Federated Learning and Prevalent Challenges

Huzaifa Arif

***Abstract*—We begin exploring the prevalent challenges in the federated learning frameworks.Robust federated learning procedures require development of algorithms that have efficient resource utilization,are communication efficient and provide defense to "data leakage" attacks.To address these concerns,we summarise the works of [1] and [2].[1]develops a control algorithm that solves the distributed gradient descent problem with budget constraints.[2] develops a CAFE loss that demonstrates very high data leakage in the vertical federated algorithms.**

I. Introduction

In a decentralized setting, where there are multiple clients and a server, the goal typically is to learn a global shared model utilizing data from the participating clients. In the case if there are multiple users who want to contribute in the training of the machine learning model, a more traditional approach was to make the clients upload their training data onto the server where the model gets trained or queried[3]. However, practically, this approach leads itself to higher communication costs and privacy vulnerabilities. Concretely, when the size of the training data is large, more bits of information need to be uploaded to train the machine learning model. From the perspective of privacy, we can see that a malicious server has direct access to training data thus compromises the privacy of the clients. To mitigate these concerns, [3] proposed a method called Federated Learning.

Federated learning methods are divided into two types: Horizontal Federated Learning (HFL)[3] and Vertical Federated Learning (VFL)[4].Consider mobile phone applications where different users want to jointly train a machine learning model.The different data from each user can help contribute train a machine learning model at the cloud server[3].If we assume that the samples at each device have the same feature space we have the HFL setting.Typically, HFL settings have the entire dataset distributed across the clients such that each client has only the subset of the training samples - typically disjoint. Simple cases assume that the data-distribution of the client's data is IID although recent research in non-IID[1] cases show promising results with regards to convergence of algorithms. In the conventional setting of HFL[1], [3], each of the clients does local update and communicates the updated parameters to the server. The server then does an averaging of the parameters and updates the resulting model before communicating them to the clients[3]. This setting allows a learning of the global model without having the client to upload their training data,thus providing relatively a higher level of client privacy. This setting also allows for more than one update of the model parameter at the client side before global aggregation but would be sub optimal because of a non-IID distribution of data at every client[1].Thus, an interesting problem become to solve for optimal weights with this non-IID distribution without requiring the clients to upload their gradients at every iteration of the algorithms.

Another federated learning algorithm can be if the clients are medical institutions who have different test results for the same patient [2]. In this case, the features have been partitioned at a client level for the same users.Learning in this setting is called VFL [5], [4], [2].This method [5], [4], [2] requires the clients only have a subset of features of all the samples of the training dataset ; the labels are kept only at one party which is typically the server. VFL in the literature assume either jointly learning a global (or centralized) model[4], [2] or a distributed model[5]. The VFL learning framework in [4] is closer to the HFL case where all the parties are learning the same global shared model. In this method, the clients exchange the intermediate results computed from the local features with each other and compute the gradients locally before uploading them to the server—-"local" here means on every client or server it is defined as opposite to "global" in this report. The server then updates the model from the locally client computed gradients. The other models more commonly assumed in the feature partitioning case is that of a distributed model[5]; the goal of the learning process here is to learn the local parameters of each participating device. The clients compute embeddings from the local features. These embeddings are then sent to the server which then sends the gradients with respect to each embedding to each of clients for gradient computation and updating of the local parameter. update[5], [6].

Thus, federated learning helps reduced communication overhead and improved privacy of the clients as client are not required to upload their private training data. However, while the federated settings have proceeded to tackle issues of communication overhead and privacy, recent works have pointed to the still remaining concerns with regards to these objectives.For instance, when the models are complex, the gradient/parameter dimension grows and practical challenges of higher communication costs need to be addressed[7]. Additionally, it was thought that communication of gradients would not leak private data of the clients. However, recent works have shown that communication of gradients presents possibilities of leakage attacks if the feature space, label space and model parameters are known by an honest but curious server[8], [2]. Thus, privacy still remains an important concern in federated learning. A resulting objective, therefore, of robust federated learning settings is to ensure development of mechanisms that ensure: reduction in communication costs and improvement in providing privacy of local training data. Additionally, ensuring better test accuracy of resulting model still remains with the aforementioned objectives. In this report we consider how these objectives are addressed by two novel papers in the ever-growing field of federated learning. In

Section II-A we discuss the [1] paper that looks at the efficient resource utilization to improve communication efficiency in the horizontal federated learning algorithms. In Section II-B we look at [2] that considers privacy concern or precisely the data leakage attacks in the vertical federated learning setting.We conclude the report in Section III by discussing the future directions that can be explored from these works.

II. Challenges in Federated Learning

A. Efficient Resource Utilization

1) Problem Formulation: [1] explores the efficient resource utilization in the HFL framework.Consider for instance,that there exist N nodes with local datasets such as $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_i, \ldots, \mathcal{D}_N$ [1]. The local loss function[1] defined at each node is :

$$F_i(\mathbf{w}) = \frac{1}{|\mathcal{D}_i|} \sum_{j \in \mathcal{D}_i} f_j(\mathbf{w}) \tag{1}$$

For the purpose of notational convenience, we have $D_i = |\mathcal{D}_i|$[1]. We also assume that the datasets at each client are disjoint. The global loss function is the weighted average of the local loss functions defined as below[1]:

$$F(\mathbf{w}) = \frac{\sum_{i=1}^{N} D_i F_i(\mathbf{w})}{D} \tag{2}$$

The objective of the machine learning model is to find the optimal weights by minimising the global loss function defined below :

$$\mathbf{w}^* = \arg\min F(\mathbf{w}) \tag{3}$$

The way [1] solves 3 is by the distributed gradient descent procedure[3].The local parameters are updated by running gradient descent on the local loss function(4) .The value of the parameter used for doing update at each iteration depends whether global aggregation was performed on the previous iteration or not.If global aggregation was performed then $\widetilde{\mathbf{w}}_i(t) = \mathbf{w}(t)$ (see (5))else $\widetilde{\mathbf{w}}_i(t) = \mathbf{w}_i(t)$. The local update equation is then defined as:

$$\mathbf{w}_i(t) = \widetilde{\mathbf{w}}_i(t-1) - \eta \nabla F_i(\widetilde{\mathbf{w}}_i(t-1)) \tag{4}$$

After every τ iterations we have global aggregation of the parameters with the following equation:

$$\mathbf{w}(t) = \frac{\sum_{i=1}^{N} D_i \mathbf{w}_i(t)}{D} \tag{5}$$

Typically, for distributed learning,we are constrained by the resources available at the edges and the server.These resources can be related to energy consumption for each local update or related to the maximum running time of local gradient descent [1]. Particularly, for distributed gradient descent, we are interested in understanding the trade-off between doing "local updates and global aggregation" [1] of the machine learning model under this resource budget constraint.

The nodes perform τ local updates between every two global aggregations and T is defined as the total number of iterations the system is running for. Additionally, we define that the local updates consume c number of resources while the global updates and aggregation consume b number of resources

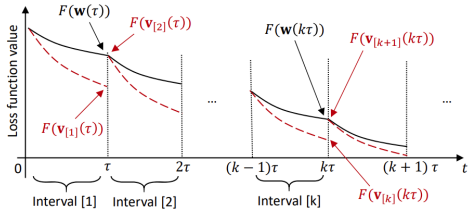


Fig. 3: Illustration of definitions in different intervals.

Figure 1. [1]Centralised vs Distributed parameters

with the total budget constraint R. Therefore,utilising this constraint,the objective becomes:

$$\begin{aligned} \min_{T,\tau} \quad & F(\mathbf{w}(T)) \\ \text{s.t.} \quad & T(c + \frac{b}{\tau}) \leq R \end{aligned} \tag{6}$$

2) Main Algorithm: The roots of the development of the algorithm come from the fact that we know the convergence bounds on the centralised gradient descent [1] and can use this to establish bounds in the distributed setting.We assume that K global aggregations happen in T iterations and in each $k \in K$ intervals we compare the behaviour of the distributed and centralised gradient descent. Consider the equation below which defines a centralised gradient descent on the global loss function 2:

$$\mathbf{v}_{[\mathbf{k}]}(t) = \mathbf{v}_{[\mathbf{k}]}(t-1) - \eta \nabla F(\mathbf{v}_{[\mathbf{k}]}(t-1)) \tag{7}$$

We compare 7 and 5 to argue for the upper bound on the convergence of the distributed gradient descent in Theoram 1 below.For each k round the behaviour of the parameters are shown by Figure 1

Note that we synchronise the weights in each round and observe the behaviour of the weights in each round.We observe that the weights in the distributed case diverge from the weights in each interval as time progresses in each interval as the data is distributed in a non-IID manner.Theoram 1 gives an upper bound in this case

Theorem 1:

$$||\mathbf{w}(t) - \mathbf{v}_{[k]}(t)|| \leq h(t - (k-1)\tau) \tag{8}$$

where

$$h(x) = \frac{\delta}{\beta}((\eta\beta) + 1)^x - 1) - \eta\delta x \tag{9}$$

The paper assumes that the global loss function is convex,ρ-Lipschitz and β-smooth. Another constraint assumed is that the local gradients have a δ divergence from the global gradients bounded as $\delta = \frac{\sum_i D_i \delta_i}{D}$ which allows us to quantify the upper bound on the weights in Theoram 1. Assuming that the $F(\mathbf{v}_{[\mathbf{k}]}(t)) - F(\mathbf{w}^*) \geq \epsilon$ for all t and $F(\mathbf{w}(T)) - F(\mathbf{w}^*) \geq \epsilon$ we have the bound on the global loss function given by Theoram 2 [1].

Theorem 2: $F(\mathbf{w}(T)) - F(\mathbf{w}^*) \leq \frac{1}{T(\omega\eta(1 - \frac{\beta\eta}{2}) - \frac{\rho h(\tau)}{\tau\epsilon^2})}$

Since the optimal point is a constant,[1] solves 6 by using the upper bound from Theoram 2[1] given by:

$$\begin{aligned} \min_{T,\tau} \quad & \frac{1}{T(\omega\eta(1-\frac{\beta\eta}{2})-\frac{\rho h(\tau)}{\tau\epsilon^2})} \\ \text{s.t.} \quad & T \leq \frac{R\tau}{c\tau+b} \end{aligned} \tag{10}$$

Utilising the maximum number of iterations T for a given budget and dividing the objective function by $\frac{R\omega}{c}$ the optimization problem to solve for τ becomes the following where $\varphi = \frac{\rho}{\omega\epsilon^2}$:

$$\tau^* = \arg\max_{\tau} \frac{\tau}{\tau + \frac{b}{c}}(\eta(1-\frac{\beta\eta}{2}) - \frac{\varphi h(\tau)}{\tau}) \tag{11}$$

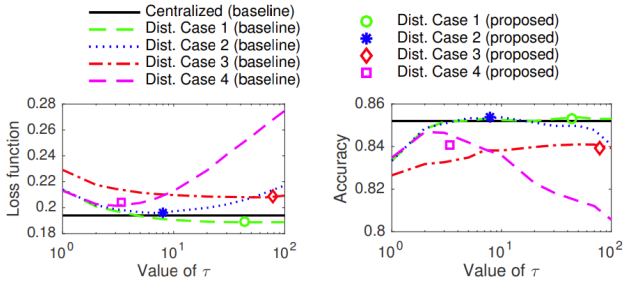
Thus the optimization problem becomes one of solving for τ given the budget constraints. Optimal T then becomes $T^* = \frac{R\tau^*}{c\tau^*+b}$

Overall,in a simplified version, the server is aware of the following information: the resource budget R and the control parameter φ. The values of τ are adaptively controlled by solving for 11 in every iteration.The systematic procedure at the server operates with the following steps:

1) The server sends the **w**(t) and τ to all the edge nodes.
2) The server waits to receive the local updates $\mathbf{w}_i$ (t).The received updates are used to perform global aggregation and calculate the new weights by 5.Also, the server receives estimation of local resource consumption c_i from each of the clients.This information,in addition to local estimate of b is used to calculate the resources consumed till time t.
3) The clients also send an estimate of the smoothness constant $\hat{\beta}_i$ and the gradients $\nabla F(\mathbf{w}_i(t))$ computed locally
4) The server uses this information to calculate an estimate of the overall smoothness constant $\beta = \frac{\sum_{i=1}^{N} D_i\hat{\beta}_i}{D}$ and the gradient divergence $\delta = \frac{\sum_i D_i\delta_i}{D}$
5) The quantities above are used to find optimal τ updates by solving the equation (11).This τ is sent to the clients to perform local updates.
6) There is also a check on the amount of resources consumed.In the case, the resources consumed have exceeded the local budget, the proposed method decreases τ before sending it to the clients and also set a stop flag to stop the gradient descent procedure.The model then outputs the final estimation of the parameters $\mathbf{w}(t)$.

The client side information proceeds with the following information:

1) It performs local updates based on τ number received from the server and sends the final parameters $\mathbf{w}_i$ after updates τ to the server
2) The client also performs local estimate of the smoothness constant $\hat{\beta}_i$ before sending them to server.
3) The client also sends the gradients to the server $\nabla F(\mathbf{w}_i(t_0))$ where t_0 is the iteration index of last transmission of $\mathbf{w}(t)$



(a) SVM (DGD) (b) SVM (DGD)

Figure 2. [1]Loss and accuracy performance on SVM

3) Results: The empirical results in the paper show performance of the machine learning model for the proposed algorithm in the distributed gradient descent scenario. Briefly we state the results on SVM only.For more detail the reader is referred to [1].The distribution of the IID and non-IID ness in the data is ensured by considering different cases of distribution of data as shown in Figure 2.The results are evaluated on the merit of performance with respect to loss and accuracy on setting that assumed a fixed number of local updates and the adaptive setting assumed in [1].Although optimal values of τ vary for each case,we do observe a certain range of τ values near which the optima exists for all cases.

B. Data Leakage in Federated Learning

1) Problem Formulation: We discuss the paper [2] to describe the privacy problem in the vertical federated learning method.The VFL framework assumed here is defined in the [4] paper. I discuss the more pertinent steps to discuss privacy in the report: Since the features are partitioned in this case amongst the clients, to compute the local gradients at each client, the server sends a batch of indices $\mathbf{s}_i^t$,a binary vector, at time t to all parties. This is known as "data index alignment" procedure and it helps to sample the appropriate samples at every client so that gradient can be computed to update the parameter. In this setting, the we assume a malicious server that is honest but curious ie: it does not actively affect the training process but tries to infer the private training data from each client utilizing its local information about the model parameters and its gradients This is commonly known as the "data leakage" problem. Previous works and in particular [8] were the first to argue an almost perfect data leakage in the federated setting.Consider for instance in a typical federated problem, [8] show that generating fake data samples and minimising the objective function 12 allows for an almost perfect recovery of the training data.

$$\mathbf{x}^{'*}, \mathbf{y}^{'*} = \arg\min_{\mathbf{x}^{'}, \mathbf{y}^{'}} ||\nabla W - \nabla W^{'}||_2^2 \tag{12}$$

However, this problem is intractable if we have large batch of data 13. The sum and the aggregated numbers do not have

a one- one mapping thus the below problem cannot give a unique solution:

$$\hat{\mathcal{D}}' = \arg\min_{\hat{\mathcal{D}}'} \Big\| \frac{1}{K} \sum_{(\mathbf{x}_n, y_n)\in\mathcal{D}} \nabla_{\mathbf{\Theta}} L(\mathbf{\Theta}, \mathbf{x}_n, y_n) - \frac{1}{K} \sum_{(\hat{\mathbf{x}}_n, \hat{y}_n)\in\hat{\mathcal{D}}} \nabla_{\mathbf{\Theta}} L(\mathbf{\Theta}, \hat{\mathbf{x}}_n, \hat{y}_n) \Big\|_2^2 \quad (13)$$

However,[2] proposes that doing this large batch gradient computation is not immune to data recovery.The authors propose a CAFE loss that allow for data leakage in this scenario.

2) Main Algorithm: CAFE observes that the server chooses the batch indices and sends them to the client;thus the server can utilise that information to obtain a stronger recovery.Each of the clients can a have complex model with many convolutional layers and a fully connected one.Figure 3 shows the model structure at every client. The main algorithm is a three step procedure: The first step in this procedure is to recover the gradients of loss with respect to the outputs of the first fully connected (FC) layer. For a certain client, let us assume for one sample, that the input to the first FC layer is $\mathbf{h}_n = h(\mathbf{\Theta}_c, \mathbf{x}_n) \in \mathbb{R}^{d_1}$.$\mathbf{\Theta}_c$ are the parameters of the first FC layer and $\mathbf{u}_n$ denote the output of the first FC layer.

$$\mathbf{u}_n = \mathbf{\Theta}_1^T \mathbf{h}_n + \mathbf{b}_1 \quad (14)$$

For training data $\mathcal{D}$ the inputs to the first FC layer are concatenated as $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N]^T \in \mathbb{R}^{N\times d_1}$ and the outputs are concatenated as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]^T \in \mathbb{R}^{N\times d_2}$ The gradient with respect to the output of the first FC layer $\mathbf{U}$ is

$$\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) = \frac{1}{N} [\nabla_{\mathbf{u}_1} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_1, y_1), \nabla_{\mathbf{u}_2} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_2, y_2), \ldots, \nabla_{\mathbf{u}_N} \mathcal{L}(\mathbf{\Theta}, \mathbf{x}_N, y_N)] \quad (15)$$

For a batch of data in the t^{th} iteration $\mathcal{D}(\mathbf{s}^t)$ we have the gradients with respect to the bias as

$$\nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(s^t)) = \sum_{n=1}^{N} \mathbf{s}^t[n] \nabla_{\mathbf{u}_n} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}^t)) \quad (16)$$

The server has access to 16 as clients upload their gradients to the server but it does not have access to $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$. The server's first job is to find an estimate of $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$.Therefore,the server generates a random intiliaztion of $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$ as $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]^T \in \mathbb{R}^{N\times d2}$ and solves the following optimization problem:

$$\mathbf{V}^* = \arg\min_{\mathbf{V}} \mathbb{E}_{s_i \sim \textbf{Unif}(\mathcal{S})} [||\mathbf{V}^T \mathbf{s}_i - \nabla_{\mathbf{b}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)||_2^2] \quad (17)$$

We can see that in expectation the minbatch stochastic gradient descent has similar convergence to the gradient descent algorithm and thus optimising the term inside the expectation will allow us to converge to the optimum. The authors [2] prove that if the batch size $\mathbf{K} \leq N$ then the above objective function is strongly convex and therefore has a unique solution.Thus this approach is one step closer to achieving the a unique solution for large batch data size.

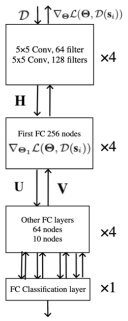


Figure 3. [2]Model Structure for the VFL case

The second step in this process is to recover inputs to the first FC layer $\mathbf{H}$.Using chain rule we have $\nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}) = \mathbf{H}^T \nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$[2]. The previous step allows us to calculate an estimate of $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$.Now using the recovered results we can solve the following optimization problem[2]:

$$\hat{\mathbf{H}}^* = \arg\min_{\hat{\mathbf{H}}} \mathbb{E}_{\mathbf{s}_i \sim \textbf{Unif}(\mathcal{S})} [|| \sum_{n=1}^{N} \mathbf{s}_i[n] \hat{\mathbf{h}}_n (\mathbf{v}_n^*)^T - \nabla_{\mathbf{\Theta}_1} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)||_F^2] \quad (18)$$

For 18 the author also argue that if the sample size $N \leq d_1$ where d_1 which is the dimension of the inputs to the first FC layer ie: $\mathbf{u}_n$, is smaller than the input size N the above objective is still strongly convex, guaranteeing a unique solution.

We now utilise these recovered terms to construct the overall loss function 20 called the CAFE loss [2] and perform the optimization as 19.Notice that this is an extension to [8] in that additional regulariser terms have been included.:

$$\hat{\mathcal{D}}^* = \arg\min_{\hat{\mathcal{D}}} \mathbb{E}_{\mathbf{s}_i \sim \textbf{Unif}(\mathcal{S})} \mathcal{F}_3(\hat{\mathcal{D}}; \mathbf{s}_i) \quad (19)$$

$$\mathcal{F}_3(\hat{\mathcal{D}}; \mathbf{s}_i) = \alpha ||\nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D}(\mathbf{s}_i)) - \nabla_{\mathbf{\Theta}} \mathcal{L}(\mathbf{\Theta}, \hat{\mathcal{D}}(\mathbf{s}_i)||_2^2 + \beta \underline{TV}_{\xi}(\hat{\mathcal{X}}(\mathbf{s}_i)) + \gamma \sum_{n=1}^{N} ||\mathbf{s}_i[n](\hat{\mathbf{h}}_n^* - \widetilde{\mathbf{h}_n})||_2^2 \quad (20)$$

The third term in the equation,known as the internal representation regularizer, is obtained from the previously calculated steps $\hat{\mathbf{H}}^*$ and $\widetilde{\mathbf{h}}_n = h(\mathbf{\Theta}_c, \hat{\mathbf{x}}_n) \in \mathbb{R}^{d_1}$ is the the representation from fake data .We include another regulariser truncated TV-norm of $\hat{\mathcal{X}}(\mathbf{s}_i)$ which is zero if the TV-norm of $\hat{\mathcal{X}}(\mathbf{s}_i) = \{\mathbf{x}_n | \mathbf{s}_i[n] = 1\}$ is smaller than ξ.Two methods are proposed to solve the CAFE Loss; one of the methods can be through the nested loop that does the estimation of the terms $\mathbf{H}$ and $\nabla_{\mathbf{U}} \mathcal{L}(\mathbf{\Theta}, \mathcal{D})$ separately and then calculates the CAFE loss. The other single loop procedure does the estimation of the parameters and solving for CAFE loss in the same iteration.

3) Results: The model structure assumed for all of the clients assume the structure as in Figure 3 .

Experiments were done using the MNIST,CIFAR-10 and Linnaeus datasets.We assume that there are 4 workers and different batch sizes are assumed. The error is measured in terms of PSNR .Figure 5 shows great recovery for batches upto size 100.

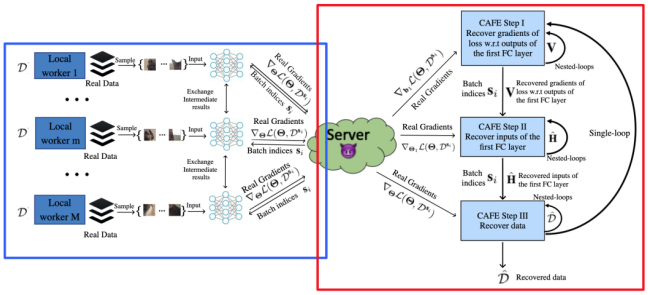


Figure 4. [2]Overview of CAFE

PSNR / Dataset / K	CIFAR-10	MNIST	Linnaeus 5
10	30.83	32.60	28.00
20	35.70	39.00	30.53
40	31.83	43.15	33.22
80	36.87	47.05	30.43
100	38.94	47.50	29.18

Figure 5. [2]PSNR vs Batch Size

PSNR / Dataset / Method	CIFAR-10	MNIST	Linnaeus 5
CAFE	31.83	43.15	33.22
DLG	9.29	7.96	7.14
Cosine Similarity	7.38	7.84	8.31
SAPAG	6.07	3.86	6.74
BN regularizer	18.94	13.38	8.09
GC regularizer	13.63	9.24	12.32

Figure 6. [2]Effect of regularizers

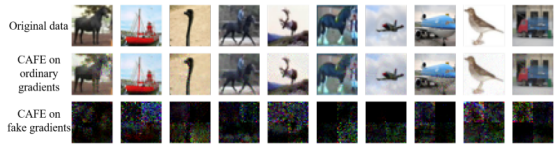


Figure 7. [2]Performance of CAFE in recovery

We observe that the performance is better than the state of the art algorithms measured in terms of PSNR values. See Figure 7 for the comparison. Analysis on the effect of different regularisers on the performance of CAFE shows the contribution of the internal representation regularizer to be most significant in the recovery [2].

III. Future Research Directions

Each paper discusses the challenges prevalent in federated learning in its own unique ways. [1]Discusses algorithmic communication efficiency by arguing efficient resource utilization in HFL scenario for the distributed gradient descent[1]. Certain assumptions about the smoothness of the function and consumption capacity at the edge devices and the server is important for the server to estimate local number of updates and thus future work can investigate methods that do not require such an estimation.However,more importantly,this paper considers efficient resource utilization in HFL leaving out privacy analysis which is discussed in papers like [8]. Particularly, in HFL we can inverstigate topics like how distributed gradient descent and the number of local updates have effect on privacy.One of the more pertinent research direction that can be directly deduced is to look at how the control algorithm proposes to solve the privacy problem.Optimising between the local updates and global aggregation can have effect on data leakage attacks like CAFE or DLG.

CAFE loss and its performance in HFL can be evaluated as well.The paper [2] does discuss the fact that if the data indices are known by the attacker then CAFE can be readily applied to the HFL setting but how to implement CAFE if the indices are not known in the HFL setting which is a more typical case.Another interesting direction is that of extending the VFL setting that is assumed for a unified model [4] to extend to the distributed model setting[5].In my view this could be done by optimising over the embeddings instead of gradients to learn the distributed model.

Another extension of the robust VFL setting here is that of arguing for better defense mechanisms.The paper proposes a defense method of generating fake gradients and compromising the quality of the leakage attack. A common framework to quantify the efficacy of defense mechanism is to discuss the differential privacy mechanism.Methods such as quantization [7] seem to provide reduced communication overheads but its privacy arguments are still not discussed widely.[9] have proposed to investigate the effect of quantization in privacy but more research can be done in this area.

A more general framework is to develop objectives that quantify the performance of algorithms that combine each of the objectives: reducing communication cost and the privacy setting.Thus,we can use this framework to build robust and practical federated learning models

References

[1] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 63–71.

[2] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "Catastrophic data leakage in vertical federated learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[4] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, jan 2019. [Online]. Available: https://doi.org/10.1145/3298981

[5] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[6] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features," *arXiv preprint arXiv:1912.11187*, 2019.

[7] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2021–2031.

[8] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," *Advances in Neural Information Processing Systems*, vol. 31, 2018.