A Secure Algorithm for Deep Learning Training Under GAN Attacks

Aseem Prashar, and Sergio A. Salinas Monroy*
Department of Electrical Engineering and Computer Science
Wichita State University
Wichita, USA
prasharaseem@gmail.com, sergio.salinasmonroy@wichita.edu

***Abstract*—Deep neural networks have outperformed traditional machine learning approaches for many tasks, and are the tool of choice in many fields. However, directly applying these techniques in fields that deal with private data is challenging. The reason is that a third-party, which organizations may not trust, usually needs to centrally collect private data. To overcome this challenge, researches have proposed distributed training algorithms that allow multiple users to collaboratively train their local deep learning models without sharing private datasets. However, these approaches are vulnerable to recently proposed attacks where a malicious user can replicate private data from another user by compromising the collaborative training algorithm. In this paper, we propose a privacy-preserving distributed deep learning algorithm that allows a user to leverage the private datasets from a group of users while protecting its privacy. Our algorithm prohibits this user from ever sharing the parameters of its model, and thus it prevents malicious users from compromising the training and replicating the user's private data. We conduct extensive experiments and observe that our algorithm can achieve a model accuracy of 95.18%, which is the same accuracy that previous approaches that are vulnerable to attacks can achieve.**

***Index Terms*—Neural Network, Deep learning, GANs**

I. Introduction

In the past few decades, deep learning has generated a lot of interest in the research and academic community due to its great ability to automatically classify large amounts of data. This has led to breakthroughs in many fields ranging from autonomous driving, and natural language processing to genetic research and speech processing [1]–[5]. This revolutionary technology is especially fruitful for large corporations that need to automatically process very large data sets to provide deep learning inference services to their users. For example, data collection at Facebook enabled them to create DeepText, a text understanding engine that is able to extract meaningful context from text [6].

Although it is possible to collect large-scale data to train deep learning models in some application domains, e.g., online social networks, there are other fields where such centralized data collection is currently infeasible due to privacy concerns. In particular, users' data can contain instances of private information that needs to be kept secret from companies that centrally collect data to provide deep learning services [7]. Such private data includes intellectual property (IP), medical data protected by Health Insurance and Portability and Accountability Act (HIPA), and student records protected by the Family Educational Rights and Privacy Act (FERPA) [8]–[10].

978-1-7281-7315-3/20/$31.00 © 2020 IEEE

Instead of sharing private data with a third-party, users could locally train their own deep learning models using their own data. However, since a single user only has access to a relatively small data set, its locally trained deep learning model has low accuracy. Moreover, locally trained models can suffer from over generalization that can prevent them from being used in practical scenarios. Hence, it is necessary to develop secure training algorithms that can achieve high accuracy while preserving privacy. .

Researchers have proposed several techniques to enable privacy-preserving training of deep learning neural networks. Specifically, researchers have used secure multiparty computations (SMC) to encrypt users' private data and distribute the training computations over many users, which preserves users' privacy [11]–[14]. This approach, however, is impractical for most users since the encryption computations in SMC add a significant computational overhead and they are required to be online during the training.

Differential privacy has also been proposed to protect users' data privacy in deep learning training [15]–[17]. By adding artificial noise to the training data, differential privacy allows a remote server to train a deep learning model without learning individual data samples. However, the accuracy of the deep learning model under a large number of users decreases due to the additional noise needed to protect their privacy. Thus, differential privacy suffers from poor scalability.

More recently, Shokri et al. [18] researchers have proposed privacy-preserving distributed learning. In this approach, users locally train a deep neural network using their own data sets. Users then upload a small subset of their model parameters to a server, where they are aggregated. The server then sends the aggregated parameters back to the users, who can use them to improve their local models. Since users only share their model parameters with the server, they avoid directly disclosing their private data sets to the server of other users. Under this algorithm, the deep learning models at the users can achieve a high accuracy.

Unfortunately, the approach [18] is vulnerable to the generative adversarial network (GAN) attack described by Hitaj et al. [19], where a malicious user is able to replicate data samples

from a victim user. In this attack, the malicious user uploads specially crafted model parameters to the aggregation server, which are eventually downloaded by the victim user to train its local model. The victim trains its local model using the compromised aggregated parameters, which results in update model parameters that contain private information about its data samples. Since the victim uploads its compromised model parameters to the server, the malicious user gains access to them. By iteratively feeding the aggregate parameters that contain private information about the victim's data sample to its GAN, the malicious user can replicate the victim's data samples, and thus compromise its privacy.

In this paper, we propose a novel privacy-preserving distributed deep learning framework that prevents the GAN attack described in [19] while allowing the users to train their local models with high accuracy. Specifically, instead of attempting to protect the privacy of all users as in [18], we focus on protecting the privacy of a single user, called the reference user. Our approach can potentially be used in a scenario where the privacy of one of the participant in distributed learning is valued over other participants. For instance, a medical facility that has classified medical data can still benefit from distributed deep learning using our proposed system. Our approach is also can also be adopted in a scenario where participants are compensated for their parameters due to the privacy risks they incur.

The GAN attack in [19] exploits 1) the uploaded model parameters from the victim, and 2) the fact that the aggregation server unconditionally accepts model parameters from any user in the system, which could potentially be malicious. Therefore, in our algorithm, we prohibit the reference user from uploading its model's parameters to the server, and the server only accepts parameters from randomly selected users at each iteration. To verify the efficacy of our approach, we run extensive experimental evaluations on Amazon Web Service's Elastic Compute Cloud (AWS EC2). We observe that our approach maintains the accuracy of the deep learning model compared to a training algorithm that does not protect privacy, even when the number of users who are allowed to upload parameters to the aggregation server is reduced. We also measure the trade-off between privacy and accuracy, and show that the main user can easily choose the most appropriate trade-off by tuning the user selection probability.

II. Problem Formulation

In this section, we first provide a brief background on deep learning, and then describe our considered system and threat models.

A. Deep Neural Networks

In this section, we describe the architecture of deep neural networks and their training methods.

1) Architecture: In this work, we use the multilayer perception (MLP), one of the most common deep neural network architectures. Figure 1 shows the structure of a typical classification MLP with m input nodes, j outputs nodes, and N hidden layers. Each layer has I neurons. Intuitively, this MLP takes a data sample represented as a vector of length m on its input layer, and outputs the probability that it belongs to the jth category on the jth output neuron.

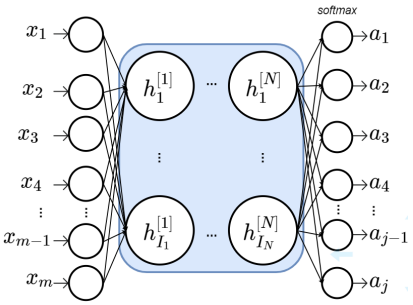

Fig. 1. A neural network with m inputs, j outputs, N hidden layers, and I neurons per layer.

The output of the ith neuron at layer k is given by,

$$a_k = f(W_k a_{k-1}) \tag{1}$$

where f is the activation function, W_k is the weight matrix of layer k, and a_{k-1} is the vector of neuron outputs from the the $k-1$th layer. We denote the flattened vector of all weights of the MLP by w.

We note that our results and formulations are generalizable to other deep neural network architecture besides the MLP.

2) Training: Before a neural network can be used to perform inference, e.g., classify images, it needs to be trained to learn the highly non-linear relationships between the inputs and the correct outputs.

To train the MLP in Section II-A1, we use the stochastic gradient descent (SGD) algorithm, which aims to find the the weights W_k for all layers k that result in the inferences with the highest accuracy. SGD is an iterative algorithm where each iteration has two steps: forward propagation and back propagation.

In the forward propagation step, the SGD algorithm feeds a randomly selected subset of samples, called the mini-batch, one by one to the MLP, and collects the MLP's output for each sample based on the current value of its weights. Then, the SGD computes the error function, which measures the difference between the outputs of the neural network and the correct classification solutions, which we call labels. In this work, we use the mean squared error function, i.e.,

$$E = \frac{1}{n}\sum_{n=1}^{n}(y_i - \hat{y_i}), \tag{2}$$

where n is the the number of samples in the training dataset, y_i is the output calculated by the neural network and $\hat{y_i}$ is the correct label for the ith sample.

In the back propagation step, the SGD algorithm computes the partial derivative of the error function with respect to the parameters of each neuron in the deep neural network, which indicate how much each parameter contributed to the error. Based on the partial derivatives, the SGD algorithm updates

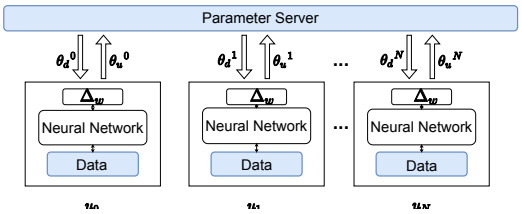

Fig. 2. An architecture for privacy-preserving distributed learning.

the MLP's parameters. Specifically, the jth parameter of the flattened vector of parameters w is given by:

$$w_j = w_j - \alpha \frac{\partial E_i}{\partial w_j}, \tag{3}$$

where E_i is the value of the error function defined in 2 computed over minibatch i, α is the learning rate, and $\frac{\partial E_i}{\partial w_j}$ denotes the partial derivative of E_i with respect to parameter w_j.

The SGD algorithm continues with the forward pass, error calculation, back propagation and parameter update iteration until a minimum error is achieved or a maximum number of iterations is reached [20].

B. System Model

We consider a distributed deep learning system formed by a set of N users $\mathcal{U} = \{u_0, \ldots, u_N\}$ and a parameter server (PS). User u_0 aims to train a local deep neural network using its own training data set d_{u_0} as well as the data sets of the other users d_{u_i}. The training data sets d_{u_i}'s of all users contain private information that cannot be shared with each other or with the PS. We assume the training data set at the reference user is significantly smaller than the total training data available in the system. Otherwise, the reference user could achieve a comparable accuracy without having to participate in the distributed learning. User u_0's deep neural network is a multilayer perceptron (MLP) as described in Section II-A1 for image classification.

C. Privacy-preserving Distributed Learning under the Semi-honest Users Threat Model

To allow user u_0 to train its deep learning network while preserving its privacy, we could use the distributed learning approach proposed by Shokri et al. [18]. In the approach proposed in [18], each user $u_i \in \mathcal{U}$ trains a local deep learning network, and uploads certain parameters of its model to a server. The server aggregates the parameters from all users and then allows the users to download some of them. This iteration can be repeated until the accuracy of the users' local deep learning models achieves a minimum value. By only sharing certain parameters of their local models with a parameter server, users avoid having to transmit any of their private samples to each other or to the server.

Specifically, let $w^{(global)}$ and w_i be the parameters at the parameter server and at user u_i, respectively. Let θ_d^i and θ_u^i be the percentage of parameters that user u_i downloads and uploads, respectively. Then, to initialize the system, each user u_i randomly sets its parameters w_i to a randomly chosen value and selects a learning rate, α_i. Then, at each iteration, user u_i downloads $\theta_d^i \times |w_i|$ parameters from the parameter server and overwrites the corresponding parameters in w_i, where $|w_i|$ denotes the total number of parameters in w_i. User u_i then runs the stochastic gradient descent algorithm described in II-A2 with the updated parameter vector as the initial vector. Based on the results of the SGD, user u_i uses (3) to calculate $w_i^{(new)}$.

Next, user u_i determines which parameters to upload to the server based on how much each parameter changed compared to the previous iteration. To this end the user first computes

$$\Delta w_i = w_i^{(new)} - w_i \tag{4}$$

which measures the changes between the old and new local parameters. Then, the user forms set $\mathcal{S}_i$ with the indices of the elements in vector Δw_i that have the top $\theta_u^i \times |\Delta w_i|$ values, where $|\Delta w_i|$ is the size of Δw_i. The user u_i forms vector $w_{\mathcal{S}_i}$, which contains the parameters in $w_i^{(new)}$ whose indexes are in set $\mathcal{S}_i$, and uploads it to the parameter server. Note that $w_{\mathcal{S}_i}$ is set to zero in the remaining positions.

After receiving the parameters from user u_i, the parameter server updates its own vector as follows:

$$w^{(global)} = w^{(global)} + w_{\mathcal{S}_i}$$

Intuitively, by using $w_{\mathcal{S}_i}$, which contains information about the parameters in the local model of u_ii that have undergone the largest changes during the local SGD training, the parameter server can improve its own parameters that are then downloaded by other users to further improve their own models.

Although this approach can train local models with high classification accuracy, it assumes a semi-honest threat model for the users, which is not realistic. That is, it assumes that the users will attempt to find private information about other users based on the parameters that they download from the server, but do not deviate from the procedure described above. As we will see in the next section, users can launch active attacks where they maliciously modify the parameters that they upload to recreate samples from other users.

D. A Malicious Threat Model for Distributed Deep Learning

In this work, we consider a malicious threat model for both the parameter server and the users. Specifically, in addition to attempting to learn private information about other users' dataset based on the parameters at the server as in the semi-honest threat model, the attacker in the malicious threat model uploads specially crafted parameters. The malicious parameters lead other users to upload their parameters in such a way that they disclose private information about their data sets. This information can the be downloaded and used by the attackers to recreate private data samples from the victims.

Such an attack has been described by Hitaj et al. [19]. In particular, the attack operates as follows. Suppose user $u_A \in \mathcal{U}$ is malicious and targets another user $u_V \in \mathcal{U}$. Further assume that all users, including the malicious one, agree on the hyper parameters of a neural network architecture such

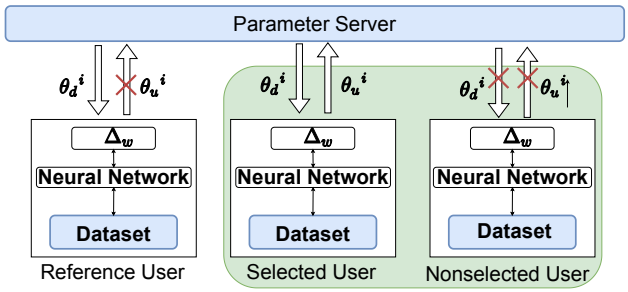

Fig. 3. One iteration of our proposed algorithm for privacy-preserving distributed learning.

as the number, size and type of layers, and the classification labels as described in Section II-B.

The victim user u_V declares that its private training data set contains samples for the labels $[a, b]$. The adversary u_A declares to have the classification labels $[b, c]$ in its training data set, which means it declares that it has no data for label a. By deploying the attack, the adversary aims to replicate samples with the same probability distributions as the private samples with label a at user u_V.

The adversary user u_A then locally uses a generative adversarial network to generate samples that have the same distribution as samples with label a at u_V. The adversary labels these malicious samples as belonging to category c, and uses them to retrain its local model. After the parameters from the malicious users are uploaded and aggregated by the server, the victim user u_V downloads them to retrain its model. Since the compromised parameters where obtained using samples of label a misclassified as label c, the parameters of the retrained model that are related to identifying label a will experience the largest changes, and thus will be uploaded by the victim to the server. The adversary u_A can then download the aggregated parameters that contain the parameters from the victim user. By iteratively following this procedure, the adversary u_A collects enough information about samples of label a at u_V to recreate their probability distribution.

III. A Privacy-preserving Distributed Learning Algorithm

In this section we describe our proposed privacy-preserving distributed learning algorithm that prevents the malicious user from recreating private samples from other users as described in Section II-D. Our main idea is to focus on protecting the privacy of a single user instead of attempting to protect the privacy of all users as in [18]. Our main approach to protect the reference user's privacy is to prohibit the reference user from uploading its parameters to the parameter server and only allow the other users to upload their parameters a limited number of times at randomly chosen iterations.

Our proposed algorithm works as follows. We assume a reference user u_0 aims to train a local deep neural network using its local training data set as well as the training data sets from other users as described in Section II-B. We then form a set $\hat{\mathcal{U}}$ of users who upload parameters by randomly selecting each user from the set $\mathcal{U}$ with probability p. The reference user is never chosen.

User $u_i \in \hat{\mathcal{U}}$ downloads a fraction θ_d^i of the parameters in the server, and uses them to overwrite the corresponding local parameters in w_i. User u_i then trains its network privately using its local data set d_i, and obtains the new parameters $w_i^{(new)}$. Then, as described in (4), user u_i calculates the change in parameter values by computing Δw_i, groups the indexes of the elements in vector Δw_i with the largest $\theta_u^i \times |\Delta w_i|$ values in set $\mathcal{S}_i$, and forms a vector $w_{\mathcal{S}_i}$ with the elements in $w_i^{(new)}$ that correspond to indexes in $\mathcal{S}_i$. User u_i uploads $w_{\mathcal{S}_i}$ to the parameter server who aggregates them.

After all selected users in $\hat{\mathcal{U}}$ have uploaded their parameters, reference user u_0 downloads a fraction θ_d^i of the aggregated parameters at the server. It then uses the downloaded parameters as initial parameters to retrain its local model. The iteration continues until the model at the reference user achieves a minimum accuracy, or until a maximum number of iteration is reached.

We note that in this algorithm the reference user u_0 never uploads its parameters to the server, and thus its private data is protected from the parameter server and from malicious users who may attempt to launch the attack in [19].

We summarize our proposed algorithm as follows:

Algorithm 1 Our proposed privacy-preserving distributed learning algorithm.

Require: $\mathcal{U}$

1: **for** N iterations. **do**
2: Choose user $u_i \in \mathcal{U} \setminus u_0$ with probability p and add it to $\hat{\mathcal{U}}$.
3: **for** Every $u_i \in \mathcal{U}$ **do**
4: User u_i downloads the θ_d^i of parameters from the server, and replaces the corresponding parameters in w_i.
5: User u_i retrains its model using its local data set d_i and obtains the new parameters w_i^{new}.
6: User u_i uploads the vector of parameters with the largest change u_i uploads $w_{\mathcal{S}_i}$ to the server.
7: **end for**
8: Reference user u_0 downloads all the parameters from the server.
9: Reference user u_0 retrains its model using the downloaded parameters and its local data set d_0
10: **if** Accuracy of model at u_0 is equal or grater than A. **then**
11: Stop the iteration.
12: **end if**
13: **end for**

Ensure: The parameters w_0 at user u_0.

IV. Experiment Results

To evaluate the performance of our proposed privacy-preserving approach, we implement it on a commercial cloud service provider, and measure the learning performance of

the user u_0, and compare it to the learning performance of a centralized deep neural network architecture that does not protect privacy, and to the distributed learning architecture proposed by Shokri et al. [18] that is vulnerable to the attack in [19].

A. Experiment Setup

We implement all algorithms using Torch with the neural network packages in the scripting language LUAJIT, and run them on an M4 instance on the Amazon Web Service's Elastic Compute Cloud (AWS EC2), which has 2.4 GHz Intel Xeon E5-2676 v3** Processor, 4 VCPUS and 16 GB RAM. As described in Section II-B, we use a a multilayer perceptron (MLP) in a feed forward arrangement. The MLP classifies 32×32 pixel images of hand-written numbers. Accordingly, our MLP has 1024 input nodes corresponding to each pixel in the images, and the output layer is a tensor of size 10 where each output corresponds to the probability that a given input is a specific number between 0 and 9. The model has 2 hidden layers where the non-linear ReLU activation function is applied to the output of each hidden layer. The first hidden applies a linear transformation and produces a tensor of size 128 as its output. The second hidden layer accepts a tensor of size 128 as input, and outputs a tensor of size 64. The last layer of the model is a log soft max layer.

We trained the MLP using using the MNIST dataset of images of hand-written numbers [21]. The MINST dataset contains 60,000 images in its training dataset and 10,000 images in its test dataset. To implement our proposed algorithm, we set the size of the local dataset of each participant to 1 % of the training dataset images. The reference user starts with a training set of 60 images. We set the learning rate α_i for all users to 0.1. The mini-batch size for stochastic gradient descent training is set to 10 samples.

V. Results

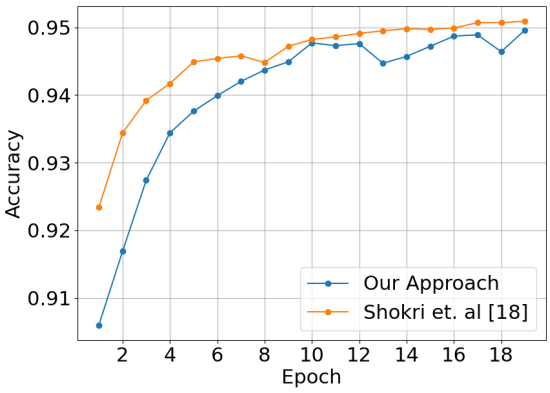

Fig. 4. Comparison between randomized interaction and complete interaction with server.

We first compare the accuracy of the reference user under our proposed privacy-preserving algorithm to that of the algorithm proposed by Shokri et al. [18]. Figure 4 shows the accuracy of the reference user u_0's deep learning neural network trained under our proposed algorithm, and under the algorithm in [18]. To make a fair comparison, we set the number of users to 20 for both approaches. We set the probability of choosing a non-reference user from $\mathcal{U}$ to 0.5. In the approach by [18], all users upload and download with a probability of 1. We see that the reference user u_0 is able to obtain a deep learning model with an accuracy that is comparable to the one obtained by [18], which is vulnerable to the attack in [19]. We next investigate the impact of the number of non-reference users on the accuracy of the reference user. Figure 5 shows the accuracy of the reference users u_0's deep neural network under a varying number of epochs for different numbers of total users in the system. We observe that the reference user u_0 achieves a higher accuracy as the total number of participants in the system increases. For example, the highest accuracy is achieved with 50 participants, while the lowest one is achieved with 10 participants. This means the parameters uploaded to the server increases with the number of users, improving the accuracy of the model. The reason is that a larger number of participants brings information from additional local datasets.

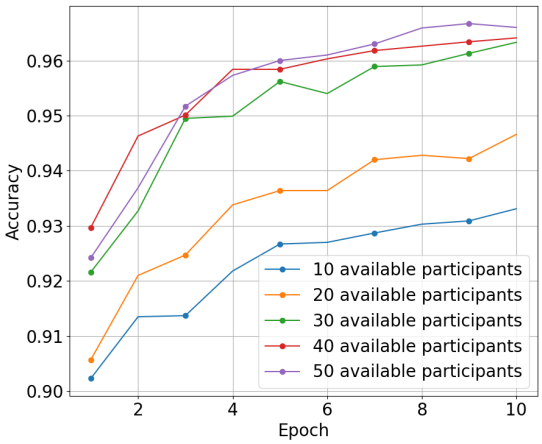

Fig. 5. Accuracy of Distributed SGD for varying participants available for interaction with the PS.

Figure 6 shows the accuracy of reference user u_0 for different probabilities of choosing non-reference users p and adding it to $\bar{\mathcal{U}}$. We see that a greater probability p results in a higher model accuracy for the reference user u_0. The highest accuracy is obtained when all non-reference users are added to the set $\bar{\mathcal{U}}$.

In Figure 7, we plot the accuracy of the reference user u_0 for different upload gradient selection fractions for the non-reference users. The plot shows that the accuracy of u_0's model increases with the percentage of uploaded parameters θ_u^i (for all i) is However, once value of θ_u^i reaches 10% the increase in accuracy for user u_0 is small. Thus, by sharing only one tenth of their parameters, the non-reference users can help u_0 achieve a high accuracy.

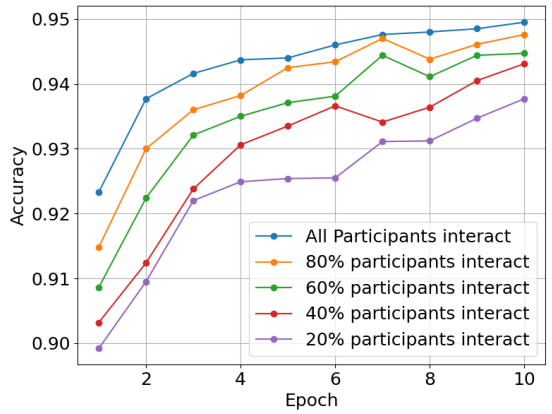

Fig. 6. Accuracy of Distributed SGD for varying probability of interaction with the PS.

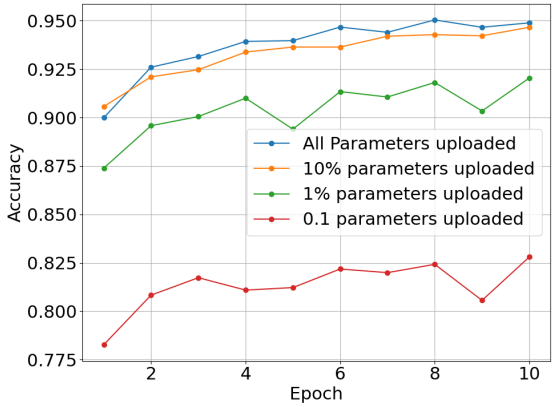

Fig. 7. Accuracy of Distributed SGD for different upload gradient selection rate.

VI. Conclusion

In this work, we have investigated the problem of protecting the privacy of a user in distributed training of deep learning models. We consider a system where a reference user aims to train a model with the help of other users. However, they are unable to directly share their data sets due to privacy concerns. Although there are some existing works on privacy-preserving distributed training, they can only protect against semi-honest attacker,i.e., attackers that follow the algorithms, and are vulnerable to malicious users that inject specially crafted parameters into the system to compromise the training procedure. To this end, we propose a distributed training algorithm that protects the privacy of the reference user under these types of attacks. The algorithm only allows the non-reference users to upload parameters to the server, and thus the reference user's private information is protected. Moreover, the non-reference users are only allowed to upload parameters at randomly chosen iterations, and thus have a limited ability to manipulate the training results. Compared to previous works, our solution does not rely on computationally expensive cryptographic process and can be used with any underlying neural network structure. Our experiment results show that the proposed algorithm can achieve the same accuracy compared to previous works that are vulnerable to attacks.

References

[1] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1891–1898.

[2] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[3] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, "Deep learning algorithm for autonomous driving using googlenet," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 89–96.

[4] P. Danaee, R. Ghaeini, and D. A. Hendrix, "A deep learning approach for cancer detection and relevant gene identification," in *PACIFIC SYMPOSIUM ON BIOCOMPUTING 2017*. World Scientific, 2017, pp. 219–229.

[5] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[6] A. Abdulkader, A. Lakshmiratan, and J. Zhang, "Introducing deeptext: Facebook's text understanding engine," *Facebook Code*, 2016.

[7] M. Chicurel, "Databasing the brain," 2000.

[8] A. Act, "Health insurance portability and accountability act of 1996," *Public law*, vol. 104, p. 191, 1996.

[9] B. Blechner and A. Butera, "Health insurance portability and accountability act of 1996 (hipaa): a provider's overview of new privacy regulations." *Connecticut medicine*, vol. 66, no. 2, pp. 91–95, 2002.

[10] D. Shultz, "When your voice betrays you," 2015.

[11] X. Ma, F. Zhang, X. Chen, and J. Shen, "Privacy preserving multi-party computation delegation for deep learning in cloud computing," *Information Sciences*, vol. 459, pp. 103–116, 2018.

[12] R. Sheikh, B. Kumar, and D. K. Mishra, "A distributed k-secure sum protocol for secure multi-party computations," *arXiv preprint arXiv:1003.4071*, 2010.

[13] H. Miyajima, N. Shigei, H. Miyajima, Y. Miyanishi, S. Kitagami, and N. Shiratori, "New privacy preserving back propagation learning for secure multiparty computation," *IAENG International Journal of Computer Science*, vol. 43, no. 3, pp. 270–276, 2016.

[14] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

[15] P. Vepakomma, T. Swedish, R. Raskar, O. Gupta, and A. Dubey, "No peek: A survey of private distributed deep learning," *arXiv preprint arXiv:1812.03288*, 2018.

[16] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[17] M. Chase, R. Gilad-Bachrach, K. Laine, K. E. Lauter, and P. Rindal, "Private collaborative neural network learning." *IACR Cryptology ePrint Archive*, vol. 2017, p. 762, 2017.

[18] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[19] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[20] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[21] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.