
DP-Compressed VFL is secure for Model Inversion Attacks

Huzaifa Arif

Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY 12180

Stacy Patterson

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180

Abstract

Making secure distributed learning frameworks is a major challenge in machine learning research. For feature partitioned framework, previously, little study has been done to investigate attack models. In this work we show leakage attacks in distributed model; we also show that top-k sparsification makes the resulting model secure against inversion attacks while scalar quantization with a noisy compressed VFL framework is also secure for model inversion attacks regardless of the depth of the client model.

1 Introduction

In case where the data sets are partitioned across clients in such a way that no single party has all the features of a data sample we have the problem of vertical federated learning. Works in Federated Learning are primarily concerned with either reducing communication overhead or to make the model more secure. We tackle the latter problem in this report.

1.1 Related Work

The strongest form of leakage attack that has been studied in the federated learning literature is the deep leakage attack [7]. [5] were the first to study gradient leakage attack in the VFL framework. However, that leakage attack assumed exchange of gradients and model parameters. In distributed model no one party has knowledge of the entire model and the parameters are distributed. We investigate that even these frameworks can have leakage attacks if a honest but curious party has access to the target model and the embeddings.

1.2 Contributions

The contributions in this report can be summarised in three points:

- 1) It is shown that inversion is possible if access to **embeddings** and **client/server model** exists
- 2) It has been shown that **quantization/compression** of these embeddings reduces communication costs without affecting model accuracy by much [2]. I look at effect of compression in model inversion attacks.

3) A new **randomized compression** mechanism is introduced that makes a DP- model robust to model inversion attacks for shallow networks for scalar quantization.

2 Background and Problem Formulation

2.1 Vanilla VFL Algorithm

Like typical federated learning environments, VFL consists of a set of M parties $\{1, \dots, M\}$ and a server. The dataset $\mathbf{X} \in \mathbb{R}^{N \times D}$ is split across the M parties. N denotes the number of data samples and D is the number of features. Formally, we assume that for each sample x^i , a party m holds a disjoint subset of the features, denoted x_m^i , so that $x^i = [x_1^i, \dots, x_M^i]$ [2]. For each sample, there is a corresponding label y^i . In our cases, we assume that this label is held by each of the participating parties. Some important notations for reference later are defined in this section : $\mathbf{y} \in \mathbb{R}^{N \times 1}$ is the vector of all sample labels. $\mathbf{X}_m \in \mathbb{R}^{N \times D_m}$ are defined as the local dataset of a party m , where the i -th row correspond to data features x_m^i [2].

We also consider the distributed model of the VFL framework [3]. Each party m holds a set of model parameters θ_m as well as a local embedding function $h_m(\cdot)$. Unlike some of the previous frameworks [3] we also assume the server to be a participating party with the model parameter called θ_0 . The goal of the participating devices is to solve the following optimization problem :

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

where $\Theta = [\theta_0^T, \dots, \theta_M^T]^T$ is the *global model*. Thus the final objective is that all the distributed parties are able to train their local models including the server.

The embeddings for one sample is a vector with $h_m(\theta_m; x_m^i) \in \mathbb{R}^{P_m}$ for $m = 0, \dots, M$, where P_m is the size of the m -th embedding. We define $\nabla_m F(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{N} \sum_{i=1}^N \nabla_{\theta_m} l(\theta_0, h_1(\theta_1; x_1^i), \dots, h_M(\theta_M; x_M^i); y^i)$ to be the gradient of θ_m .

We also define important notions for considering different batch sizes. Let $\mathbf{X}^{\mathbf{B}}$ and $\mathbf{y}^{\mathbf{B}}$ be the set of samples and labels corresponding to a randomly sampled mini-batch \mathbf{B} of size B . Local party gradients are now defined as $\nabla_m F_{\mathbf{B}}(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{B} \sum_{x^i, y^i \in \mathbf{X}^{\mathbf{B}}, \mathbf{y}^{\mathbf{B}}} \nabla_{\theta_m} l(\theta_0, h_1(\theta_1; x_1^i), \dots, h_M(\theta_M; x_M^i); y)$. For a minibatch we also define a matrix of embeddings ,particularly, we let $h_m(\theta_m; \mathbf{X}_m^{\mathbf{B}}) := \{h_m(\theta_m; x_m^{\mathbf{B}^1}), \dots, h_m(\theta_m; x_m^{\mathbf{B}^B})\}$ be the set of embeddings for party m associated with mini-batch \mathbf{B} .

2.2 Differential Privacy

We start by defining the notion of differential privacy. Formally, given a set of data sets \mathcal{D} provided with a notion of neighboring data sets $\mathcal{N}_{\mathcal{D}} \subset \mathcal{D} \times \mathcal{D}$ and a query function $f: \mathcal{D} \rightarrow \mathcal{X}$, a mechanism $\mathcal{M}: \mathcal{X} \rightarrow \mathcal{O}$ to release the answer of the query, is defined to be (ϵ, δ) differentially private if for any measurable subset $S \subseteq \mathcal{O}$ and two neighboring data sets $(\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{N}_{\mathcal{D}}$

$$Pr(\mathcal{M}(f(\mathcal{D}_1)) \in S) \leq e^\epsilon Pr(\mathcal{M}(f(\mathcal{D}_2)) \in S) + \delta. \quad (2)$$

A simpler version of this definition is the ϵ differential privacy[?] and it is said that a randomised mechanism \mathcal{M} gives ϵ differential privacy if for all datasets \mathcal{D}_1 and \mathcal{D}_2 differing on at most by one element, and all $S \subseteq \text{Range}(\mathcal{M})$

$$\frac{Pr[\mathcal{D}_1]}{Pr[\mathcal{D}_2]} \leq e^\epsilon$$

2.2.1 Local Differential Privacy

In addition to the notion of DP we introduce another notion called the LDP that is particularly helpful and important in our analysis. The one major area where LDP differs from DP is that LDP does not require trust in any central party (for instance the server in the case of federated learning) as noise is added to the users (input -?) [?]

[?] Definition [?]: We say that an algorithm π satisfies ϵ local differential privacy where $\epsilon > 0$ if only if for any input v and v'

$$\forall y \in \text{Range}(\pi): \frac{\Pr[\pi(v) = y]}{\Pr[\pi(v') = y]} \leq \epsilon^\delta \quad (3)$$

3 Proposed Algorithm

We now propose a new variant of VFL that builds on [2] in that it effectively aims to reduce the communication costs between the parties and provide security for model inversion. A batch of size \mathbf{B} is randomly chosen and information is exchanged between parties so that local iterations can be performed. Each party then runs block-coordinate stochastic gradient descent on its local model for Q local iterations. Parties run Q local iterations and the algorithm runs for R global rounds thus running for $T = RQ$ total local iterations.

[2] defines a set of general compressors for compressing party embeddings and the server model: $\mathcal{C}_m(\cdot) : \mathbb{R}^{P_m} \rightarrow \mathbb{R}^{P_m}$ for $m = 0, \dots, M$. For the case that the compression scheme is scalar quantization we also add binomial noise to the compressed embeddings. Binomial noise from distribution $Z_m \sim \text{Bin}(N, p)$ is added to the compressed embeddings such that $Z_m \in \mathbb{R}^{P_m}$. To calculate the gradient for data sample x^i , party m receives $\mathcal{C}_j(h_j(\theta_j; x_j^i)) + Z_j$ from all parties $j \neq m$. With this information, a party m can compute $\nabla_m F_{\mathbf{B}}$ and update its parameters θ_m for multiple local iterations. Algorithm 1 details this procedure in more detail. Note, the loss function changes:

$$F(\Theta; \mathbf{X}; \mathbf{y}) := \frac{1}{N} \sum_{i=1}^N l(\theta_0, \mathcal{C}_1(h_1(\theta_1; x_1^i)) + Z_1, \dots, \mathcal{C}_M(h_M(\theta_M; x_M^i)) + Z_M; y^i) \quad (4)$$

Noisy compressed embeddings are shared after every global round to all the parties for gradient computation. These embeddings correspond to a batch and is therefore a matrix of embeddings vectors corresponding to each data sample of the minibatch for each client. Notice line(6) is matrix addition to each compressed embedding. An addition notation $\hat{\Phi}^{t_0}$ denotes the information received by each party for local gradient computation :

$$\hat{\Phi}^{t_0} := \{\mathcal{C}_0(\theta_0^{t_0}) + Z_0, \mathcal{C}_1(h_1(\theta_1^{t_0})) + Z_1, \dots, \mathcal{C}_M(h_M(\theta_M^{t_0})) + Z_M\}. \quad (5)$$

The notation $\hat{\Phi}_{-m}^{t_0} \leftarrow \hat{\Phi}^{t_0} - Np$ is used to explain the process that the mean of the distribution is subtracted from each of the received embeddings. To distinguish the received embeddings we introduce two additional notations. We let $\hat{\Phi}_{-m}^{t_0}$ be the set of compressed embeddings from parties $j \neq m$, and let $\hat{\Phi}_m^t := \{\hat{\Phi}_{-m}^{t_0}; h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}})\}$ to be the set of information required for local training.

4 Analysis

4.1 Convergence Analysis

Let vectors $\epsilon_m^{x^i}$ for $m = 0, \dots, M$, be the compression errors of $\mathcal{C}_m(\cdot)$ on a data sample x^i . The compression error for an embedding is denoted by $\epsilon_m^{x^i} := \mathcal{C}_m(h_m(\theta_m; x^i)) - h_m(\theta_m; x^i)$ for non-randomized compression while for randomized compression this error becomes $\tilde{\epsilon}_m^{x^i} := \mathcal{C}_m(h_m(\theta_m; x^i)) - h_m(\theta_m; x^i) + Z_i - Np$.

We assume that the variance of batch gradient descent is bounded so $\mathbb{E}_{\mathbf{B}} \|\nabla_m F(\Theta) - \nabla_m F_{\mathbf{B}}(\Theta)\|^2 \leq \frac{\sigma_m^2}{B}$

H_m and G_m denote the bounding hessian and bounded embedding gradients respectively. Let $\epsilon_m^{t_0}$ be the $P_m \times B$ matrix with $\epsilon_m^{x^i}$ for all data samples x^i in mini-batch \mathbf{B}^{t_0} as the columns. We denote the expected squared message compression error from party m at round t_0 as $\mathcal{E}_m^{t_0} := \mathbb{E} \|\epsilon_m^{t_0}\|_{\mathcal{F}}^2$.

Algorithm 1 Noisy Compressed Vertical Federated Learning

```
1: Initialize:  $\theta_m^0$  for all parties  $m$  and server model  $\theta_0^0$ 
2: for  $t \leftarrow 0, \dots, T-1$  do
3:   if  $t \bmod Q = 0$  then
4:     Randomly sample  $\mathbf{B}^t \in \{\mathbf{X}, \mathbf{y}\}$ 
5:     for  $m \leftarrow 1, \dots, M$  in parallel do
6:       Send  $\mathcal{C}_m(h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^t})) + Z_m$  to server
7:     end for
8:      $\hat{\Phi}^{t_0} \leftarrow \{\mathcal{C}_0(\theta_0) + Z_0, \mathcal{C}_1(h_1(\theta_1^t)) + Z_1, \dots, \mathcal{C}_M(h_M(\theta_M^t)) + Z_M\}$ 
9:     Server sends  $\Phi^{t_0}$  to all parties
10:  end if
11:  for  $m \leftarrow 0, \dots, M$  in parallel do
12:     $\hat{\Phi}_{-m}^{t_0} \leftarrow \hat{\Phi}_{-m}^{t_0} - Np$ 
13:     $\hat{\Phi}_m^t \leftarrow \{\hat{\Phi}_{-m}^{t_0}; h_m(\theta_m^t; \mathbf{X}_m^{\mathbf{B}^{t_0}})\}$ 
14:     $\theta_m^{t+1} \leftarrow \theta_m^t - \eta^{t_0} \nabla_m F_{\mathbf{B}}(\hat{\Phi}_m^t; \mathbf{y}^{\mathbf{B}^{t_0}})$ 
15:  end for
16: end for
```

The following theorem holds true for both randomized and non-randomized compression with the proof for non-randomized C-VFL available in [2]. We show here the convergence bound of noisy C-VFL.

Theorem 4.1. Convergence with fixed step size:

If $\eta^{t_0} = \eta$ for all iterations and satisfies $\eta^{t_0} \leq \frac{1}{16Q \max\{L, \max_m L_m\}}$, then the average squared gradient over R global rounds of Algorithm 1 is bounded by:

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

Proof:

Effectively Noisy C-VFL changes the compression error from C-VFL of each party m [2]. The new compression error is simply a function We show how the new compression error is an addition of variance of the noise matrix with the previous compression error. In the proof we show how the compression error has effectively changed of each party m

Before beginning the proof we define some important variables:

$\tilde{\mathcal{E}}_m^{t_0}$ denotes the compression error after noise addition

$\mathcal{E}_m^{t_0}$ denotes compression error without noise addition

$\tilde{Z}_m^{t_0} = Z_m^{t_0} - Np$ is a matrix of dimension $Pm \times B$

For the ease of the proof we assume $K = Pm \times B$.

$Z_m^{t_0}$ is a matrix of dimension $Pm \times B$ where each entry of the matrix is from a binomial distribution
: $Z_{m,i}^{t_0} \sim \text{Bin}(N, p)$

Proof Starts:

$$\mathcal{E}_m^{t_0} := \mathbb{E}_B \|\epsilon_m^{t_0}\|_{\mathcal{F}}^2.$$

$$\hat{\mathcal{E}}_m^{t_0} := \mathbb{E}_B \|\epsilon_m^{t_0} + \tilde{Z}_m^{t_0}\|_{\mathcal{F}}^2.$$

$$\hat{\mathcal{E}}_m^{t_0} := \sum_{i=1}^K \mathbb{E}_B [(\epsilon_{m,i}^{t_0} + \tilde{Z}_{m,i}^{t_0})^2]$$

$$\hat{\mathcal{E}}_m^{t_0} := \sum_{i=1}^K \mathbb{E}_B [(\epsilon_{m,i}^{t_0})^2 + 2(\tilde{Z}_{m,i}^{t_0})(\epsilon_{m,i}^{t_0}) + (\tilde{Z}_{m,i}^{t_0})^2]$$

$$\hat{\mathcal{E}}_m^{t_0} := \mathbb{E}_B [\sum_{i=1}^K (\epsilon_{m,i}^{t_0})^2] + 2(\sum_{i=1}^K \mathbb{E}_B [(\tilde{Z}_{m,i}^{t_0})(\epsilon_{m,i}^{t_0})]) + \sum_{i=1}^K (\tilde{Z}_{m,i}^{t_0})^2$$

$$\hat{\mathcal{E}}_m^{t_0} := \mathbb{E}_B \|\epsilon_m^{t_0}\|_{\mathcal{F}}^2 + \sum_{i=1}^K (\tilde{Z}_{m,i}^{t_0})^2 + 2(\sum_{i=1}^K \mathbb{E}_B [(\tilde{Z}_{m,i}^{t_0})(\epsilon_{m,i}^{t_0})])$$

Note that the term $\hat{\mathcal{E}}_m^{t_0}$ is still random with respect to the binomial noise. We now look at the expression in terms of expectation with respect to the noise term. That new error is the compression error $\tilde{\mathcal{E}}_m^{t_0}$

$$\tilde{\mathcal{E}}_m^{t_0} := \mathbb{E}_Z [\mathcal{E}_m^{t_0} + \sum_{i=1}^K (\tilde{Z}_{m,i}^{t_0})^2 + 2(\sum_{i=1}^K \mathbb{E}_B [(\tilde{Z}_{m,i}^{t_0})(\epsilon_{m,i}^{t_0})])]$$

$$\tilde{\mathcal{E}}_m^{t_0} := \mathcal{E}_m^{t_0} + \sum_{i=1}^K \mathbb{E}_Z [\sum_{i=1}^K (\tilde{Z}_{m,i}^{t_0})^2] + 2(\sum_{i=1}^K \mathbb{E}_Z [(\tilde{Z}_{m,i}^{t_0})] \mathbb{E}_B [(\epsilon_{m,i}^{t_0})])]$$

Since $\mathbb{E}_Z [(\tilde{Z}_{m,i}^{t_0})] = 0$ and $\mathbb{E}_Z [(\tilde{Z}_{m,i}^{t_0})^2] = \mathbb{E}_Z [(Z_{m,i}^{t_0} - Np)^2] = Np(1-p)$ and $K = Pm \times B$

Therefore,

$$\tilde{\mathcal{E}}_m^{t_0} := \mathcal{E}_m^{t_0} + (Pm)(B)(Np)(1-p)$$

Following the theoretical analysis from [2] we show that the error bound has changed and the algorithm converges with this bound of the error.

4.2 Communication Complexity

Noisy VFL has complexity

Theorem 4.2. $O(R \cdot M \cdot (B \cdot (\log_2(k) + \log_2(1 + N/k)) + \log_2 z))$. Thus if the variance of the noise and the quantization levels are of the same order there is only a minor increase in the noisy compressed VFL over the compressed VFL framework.

Proof Before Compression the Cost is : $O(R \cdot M \cdot (B \cdot P_m + |\theta_0|))$

Post Compression the Cost is :

$O(R \cdot M \cdot (B \cdot \log_2 k + \log_2 z))$ where k is the quantization level of the clients and z is the quantization level of the server model.

Post Noisy Compression the Cost is : $O(R \cdot M \cdot (B \cdot \log_2(k + N) + \log_2 z))$ where the Binomial noise is $\sim (N, p)$

A simple algebraic manipulation shows that : $O(R \cdot M \cdot (B \cdot \log_2(k) + \log_2(1 + N/k) + \log_2 z))$. Thus if the variance of the noise and the quantization levels are of the same order there is only a minor increase in the noisy compressed VFL over the compressed VFL framework.

4.3 Privacy Analysis

If $F(D) = \mathcal{C}_m(h_M(\theta_M; x_M^i)) + (Z_i - Np)$ the embeddings are (ϵ, δ) DP if their sensitivity is bounded [1]

$$\mathcal{M}_b^{N,p}(f(D)) \triangleq f(D) + (Z - Np) \quad (7)$$

Theorem 4.3. For any δ , parameter N and p with sensitivity bounds $\Delta_1, \Delta_2, \Delta_{\inf}$ such that

$$Np(1-p) \geq \max(23\log(10d/\delta), 2\Delta_{\inf}) \quad (8)$$

and



Figure 1: Depth of layer makes reconstruction hard

$$\epsilon(\Delta_1, \Delta_2, \Delta_{\text{inf}})(9)$$

$$\mathcal{M}_b^{N,p} \text{is}(\epsilon, \delta) \text{Differentially Private (DP)}$$

Applying the post processing inequality [4] we can show that the gradients are differentially private. The composition theorem can be extended to show that the resulting model is Differentially Private as well [4]

We notice that if the model is deep at each party then the recovery is strong in the case of model inversion attack. See Figure 1 for illustration. However parties may be constrained to support a less complex architecture and that makes the embeddings prone to full inversion attacks as shown by Fig1(a).

4.4 Model Inversion Attacks

Model inversion attacks in the case of distributed VFL frameworks have not been studied previously. In this report we look at model inversion from the perspective of the honest and curious party (HCP) that has access to the model and the received embeddings. The HCP aims to solve the following optimization problem [6]

$$\mathbf{x}_m^* = \arg\min_{\mathbf{x}_m \in \mathcal{R}^{H \times W \times C}} \mathcal{L}(\Phi(\mathbf{x}_m), h_m(\theta_m; x_m)) + \lambda \mathcal{R}(\mathbf{x}_m) \quad (10)$$

$$\mathcal{L}(\Phi(\mathbf{x}_m), h_m(\theta_m; x_m)) = \|\Phi(\mathbf{x}_m) - h_m(\theta_m; x_m)\|^2 \quad (11)$$

$$E(\mathbf{x}_m) = \mathcal{L}(\Phi(\mathbf{x}_m), h_m(\theta_m; x_m)) + \lambda \mathcal{R}(\mathbf{x}_m) \quad (12)$$

$$\mu_{t+1} \leftarrow m\mu_t - \eta_t \nabla E(\mathbf{x}_m) \quad (13)$$

$$\mathbf{x}_m^{t+1} \leftarrow \mathbf{x}_m^t + \mu^t \quad (14)$$

We notice that any party that has access to these embeddings performs model inversion according to the equation above. The regulariser used is the TV norm.

4.4.1 Caveat in the MIA for Noisy Compression

When evaluating MIA attacks in the case of HCP we assume full access to the model parameters of the attacked party. However, when noisy compressed VFL is performed we assume that the mean of the distribution is kept private and the attacker does not have access to it. Thus in our framework, the efficacy of the MIA is only for a third party that is not involved in the training process.

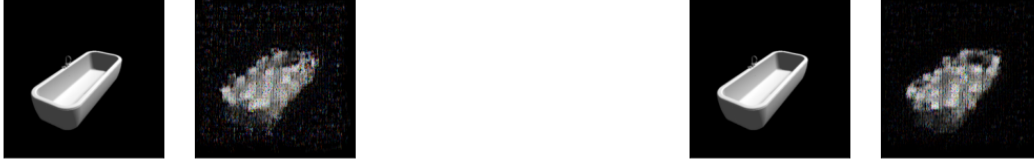
5 Experimental Results

Experiments were performed on the ModelNet-10 datasets with the assumption that each party holds very different set of convolutional and max pool layers. For the experimental work we assume batch size is 1, however increasing batch size would not decay the performance as there is no gradient addition problem addressed by [5]

5.1 Model Inversion Attacks

5.2 C-VFL

Sending compressed embeddings using Top-K sparsification and Scalar Quantization maybe one such solution. See Fig 2 and 3 for reference. Increasing compression ratios makes it harder for the MIA



(a) Scalar Quantization Level 2

(b) Scalar Quantization Level 8

Figure 2: Scalar Quantization



(a) Top-K : 0.12

(b) Top-K : 0.23

Figure 3: Top-K Sparsification

to fully recover the true class in the case of top-K sparsification. However, we notice that for scalar quantization the MIA attack is strong enough to recover the true class of the object even for lower quantization levels.

5.3 Noisy C-VFL

To address this concern we propose the Noisy Compressed VFL. Table 1 illustrates the performance of Noisy C-VFL in the case of model inversion attacks. The compression scheme used was scalar quantization. We can clearly see that for scalar quantization adding biased noise and subtracting the mean before gradient computation provides stronger defense for recovery in the MIA for the VFL setting.

Although we omit the details of the convergence analysis in this work, it is shown in [2] that if the compression error is bounded the algorithm converges. A straight forward extension of that work shows that for bounded variance of the noise the algorithm Noisy-CVFL also converges.

6 Conclusions and Future Work

In summary we have shown that C-VFL is secure for model inversion attacks for top-k sparsification. If the compression scheme was changed to scalar quantization we have proposed a novel Noisy version of the C-VFL that separates the process of biasing and unbiasing of the embeddings for gradient computation. This separation in the process allows for protection against MIA. Noisy C-VFL also does



(a) Scalar Quantization

(b) Top-K Sparsification

Figure 4: Compression Schemes


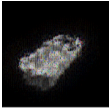
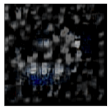

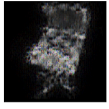
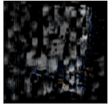

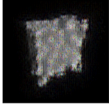
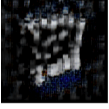
Mechanism	Original	VFL	Noisy-CVFL
Tub			
Chair			
Table			

Table 1: Results of Compression

not incurs a lot of communication overload if the order of the variance and quantization levels are close together (see Additional Analysis)

Future work explores the effect on model performance in terms of relating variance of binomial noise and test accuracy of the C-VFL model. Also, we investigate how the addition of bits in communication overhead be related as a function of ϵ and δ . We also have used image datasets and the model used are convolutional layers. We can extend this framework to datasets of other nature as well.

Moments accountant analysis can allow us to calculate how much privacy budget was consumed to make the model private.

References

- [1] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *Advances in Neural Information Processing Systems*, 31, 2018.
- [2] Timothy Castiglia, Anirban Das, Shiqiang Wang, and Stacy Patterson. Compressed-VFL: Communication-efficient learning with vertically partitioned data, 2022.
- [3] Tianyi Chen, Xiao Jin, Yuejiao Sun, and Wotao Yin. Vaf: a method of vertical asynchronous federated learning. *arXiv preprint arXiv:2007.06081*, 2020.
- [4] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.
- [5] Xiao Jin, Ruijie Du, Pin-Yu Chen, and Tianyi Chen. Cafe: Catastrophic data leakage in federated learning. 2020.
- [6] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.
- [7] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32, 2019.