(\boldsymbol{L})\|_{2,1} + \lambda \|(\boldsymbol{H} - \boldsymbol{L})^T\|_{2,1} + \beta \|\boldsymbol{L} - \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{L}))\|_F^2 \tag{7}$$

To optimize this, one can alternate between updating $\boldsymbol{L}$ and updating paramters $\phi$ and $\psi$ of the AutoEncoder via gradient descent.

Stage 3 (**Feature purifying based on the recovered feature subspace**): Recovering the feature subspace is not enough, as we want to recover the original feature to train the server. Fortunately, there is another important prior information that we can and should use: the fraction $\alpha$ of the malicious agents is small. For each training instance $\boldsymbol{h}^{\{j\}}$, we use the learned AutoEncoder to decompose it as $\boldsymbol{h}^{\{j\}} = \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}^{\{j\}})) + \boldsymbol{e}^{\{j\}}$, where $\boldsymbol{e}^{\{j\}}$ models the block-sparse corruptions (each block corresponds to an agent). Ideally we want to solve the following:

$$\min_{\boldsymbol{l}, \boldsymbol{e}} \sum_{i=1}^{M} \mathbb{1}\{\boldsymbol{e}_i \neq \boldsymbol{0}\} \quad s.t. \quad \boldsymbol{h} = \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l})) + \boldsymbol{e} \tag{8}$$

which is equivalent to:

$$\min_{\boldsymbol{l}} \sum_{i=1}^{M} \mathbb{1}\{[\boldsymbol{h} - \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}))]_i \neq \boldsymbol{0}\} \tag{9}$$

However, the above block-sparsity minimization objective is NP-hard to solve, we instead relax it to the $\ell_{2,1}$-norm:

$$\min_{\boldsymbol{l}} \sum_{i=1}^{M} \|[\boldsymbol{h} - \mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}))]_i\|_2 \tag{10}$$

Our theoretical analysis in next section shows that under the outlined conditions, solving the above relaxed objective is still able to recover the underlying true embedded feature.

Stage 4 (**Server training on purified feature**): The server trains its own ML model (*e.g.*, neural network) parameterized by $\theta_0$ based on the above purified features $\mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}))$ and the corresponding labels.

### 4.3 Inference Procedure of RVFR

During inference, the server receives embedded features of a test instance from all the agents. It first purifies the overall embedded features using Stage 3 of the training procedure. Then it feeds the purified feature to its trained ML model for prediction.

## 5 Theoretical Analysis of RVFR

We first analyze the proposed Robust AutoEncoder, which is the backbone of the proposed defense framework. Then, we prove that the proposed feature purifying method can recover the underlying features exactly under mild conditions, which means that the server's global model can be both *trained and tested on correct features*, and thus is robust against diverse types of attacks.

The Robust AutoEncoder can be viewed as the generalization of Robust PCA, where the low-dimensional linear subspace is extended to the more general low-dimensional manifold. So far, there is still no guarantee for the exact recovery of the non-linear manifold in the presence of outlier corruptions. However, in the following we will show that when the underlying feature subspace is linear, the Robust AutoEncoder with linear activation functions can exactly recover that subspace in the presence of corruptions.

**Theorem 2. (Exact subspace recovery)** Assume $\boldsymbol{l}^{*\{i\}} \in \mathbb{R}^d, i = 1, ..., n$ lie on a low-dimension subspace, *i.e.*, rank$(\boldsymbol{L}^*) = r \ll d$. For the AutoEncoder, assume there are no non-linear activation functions nor bias terms, while setting the dimension of the code layer of AutoEncoder to be $r =$ rank$(\boldsymbol{L}^*)$ and restricting the weight matrices to be orthonormal. Then under the conditions assumed in Theorem 1, the global optimal solution of Eq. 5 is equivalent to the solution of Eq. 2, and the corresponding weight matrix of the Encoder is guaranteed to recover the underlying subspace of $\boldsymbol{L}^*$ exactly.

**Remark 2.** The conditions required in Theorem 2 provide some useful insights. First, the fraction $\beta$ of poisoned instances needs to be small. Second, the dimension of the underlying feature subspace also needs to be small. If the dimension of the feature subspace is too large, it is difficult to distinguish corrupted features from the uncorrupted ones. On the other hand, low dimension implies that there is enough redundancy among the features computed by the agents. This is preferred since redundancy provides more robustness. Third, as discussed in Remark 1 regarding the incoherence condition, preferring small incoherence parameter implies that we don't want any dimension of the feature subspace to be defined by very few data points. Otherwise it's very risky if those data points are poisoned, as it becomes impossible to recover that dimension of the feature subspace. In other words, we need enough redundancy among the training instances to be robust to the adversarial corruptions.

Note that exact recovery of the subspace does not mean exact recovery of $\boldsymbol{L}$. For the corrupted/poisoned instances, usually it is impossible to restore them as the corruptions can be arbitrary. Fortunately, once we have recovered the underlying feature subspace, it is possible to recover the original features exactly by utilizing the fact that the fraction of malicious agents is small. This holds for both training (*i.e.*, used in feature purifying stage) and inference time. The following theorem shows that we can recover the underlying embedded features exactly in both training and inference.

**Theorem 3. (Exact feature recovery in training and inference)** Let $\boldsymbol{W}_{end}$ be the weight matrix of the last layer of the AutoEncoder. Assume there are no bias term nor non-linear activation function in the *last layer* of the AutoEncoder. Assume the trained AutoEncoder captures the underlying feature subspace (*i.e.*, $\mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l})) = \boldsymbol{l}$ for any uncorrupted feature vector $\boldsymbol{l}$). If $\forall \boldsymbol{v} \in Range(\boldsymbol{W}_{end})\backslash 0$, for any partition $\{S, \bar{S}\}$ of $M$ agent-wise blocks with $|S| = g > M/2$, it holds that $\sum_{i \in S} \|\boldsymbol{v}_i\|_2 > \sum_{i \in \bar{S}} \|\boldsymbol{v}_i\|_2$, then for any $\boldsymbol{h} = \boldsymbol{l}^* + \boldsymbol{e}^*$ with $\mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}^*)) = \boldsymbol{l}^*$ and $\sum_{i=1}^{M} \mathbb{1}\{\boldsymbol{e}_i^* = \boldsymbol{0}\} \geq g$, $\mathcal{D}_\phi(\mathcal{E}_\psi(\boldsymbol{l}^*))$ is the unique solution of Eq. 10 and Eq. 9.

Note that in Theorem 3, all layers except the last layer of AutoEncoder can be non-linear.

In summary, Theorem 2 shows that under certain conditions (*e.g.*, the fraction of poisoning instances $\beta$ is small), we can recover the underlying feature subspace exactly. Further, Theorem 3 shows that under certain conditions (*i.e.*, the fraction of malicious agents $\alpha$ is small and the AutoEncoder we have learnt captures the underlying feature subspace), we can recover the underlying embedded features exactly during both training and inference. Then the server's ML model is trained and tested on the corrected features.

## 6 Empirical Studies

Our experiment setting is similar to the VFL backdoor attack work [15], except that we additionally studied the 4-agents setting besides the original 2-agents setting. More specifically, on the NUS-WIDE dataset, each sample has 634 image features and 1000 text features. When there are 2 agents, one agent holds the image features while the other agent holds the text features. When there are 4 agents, Agent 1 holds first 225 dimension image features, Agent 2 holds the rest of the image features, while Agent 3 and Agent 4 each holds 500 dimension text features. In both settings, the last agent is malicious and performs the backdoor attacks. The backdoor attack goal is to change the predicted label of the instance with a backdoor trigger to be a specific target class (*i.e.*, 'buildings'), where the backdoor trigger is 'the last text feature equals 1'. The detailed backdoor attack method is described in [15, Algorithm 2]. During the inference, the malicious agent uses its backdoor feature extractor to provide adversarial embedded features. The dimension of the output layer of the feature extraction neural network at each agent is set to be 32 as in [15]. ReLU is used as the activation function for all the neural networks except for the output layer.

**Comparisons:** We compare our proposed defense method with several state-of-the-art VFL defense techniques, *e.g.*, *Gradient Sparsification* method by sparsifying the intermediate gradients sent from the server to agents, and *Differential Privacy* method by adding noise to such exchanged gradients [15]. The original VFL framework without defense [15, Algorithm 1] is viewed as baseline. The main task accuracy and backdoor accuracy during inference time for each method are reported in Table 1. First, notice that for the baseline VFL method, when adding more benign agents, the backdoor attack accuracy drops. The Gradient Sparsification and Differential Privacy methods do not have any theoretical guarantee and do not defend against the backdoor attack very well. The proposed RVFR is much more robust against backdoor attacks in both settings.

Table 1: Inference Time Main Task Accuracy & Backdoor Accuracy of Different VFL Methods

| Setting | Metrics | Baseline | Laplacian Noise | Gradient Sparsification | Proposed RVFR |
|---|---|---|---|---|---|
| 4 Agents | Main Task Acc | 86.4% | 85.3% | 86.2% | 83.5% |
| | Backdoor Acc | 49.3% | 31.7% | 12.7% | 3.6% |
| 2 Agents | Main Task Acc | 86.6% | 82.4 % | 84.4 % | 86.2% |
| | Backdoor Acc | 99.3% | 35.3 % | 23.5 % | 9.5 % |

## 7 Conclusions

In this work, we proposed a novel robust feature subspace recovery based VFL framework to defend against backdoor attacks during training, and adversarial feature attacks during inference, both with theoretical guarantees. An important byproduct of our analysis is the first theoretical justification for the Robust AutoEncoder, which may be of independent interest. Experiments on NUS-WIDE dataset further verify the robustness of the proposed framework.

## Acknowledgments and Disclosure of Funding

Li acknowledges partial funding from the NSF grant No.1910100, NSF CNS 20-46726 CAR, and the Amazon Research Award.

Koyejo acknowledges partial funding from a C3.ai Digital Transformation Institute Award, a Jump Arches Award, NSF III 2046795 and IIS 1909577.

## References

[1] Sebastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. Baffle: Backdoor detection via feedback-based federated learning. *arXiv preprint arXiv:2011.02167*, 2020.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[3] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[4] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 118–128, 2017.

[5] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[6] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Provably secure federated learning against malicious clients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6885–6893, 2021.

[7] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vafl: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.

[8] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.

[9] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. Attack-resistant federated learning with residual-based reweighting. *arXiv preprint arXiv:1912.11464*, 2019.

[10] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020)*, pages 301–316, 2020.

[11] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR, 2018.

[12] Vassilis Kekatos and Georgios B. Giannakis. From sparse signals to sparse residuals for robust sensing. *IEEE Transactions on Signal Processing*, 59(7):3355–3368, 2011.

[13] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[14] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient collaborative learning framework for distributed features. *arXiv preprint arXiv:1912.11187*, 2019.

[15] Yang Liu, Zhiqian Yi, and Tianjian Chen. Backdoor attacks and defenses in feature-partitioned collaborative learning. *ArXiv*, abs/2007.03608, 2020.

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[17] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[18] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[19] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *arXiv preprint arXiv:2007.05084*, 2020.

[20] Chulin Xie, Minghao Chen, Pin-Yu Chen, and Bo Li. Crfl: Certifiably robust federated learning against backdoor attacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 11372–11382. PMLR, 2021.

[21] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[22] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pages 6893–6901. PMLR, 2019.

[23] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. *IEEE Transactions on Information Theory*, 58(5):3047–3064, 2012.

[24] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[25] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[26] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.