Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Digestive neural networks: A novel defense strategy against inference attacks in federated learning[☆]

Hongkyu Lee[a], Jeehyeong Kim[b], Seyoung Ahn[c], Rasheed Hussain[d], Sunghyun Cho[c], Junggab Son[a,1,*]

[a] Information and Intelligent Security (IIS) Lab, Kennesaw State University, Marietta, GA 30060, USA
[b] Korea Electronics Technology Institute, Seongnam, South Korea
[c] Department of Computer Science and Engineering, Major in Bio Artificial Intelligence, Hanyang University, Ansan, South Korea
[d] Networks and Blockchain Lab, Innopolis University, Innopolis, Russia

ARTICLE INFO

Article history:
Received 1 March 2021
Revised 3 June 2021
Accepted 14 June 2021
Available online 23 June 2021

Keywords:
Federated learning (FL)
Inference attack
White-box assumption
Digestive neural networks
t-SNE analysis
Federated learning security
ML Security
AI Security

ABSTRACT

Federated Learning (FL) is an efficient and secure machine learning technique designed for decentralized computing systems such as fog and edge computing. Its learning process employs frequent communications as the participating local devices send updates, either gradients or parameters of their models, to a central server that aggregates them and redistributes new weights to the devices. In FL, private data does not leave the individual local devices, and thus, rendered as a robust solution in terms of privacy preservation. However, the recently introduced membership inference attacks pose a critical threat to the impeccability of FL mechanisms. By eavesdropping only on the updates transferring to the center server, these attacks can recover the private data of a local device. A prevalent solution against such attacks is the differential privacy scheme that augments a sufficient amount of noise to each update to hinder the recovering process. However, it suffers from a significant sacrifice in the classification accuracy of the FL. To effectively alleviate the problem, this paper proposes a Digestive Neural Network (DNN), an independent neural network attached to the FL. The private data owned by each device will pass through the DNN and then train the FL. The DNN modifies the input data, which results in distorting updates, in a way to maximize the classification accuracy of FL while the accuracy of inference attacks is minimized. Our simulation result shows that the proposed DNN shows significant performance on both gradient sharing- and weight sharing-based FL mechanisms. For the gradient sharing, the DNN achieved higher classification accuracy by 16.17% while 9% lower attack accuracy than the existing differential privacy schemes. For the weight sharing FL scheme, the DNN achieved at most 46.68% lower attack success rate with 3% higher classification accuracy.

https://doi.org/10.1016/j.cose.2021.102378

1. Introduction

Conventional ML approaches for distributed data require aggregation of the data into a single repository. This is particularly problematic for several reasons. Not only the congregation requires excessive network resources, but the data center itself also requires a frequent update from mobile devices to make the up-to-date data available for proper training. The data often includes privacy-sensitive information, entailing a privacy concern on the fabrication of a monolithic data repository. Moreover, from the ML perspective, a huge dataset can slow down the learning speed.

In contrast, Federated Learning (FL) is a decentralized Machine Learning (ML) algorithm that is more suited to learn from distributed data (McMahan et al., 2017). The learning process in FL involves a central server and multiple participating mobile devices. It begins with the server's coordination of the learning model with the participating devices. In each communication round, the server broadcasts parameters generated from its global model to the devices. Each device then copies the parameters into their local model and trains it using the private data. The device generates updates, either parameters or gradients depending on the type of the FL protocols, that includes changes in the local model and sent it back to the server. To this end, FL intrinsically resolves the privacy and efficiency problems as the devices need to send a piece of information rather than whole data for training purposes.

The reason that FL is secure (despite it involves frequent communication) is because extracting private data from an ML model is infeasible. However, recent studies have shown that retrieving private training data from an ML model is possible. The membership inference attack is an adversarial algorithm designed to recover training data of an ML model (Li et al., 2020; Liang et al., 2018; Shokri et al., 2017). Shokri et al. identified that it is possible to recognize a sample in a private training dataset only by assessing the output of an ML model (Shokri et al., 2017). An over-fitted classifier model tends to yield higher classification confidence for samples of the training data. The authors trained shadow models in order to exploit this divergent behavior of the target model. Followed by the initial work, several research results have identified various techniques to recover private information from a target model (Hisamoto et al., 2020; Salem et al., 2019; Shokri et al., 2017; Yeom et al., 2018). The membership inference attack typically has either a white-box or a black-box assumption (Salem et al., 2019; Yeom et al., 2018). In the black box assumption, the attacker can only feed inputs and observe the resulting prediction vector. On the other hand, attackers can fully access the target model in the white box assumption. It is obvious that the white box assumption is stronger than the black box assumption since the attacker can fully investigate the target model.

FL is more vulnerable to the inference attack than the conventional ML since its training topology divulges parameters during communications. Accordingly, the attack with the white-box setting can be more critical in FL as the attackers can have full access to the parameters and gradients of the target model. Nasr *et* al. designed a membership inference attack against FedAvg algorithm (Nasr et al., 2019). They showed that a malicious device could send a malicious update to make the target model reveal private information more easily. A gradient resulting from the stochastic gradient descent algorithm suggests a direction in which model parameters need to be updated. This direction in gradient is strongly correlated with the corresponding mini-batch of the training data. For these reasons, other attacks have also exploited gradients shared during decentralized training in FedSgd. To this end, deep leakage from gradients is one crucial example where it is possible to recover raw training data from the gradient updates (Zhu et al., 2019). It updates a data sample so that a gradient of the sample can have a smaller distance from a captured gradient. Similar studies have demonstrated that training data can be fully recovered through alternative approaches (Geiping et al., 2020; Zhao et al., 2020).

Existing works use differential privacy to protect the privacy of FL (Agarwal et al., 2018; Geyer et al., 2017; Wei et al., 2020). Although FL with differential privacy is proven to converge, it compromises the classification accuracy of the trained model (Geyer et al., 2017). Therefore, encryption has been used as an alternative solution as well as other security protocols for FL to eliminate the exposure of raw updates to an adversary (Dong et al., 2020; Liu et al., 2019; Phong et al., 2017; Zhang et al., 2020a). However, incorporating a cryptographic scheme incurs computational overhead and could potentially become bottleneck throughout the learning process. In summary, both differential privacy and cryptographic schemes are not sufficient because they either degrade the accuracy or processing speed.

In this paper, we propose a novel FL scheme with Digestive Neural Network (DNN). DNN is a set of sequential convolutional neural networks that have identical input and output dimensions. A mini-batch is transformed within the DNN into another form, hence we denote it as the *digested* mini-batch. Each mobile device has a DNN and a collaborative neural network. The collaborative neural network accepts the digested mini-batch and yields a corresponding prediction vector. Each device collects updates from its collaborative neural network using the digested mini-batch. The DNN is trained on digestive loss, conjunction of distance loss and classification loss. Using the digestive loss, the DNN produces digested mini-batch that has features useful for classification and has the maximum distance from the original mini-batch. The digestive loss is controlled with a threshold value, which can be chosen by the FL service provider to offer various privacy levels. The proposed DNN achieves high classification accuracy and low attack success rate than FL schemes with differential

☆ The preliminary version of this paper was accepted in IEEE International Conference on Communications (ICC'21)

* Corresponding author.

E-mail addresses: hlee115@students.kennesaw.edu (H. Lee), jkim8@keti.re.kr (J. Kim), tpdud1014@hanyang.ac.kr (S. Ahn), r.hussain@innopolis.ru (R. Hussain), chopro@hanyang.ac.kr (S. Cho), json@kennesaw.edu (J. Son).

[1] http://i2s.kennesaw.edu

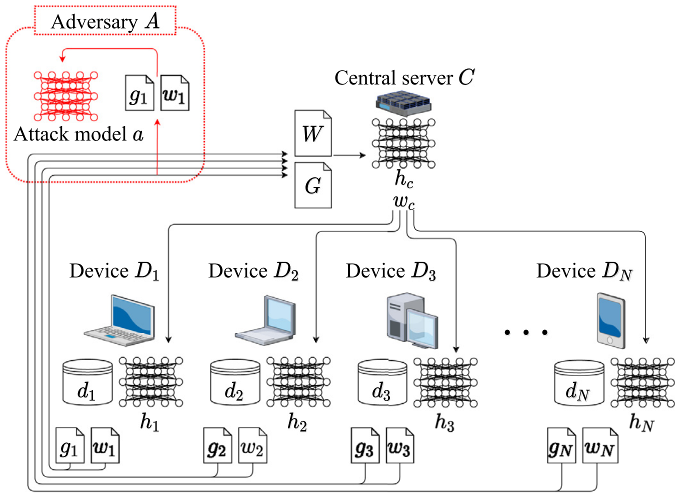


Fig. 1 – Various FL protocols with adversary performing membership inference attack.

privacy. The proposed DNN is applicable to both FedAvg and FedSgd, and is robust against the state-of-the-art membership inference attacks.

The contributions of this paper are summarized as follows:

- A novel DNN is proposed to defend against inference attacks in FL by transforming input data into an unrecognizable, yet trainable data. The proposed scheme has a negligible effect on the accuracy.
- Scalability of the proposed scheme has been demonstrated with experiments on both gradient sharing FL (FedSgd) and parameter sharing FL (FedAvg).
- Convergence stability for the proposed scheme is illustrated.
- Experimental analysis on the train-ability of the digested mini-batch is demonstrated with t-distributed Stochastic Neighbor Embedding (t-SNE) (Maaten and Hinton, 2008).
- To assess the resilience against the membership inference attack, pre-trained models are utilized to quantify the resemblance of the recovered data and original data.

The rest of the paper is organized as follows. Section 2 introduces the backgrounds and preliminaries of this research. Section 3 introduces our proposed scheme, Digestive Neural Network (DNN) with a federated learning scheme. In Section 4, we show the experimental results. Section 5 provides related work. Finally, conclusions are drawn in Section 6.

2. Preliminaries

In this section, we describe the fundamentals of federated learning, inference attacks, and the differential privacy.

2.1. *Federated learning*

The FedAvg (McMahan et al., 2017), a method updating parameters, and FedSgd (Shokri and Shmatikov, 2015), a method updating gradients, are two prevalent FL protocols. Fig. 1 describes the training protocol for both of them. Let D_i be an i-th participating device and d_i be a private dataset of i-th device, where $i \in \{1, 2, \cdots N\}$. We denote FL scheme as a comprehensive terminology for federated learning that includes every D_i and a central server C with a training protocol. Each D_i has its private dataset d_i along with its local model h_i which has identical structure with the central server's model h_c. Each h_i is parameterized by w_i, and gradient of w_i is $g_{i,b}$ which is defined as follows:

$$g_{i,b} = \frac{\partial \mathcal{L}(h_i(w_i, x_{i,b}), y_{i,b})}{\partial x_{i,b}}, \tag{1}$$

where $\mathcal{L}$ is cross-entropy loss and $(x_{i,b}, y_{i,b})$ are b-th mini-batch data samples. We define mini-batch data a set of data samples that satisfy $(x_{i,b}, y_{i,b}) \in d_i$ and $\bigcup_{b=1}^{N/B} (x_{i,b}, y_{i,b}) = d_i$.

In FedAvg, each device trains its local model $\forall (x_{i,b}, y_{i,b}) \in d_i$. After training, device sends w_i from h_i directly to the central server. The central server obtains the aggregated parameter W as follows:

$$W = \frac{1}{N} \sum_{i=1}^{N} w_i,$$

$$w_c \leftarrow W.$$

The central server exchanges w_c with W and distributes W to all devices. Each device replaces its w_i with W and continues the protocol.

In FedSgd protocol, each device sends g_i to the central server. The central server aggregates all gradients to G, and executes gradient descent on the h_c. Mathematically, this can be described as follows:

$$G = \frac{1}{N} \sum_{i=1}^{N} g_{i,b},$$

$$w_c \leftarrow w_c - \gamma G,$$

where γ is a learning rate. The w_c is then distributed back to all devices. Devices replace their w_i with w_c.

The main difference between the two protocols is the type and the frequency of the update. FedAvg performs communication once per a training epoch. Whereas devices participating in FedSgd communicate for every $(x_{i,b}, y_{i,b})$. For theses reasons, FedAvg is often incorporated in FL scheme with wireless communication, while FedSgd is often used for collaborative learning in distributed computing units (Lim et al., 2020).

2.2. *Inference attack*

A successfully trained ML model predicts unseen data with high accuracy using features learned from its training dataset. This implies that the performance of an ML model is determined by its training dataset. Therefore, it is possible to trace back what kind of samples exist in the training data of an ML model based on its performance. This introduces a significant threat to the dataset itself. The adversary can recover samples in a commercially available dataset by analyzing the ML model trained on the dataset. Also, if the dataset includes private information, then the adversary may divulge the private information from the dataset. As the title suggests, the membership inference attack is an adversarial algorithm that analyzes a target ML to discover samples of the training dataset.

There are two assumptions on the target model for membership inference attack; the black-box and the wite-box. In the black-box setting, the adversary cannot have access to the target model. The adversary is only allowed to observe the output prediction vector from the corresponding input sample. Based on the model's behavior, the adversary analyzes the distributions of the dataset. Whereas the white-box setting assumes that the adversary has full access to the target model. The adversary has access to a fully functioning target model and can compute gradient or inference. Therefore, the white-box attack is more destructive than the black-box attack since the assumption provides more knowledge to the adversary.

Pivotal research conducted on membership inference attack introduced an effective performance on discriminating samples that are in the training dataset (Agarwal et al., 2018). The authors of the research developed a membership inference attack model based on neural networks. To mimic the target model's behavior, the authors trained a number of shadow models. Using the shadow model, the authors trained the attack model that determines whether a given sample is included in the target's training dataset. The ML model shows more prediction confidence on samples that are in the training dataset.

The red box on the left top of Fig. 1 depicts an adversary conducting the membership inference attack on the FL scheme. The adversary A is targeting the D_1 and eavesdrops on the update from the D_1. The A eavesdrops w_1 or g_1 depending on the FedAvg or FedSgd protocol respectively. The adversary has its attack model a that is trained on either w_1 or g_1. The goal of the a can be defined mathematically as follows.

$$a(x) = \begin{cases} 1 & \text{if } x \in d_i \\ 0 & \text{otherwise} \end{cases}$$

2.3. *Differential privacy*

Constructing a dataset involves data sanitation. That is, the data should not include any information that leads to the leakage of one's private information. However, with a sophisticated set of queries, it is possible to extract private information from the data. An example can be querying the hospital records. Although names of patients are removed from the dataset, it is possible to identify particular individuals by using multiple datasets. If a patient living in a particular area has a unique pattern of disease, using other data, it is possible to retrieve the individual's information. This attack is known as the linkage attack.

To prevent privacy abuse, differential privacy has been introduced. The differential privacy hides the contribution of a particular sample on a query. Dwork *et* al. proposed an ϵ-differential privacy (Dwork, 2006) to assure the confidentiality of a statistical dataset based on how much contribution a single data point has on a query. Formally, ϵ-differential privacy is defined as follows. Let D_1 and D_2 are two datasets that have one sample difference. Let $\mathcal{A}$ is a randomized algorithm and let $\mathcal{S}$ is a subset of the output of $\mathcal{A}$. The algorithm $\mathcal{A}$ provides ϵ-differential privacy under the following equation:

$$Pr[\mathcal{A}(D_1) \in S] \leq exp(\epsilon) \cdot Pr[\mathcal{A}(D_2) \in S]$$

Thus, the dataset provider can inject ϵ amount of noise into the dataset to achieve differential privacy. The differential privacy is highly scalable as the definition can be utilized by injecting noise into the parameters of ML models.

3. Proposed digestive neural network based secure federated learning against inference attack

3.1. *Baseline overview*

Fig. 2 illustrates the proposed DNN scheme with FL. The scheme consists of a single central server and multiple mobile devices. Each device has a DNN and a collaborative neural network, while the central server only has the collaborative neural network. The collaborative neural network is a classifier; the architecture accepts multi-dimensional data by convolutional neural networks and yields a one-dimensional prediction vector. Devices and the central server must have a homogeneous architecture of collaborative neural networks. This is because only the parameters of collaborative neural networks are communicated during the protocol. The DNN accepts multi-dimensional data and produces multi-dimensional data with identical dimensions with the input. The parameters of the DNN are not shared. More precisely, the devices do not reveal the DNN to the communication channel.

The arrow signs in Fig. 2 depict our proposed training protocol for DNN. A mini-batch of private training data sequentially passes through the DNN and the collaborative neural network. The DNN *digests* features of the given mini-batch into completely different domains to remove private information of the data. The digested mini-batch needs to exclude original information and must contain features useful for high classification accuracy. The collaborative network receives digested mini-batch and optimizes its parameter to improve the classification accuracy of the digested data. To increase the prediction accuracy of the collaborative neural networks, both DNN and collaborative networks need to be trained cooperatively.

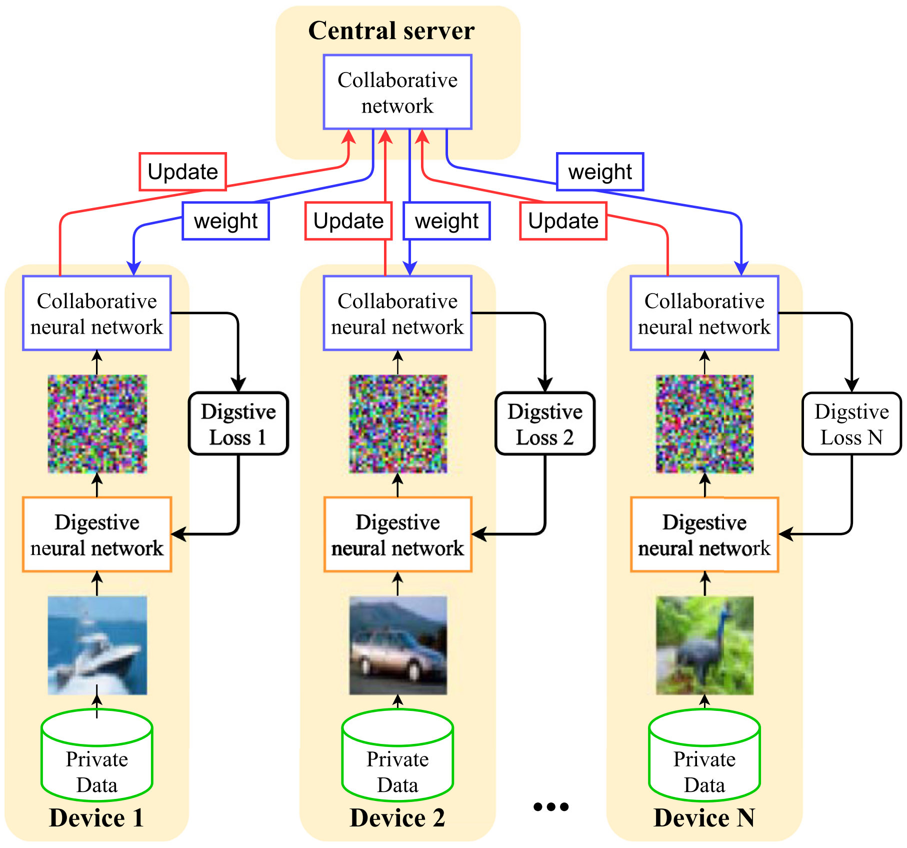


Fig. 2 – Overview of FL procedures with the proposed scheme.

Each device prepares an update to send to the central server, using its collaborative neural network. The update is generated only using digested mini-batch and collaborative neural networks. After every device finishes sending its update, the server distributes new parameters obtained from the aggregation of all the updates. Mobile devices train their DNN based on the updated collaborative neural network. Training of DNN is determined by the digestive loss, a conjunction of the classification loss, and the distance loss controlled by a threshold. The goal of the digestive loss is to maximize the distance between the original mini-batch and the digested mini-batch while including crucial information within the digested mini-batch for high classification accuracy. The classification loss is a cross-entropy loss and the distance loss focuses to increase the distance between input mini-batch and digested mini-batch. More specifically, the distance loss increases the distance between gradients of mini-batch and digested mini-batch for FedSgd, while it increases the pixel-wise distance of mini-batch and digested mini-batch for FedAvg protocol. After updating DNN, each device proceeds to the next training round. It is worth mentioning that the proposed DNN applies to both FedAvg and FedSgd protocols. A detailed description of each training protocol of the proposed scheme is illustrated in a subsequent section.

The adversary in FedSgd and FedAvg utilizes gradient update and parameter update from the target device. Furthermore, the adversary in FedSgd eavesdrops gradient update to recover the mini-batch. To this end, the proposed FL scheme with DNN prevents the attack since the target device generates the gradient from the digested mini-batch. This entails the adversary to recover information that is irrelevant to the original mini-batch. The adversary in FedAvg is assumed to already possess a portion of the target device's private training data. Therefore, the adversary trains an attack model that can identify a given sample's membership of the target device's dataset using eavesdropped parameters and the known portion of the private training data. The proposed FL scheme with DNN mitigates the attack since the parameter update from the target device is trained on the digested mini-batch. This makes the adversary's knowledge of private data totally useless. This also entails the adversary to train the attack model on a completely erroneous environment and mitigates the attack that is aimed at properly identifying the private training data.

3.2. *Training protocol: FedSgd*

Fig. 3 depicts the procedure of a communication round in the training protocol of the proposed scheme with FedSgd protocol. A communication round begins with training of the collaborative neural network $h_{\mathcal{C},i}$ followed by training of the DNN $h_{\mathcal{D},i}$. Let S be a central server and I be a set of all devices participating in the training. In a communication round t, device $i \in I$ gets a mini-batch $(x_{i,t}, y_{i,t})$ from its private dataset. We denote mini-batch after passing the digestive network as $x'_{i,t}$, which is $x'_{i,t} = h_{\mathcal{D},i}(x_{i,t})$. In training collaborative neural net-

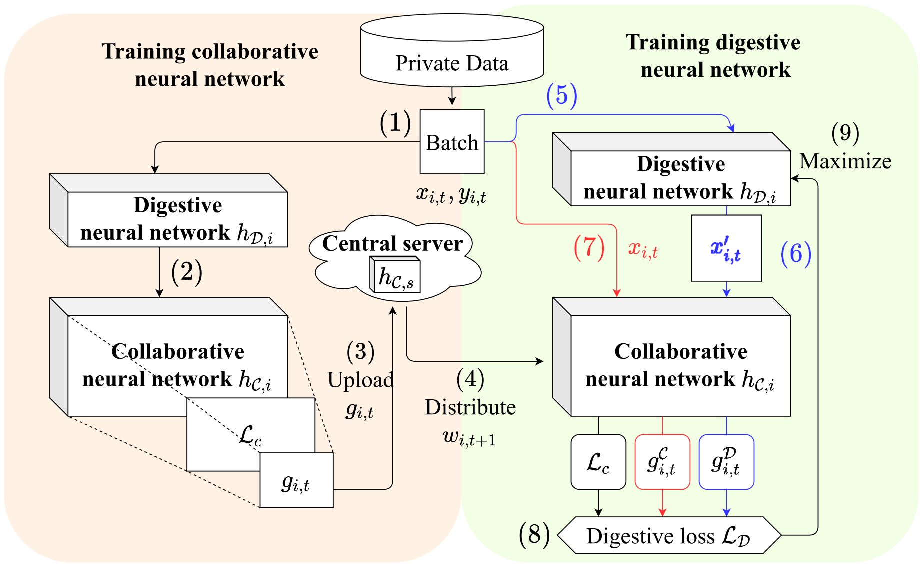

Training collaborative neural network
Private Data
Training digestive neural network
(5)
(1)
Batch
(9) Maximize
Digestive neural network $h_{\mathcal{D},i}$
$x_{i,t}, y_{i,t}$
Digestive neural network $h_{\mathcal{D},i}$
(2)
Central server
$h_{\mathcal{C},s}$
(7)
$x_{i,t}$
$x'_{i,t}$
(6)
Collaborative neural network $h_{\mathcal{C},i}$
(3) Upload $g_{i,t}$
(4) Distribute $w_{i,t+1}$
Collaborative neural network $h_{\mathcal{C},i}$
$\mathcal{L}_c$
$g_{i,t}$
$\mathcal{L}_c$
$g^{\mathcal{C}}_{i,t}$
$g^{\mathcal{D}}_{i,t}$
(8)
Digestive loss $\mathcal{L}_{\mathcal{D}}$

Fig. 3 – Training protocol of the proposed scheme on FedSgd.

work, the device obtains gradient $g_{i,t}$ using classification loss $\mathcal{L}_c$ from the collaborative neural network, where $g_{i,t}$ is defined as follows:

$$g_{i,t} = \frac{\partial \mathcal{L}_c\left(h_{\mathcal{C},i}\left(x'_{i,t}\right)\right)}{\partial w_{i,t}}. \tag{2}$$

The gradient $g_{i,t}$ is an update for the server S that distributes the new parameter w_{t+1} after receiving and accumulating $g_{i,t}$ from all the participating devices. The server conducts a step in gradient descent with a learning rate τ as follows:

$$w_{t+1} = w_t - \tau \frac{1}{|I|} \sum_{j=1}^{I} g_{i,t}. \tag{3}$$

Once mobile devices receive the w_{t+1} for their collaborative neural networks, each device trains its DNN. The training objective of the DNN is to digest the data so that the attack algorithm fails to recover the original data. Simultaneously, $x'_{i,t}$ needs to contain recognizable features for the collaborative neural network to make accurate predictions. To achieve both requirements, we present the loss $\mathcal{L}_{\mathcal{D}}$ defined as follows:

$$\mathcal{L}_{\mathcal{D}} = \mathcal{L}_c + \max\left(\alpha - \mathcal{L}_d, 0\right), \tag{4}$$

where $\mathcal{L}_c$ is a classification loss and $\mathcal{L}_d$ is a distance loss. $\mathcal{L}_c$ is responsible for assisting the collaborative neural network to increase the classification accuracy, while $\mathcal{L}_d$ updates DNN to discourage the attack. We define a threshold α to properly regulate $\mathcal{L}_d$ as the static influence of $\mathcal{L}_d$ impairs the convergence of the entire FL scheme. $\mathcal{L}_d$ requires $g^{\mathcal{C}}_{i,t}$, $g^{\mathcal{D}}_{i,t}$ and $\mathcal{L}_c$, where $g^{\mathcal{C}}_{i,t}$, $g^{\mathcal{D}}_{i,t}$ are the gradients of the collaborative neural network on $x_{i,t}$ and $x'_{i,t}$, respectively. Gradients and losses are derived from the same mini-batch that were used to train the collaborative network. Mathematically, two gradients can be represented as follows:

$$g^{\mathcal{C}}_{i,t} = \frac{\partial \mathcal{L}_c(h_{\mathcal{C},i}(w_{i,t}, x_{i,t}))}{\partial w_{i,t+1}}, \tag{5}$$

$$g^{\mathcal{D}}_{i,t} = \frac{\partial \mathcal{L}_c\left(h_{\mathcal{C}i}\left(w_{i,t} x'_{i,t}\right)\right)}{\partial w_{i,t+1}}. \tag{6}$$

The digestive loss $\mathcal{L}_{\mathcal{D}}$ only updates the DNN. Furthermore, training of the digestive network does not affect the collaborative neural network and vise versa. After every device updates its DNNs, mobile devices collect the next mini-batch for the subsequent communication round.

3.3. *Attack scenario: FedSgd*

We suppose an adversary defined in (Geiping et al., 2020; Zhao et al., 2020; Zhu et al., 2019) for attacking the FedSgd protocol. The adversary eavesdrops on the communication channel and acquires $w_{i,t}$ and $g_{j \in I,t}$. It is assumed that the adversary does not possess any prior knowledge of each device's private dataset but does have enough computing resources to fabricate a data sample and derive a gradient using the acquired weights $w_{i,t}$. When the adversary eavesdrops $g_{j,t}$ and $w_{i,t+1}$, it generates a random data sample x_{adv}, y_{adv} and starts to minimize the distance of gradient of x_{adv} with the eavesdropped $g_{j,t}$. The adversary updates the random data sample to minimize the optimization goals:

$$\underset{x_{adv}}{\arg\min} \left\| \frac{\partial \mathcal{L}_c(y_{adv}, h_{\mathcal{C},adv}(w_{i,t}, x_{adv}))}{\partial w_{i,t}} - g_{j,t} \right\|^2. \tag{7}$$

That is, the attacker updates x_{adv} and y_{adv} so that the gradient of x_{adv} can be similar to the $g_{i,t}$. Iterating the process alters x_{adv} to be analogous to $x_{i,t}$.

The attack model presented in (Geiping et al., 2020) minimizes cosine similarity between two gradients. The intuition behind the cosine similarity is that the gradient suggests a direction to optimize the neural network with respect to the mini-batch. Finding a data sample that suggests the same direction produces the original data. A detailed formulation of the direction in optimization problem is defined as follows, where $\langle x, y\rangle / ||x|| \cdot ||y||$ is cosine similarity of x and y:

$$\arg\min_{x\in[0,1]^n} 1 - \frac{\left\langle \frac{\partial \mathcal{L}_C\left(y_{adv}, h_{\mathcal{C},adv}\left(w_{i,t}, x_{adv}\right)\right)}{\partial w_{i,t}}, g_{j,t} \right\rangle}{\left\| \frac{\partial \mathcal{L}_C\left(y_{adv}, h_{\mathcal{C},adv}\left(w_{i,t}, x_{adv}\right)\right)}{\partial w_{i,t}} \right\| \cdot \left\| g_{j,t} \right\|}. \tag{8}$$

In the simulations, we experimented with both attackers to examine the effectiveness of the two attacks and compare them with our proposed defense mechanisms.

3.4. *Digestive loss: FedSgd*

The digestive loss consists of $\mathcal{L}_d$ and $\mathcal{L}_c$, which are responsible for improving privacy and accuracy, respectively. The design of $\mathcal{L}_d$ starts from scrutinizing the attack scenario. A gradient-based attack minimizes the distance between a target gradient and a gradient from the dummy data. The DNN originated from an intuition of reversing the attack process using a neural network architecture. To this end, the optimization goal of the DNN is to increase the distance of the two gradients. The update from device i is $g_{i,t}$, which is defined as in Eq. 2 and it is the gradient of $x'_{i,t}$ with respect to the $h_{\mathcal{C},i}$. Thus, when an adversary launches a successful attack using $g_{i,t}$, it will recover $x'_{i,t}$. To prevent the attack, we need to convert $g_{i,t}$ to pose difficulty in processing the attack algorithm. Moreover, even if the attack algorithm successfully recovers the $x'_{i,t}$, recovering $x'_{i,t}$ is of no use if $x_{i,t} \neq x'_{i,t}$. In other words, it does not give any advantage to the adversary.

Thus, we propose pushing $\frac{\partial h_{\mathcal{C},i}(x'_{i,t})}{\partial w_{\mathcal{C},i}}$ far from $\frac{\partial h_{\mathcal{C},i}(x_{i,t})}{\partial w_{\mathcal{C},i}}$. This will introduce difficulty to the adversarial algorithm as pushing the gradient increases the distance that the adversarial algorithm has to minimize. Accordingly, pushing makes $x'_{i,t} \neq x_{i,t}$ as two data will produce two distant gradients. This way, the adversary will not be able to recover private information of $x_{i,t}$ from $x'_{i,t}$ as it is completely different from $x_{i,t}$. To achieve this, we propose an optimization goal of $h_{\mathcal{D},i}$ as follows:

$$\frac{\partial h_{\mathcal{C},i}(h_{\mathcal{D},i}(x_{i,t}))}{\partial w_{\mathcal{C},i}} = \max_{x_{i,t}} \left(dist\left(g_{i,t}, \frac{\partial h_{\mathcal{C},i}(w_{\mathcal{C},i}, x_{i,t})}{\partial w_{\mathcal{C},i}} \right) \right). \tag{9}$$

The $x'_{i,t}$ generates gradient $g_{i,t}$ that has significant difference with $\frac{\partial h_{\mathcal{C},i}(w_{\mathcal{C},i}, x_{i,t})}{\partial w_{\mathcal{C},i}}$. To optimize $h_{\mathcal{D},i}$ for successful generation of $x'_{i,t}$, the distance loss that contributes to update $h_{\mathcal{D},i}$, is defined as follows:

$$\mathcal{L}_d = -\frac{dist\left(\frac{\partial \mathcal{L}_C\left(h_{\mathcal{C}}\left(w_{i,t+1} x_{i,t}\right)\right)}{\partial w_{i,t+1}}, \frac{\partial \mathcal{L}_C\left(h_{\mathcal{C}}\left(w_{i,t+1}, x'_{i,t}\right)\right)}{\partial w_{i,t+1}} \right)}{|h_{\mathcal{C}}|}. \tag{10}$$

The distance loss is defined as an average distance between two gradients normalized by the size of the $h_{\mathcal{C},i}$. It updates the $h_{\mathcal{D},i}$ so that the gradient of $x'_{i,t}$ can be distant from the gradient of the $x_{i,t}$. The difference between the two gradients is determined by various distance metrics, including cosine similarity, Minkowski distance with $p = 1$ and $p = 2$. We denote the Minkowski distance with $p = 2$, or Euclidean distance as L2 and Minkowski distance with $p = 1$ as L1, as they are L_p norm of differences (Friedrich, 1910).

In the proposed FL scheme, the constant presence of $\mathcal{L}_d$ impairs the global convergence of all mobile devices. Thus we introduce a threshold α to control the influence of $\mathcal{L}_d$ over the scheme. As in Eq. (4), $\mathcal{L}_d$ becomes 0 when it exceeds α. This keeps distance of gradients of $x'_{i,t}$ and gradient of $x_{i,t}$ to stay farther than α. Unlike $\mathcal{L}_d$, $\mathcal{L}_c$ has a constant influence on the training of DNN to maintain high accuracy.

3.5. *Training protocol: FedAvg*

Devices participating in the FedAvg algorithm upload weight parameter of their local neural networks after training for an epoch. The central server distributes averaged weights to all the devices. The training protocol of the proposed scheme includes training of both collaborative neural network and digestive neural network without violating the underlying FedAvg algorithm. Fig. 4 depicts the training protocol of i-th device participating in the FL of the proposed scheme for FedAvg algorithm (McMahan et al., 2017). The training is divided into two phases. In training a collaborative neural network, each device freezes the digestive neural network and trains only the collaborative neural network. Whereas in training the digestive neural network, each device freezes the collaborative neural network and trains only the digestive neural network.

The initial phase is training collaborative neural network $h_{\mathcal{C},i}$. First each device inputs its private data x_i, y_i into its own digestive neural network $h_{\mathcal{D},i}$ and collaborative neural network $h_{\mathcal{C},i}$. The device then derives only updates from its $h_{\mathcal{C},i}$ using cross-entropy loss $\mathcal{L}_c$ from the outputs. After training for an iteration on whole data, the device sends its trained parameters of $w_{i,t}$ to the central server. As the central server receives all parameters from all the devices, the server averages parameters and distributes the newly updated parameter.

The second phase is training the digestive neural network $h_{\mathcal{D},i}$. Similar to the proposed scheme on FedSgd, the digestive neural network needs to transform the original mini-batch x_i into a totally divergent mini-batch, however, it yet needs to contain proper features that can assist the classification performed in the collaborative neural network. After receiving a new parameter from the server, the device freezes the parameter of $h_{\mathcal{C},i}$. For each mini-batch x_i collected from the private data, the device obtains digested data x'_i by passing x_i through the digestive neural network $h_{\mathcal{D},i}$. Using x_i and x'_i, the device calculates the distance loss $\mathcal{L}_D$ where $\mathcal{L}_d$ is a batch-wise mean distance between x_i and x'_i. This will maximize the pixel-wise distance between x_i and x'_i. Also, the device sends its digested data x'_i to the collaborative neural network and obtains cross-entropy loss $\mathcal{L}_c$. Two losses are aggregated into $\mathcal{L}_{\mathcal{D}}$ and used for updating the digestive neural network.

$$\mathcal{L}_{\mathcal{D}} = \mathcal{L}_c + \max(\alpha - \mathcal{L}_d, 0), \tag{11}$$

The digestive loss is controlled by a threshold value α. If the distance loss is greater than the threshold value, the distance loss is deactivated. After deactivation, the DNN minimizes the classification loss only.

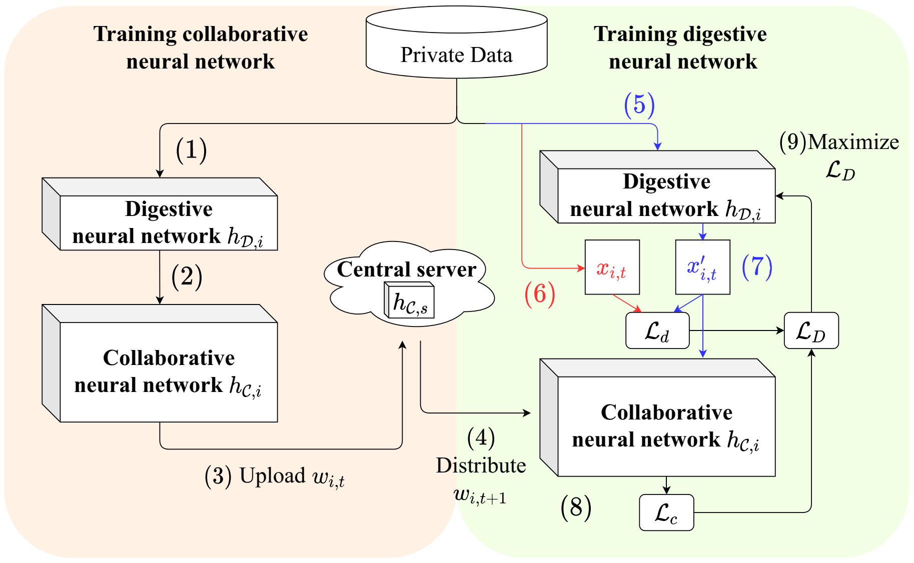

Training collaborative neural network
Private Data
Training digestive neural network
(1)
(5)
(9)Maximize $\mathcal{L}_D$
Digestive neural network $h_{\mathcal{D},i}$
(2)
Central server
$h_{\mathcal{C},s}$
$x_{i,t}$
$x'_{i,t}$
(7)
(6)
$\mathcal{L}_d$
$\mathcal{L}_D$
Collaborative neural network $h_{\mathcal{C},i}$
(3) Upload $w_{i,t}$
(4) Distribute $w_{i,t+1}$
(8)
$\mathcal{L}_c$

Fig. 4 – Training protocol of the proposed scheme on FedAvg algorithm.

3.6. *Attack scenario: FedAvg*

We define the adversary attacks on the FedAvg protocol using the *passive global attacker* in (Nasr et al., 2019). The goal of the passive global attacker is to build an attack model that can identify whether a data sample is included in the target device's training dataset. The attack model is a binary classifier that predicts the membership of the input sample. The attacker already has a portion of the target's data and the attacker has enough computing resource to simulate a neural network with the target's parameters. In the designated training rounds, the attacker eavesdrops on the parameter update from the target device. It saves the parameter to its local storage until the FL protocol is completely executed. After acquiring parameters from all the training rounds, the attacker trains its attack model using the target's training data that the attacker already has. Then the attacker extracts gradient, feature map, and output from neurons and loss values by simulating the target's neural network. The attack model is trained in a supervised manner. Let A is an attack model, d_t is a target device's private dataset, and let $(x_t, y_t) \in d_t$ is a member data of d_t and $(x', y') \notin d_t$ is non-member data. The adversary is assumed to have a partial dataset $d_{a,t} \subset d_t$ that satisfies $d_{a,t} \neq d_t$. The adversary aims to infer the unknown portion of the dataset $d_{u,t} = d^{\complement}_{a,t}$. We also assume the target model with target's weight $h_t(w_t)$. Then the attacker forms the attack dataset as follows:

$$d_{atk} = \{(x_i, y_i, z) \forall (x_i, y_i) \in d_{a,t}\} \cup \{\left(x_j, y_j, z\right)\}, \tag{12}$$

$$z = \begin{cases} 1 & \text{if } (x_k, y_k) \in d_{a,t} \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

The attacker uses d_{atk} for training the attack model using gradient, loss value, feature maps and output of the target model as follows:

$$o_k = h_t(w_t, x_k), \tag{14}$$

$$l_k = \mathcal{L}(h_t(w_t, x_k), y_k), \tag{15}$$

$$g_k = \frac{\partial \mathcal{L}(h_t(w_t, x_k), y_k)}{\partial x_k}, \tag{16}$$

$$f_k = \{h_{n,t}(w_t, x_k) \forall h_{n,t} \in h_t\}. \tag{17}$$

where $h_{n,t}$ is the k-th layer of the target model. Thus $h_{n,t}(x_k, y_k)$ represents the feature map of k-th layer. The attacker trains the attack model to build the following binary classifier.

$$A(x_k, y_k, o_k, l_k, g_k, f_k) = \begin{cases} 1 & \text{if } (x_k, y_k) \in d_t \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

That is, the attacker tries to optimize A to predict the membership of (x_k, y_k).

3.7. *Digestive loss: FedAvg*

Similar to the FedSgd, the digestive loss consists of distance loss $\mathcal{L}_\lceil$ and classification loss $\mathcal{L}_c$. In FedAvg, The $\mathcal{L}_d$ directly derives the mathematical distance of a data sample x and the corresponding digested sample x'. The $\mathcal{L}_d$ attempts to increase pixel-wise distance between the x and x'. By iteratively optimizing the DNN using $\mathcal{L}_d$ converts DNN so that DNN delivers x' to the collaborative neural network. Mathematically, it can be defined as follows:

$$\mathcal{L}_d = -\frac{dist(x, x')}{|x|}. \tag{19}$$

Table 1 – Simulation configurations for digestive network.

Type 1		Type 2		Type 3		Type 4		Type 5		Type 6	
K	S	K	S	K	S	K	S	K	S	K	S
32	1	32	2	32	1	32	1	32	2	32	2
32	1	32	2	16	1	64	1	16	2	64	2
32	1	32	1	16	1	64	1	16	1	64	1
32	1	32	0	16	1	64	1	16	0	64	0
32	1	32	0	32	1	32	1	32	0	32	0

K: Number of kernels, S: Strides

The difference between the two samples is determined by various distance metrics, including cosine similarity and Minkowski distance with $p = 1$ and $p = 2$. We will denote Minkowski distance with $p = 2$, or Euclidean distance as L2, and Minkowski distance with $p = 1$ as L1, as they are L_p norm of the differences (Friedrich, 1910).

The goal of the attacker of FedAvg protocol is to infer the portion of the target's private data using another portion of the data that the attacker already has. Therefore, DNN delivers x' that is totally different from the x for training the collaborative neural network. Thus, the attacker will receive the parameters of the collaborative neural network that was trained on completely different data x' than what the attacker currently has. This fails the training of the attack model as the data the attacker has, is different from the x'.

4. Experiment results and evaluation

We simulated the proposed scheme in various conditions and quantified the performance of the DNN. The simulation was conducted with different architectures of DNNs including three different distance metrics. The detailed configuration of DNN is described in Table 1. The DNN includes five residual blocks. Each block includes three convolutional layers with a residual path. Each row in Table 1 is a configuration for the residual block. The number of kernels in convolutional layers is determined by K, and S determines to downsample and upsampling. Columns of S indicate strides for convolutional layers. S = 2 means that the corresponding block includes a convolutional layer that downsamples the input mini-batches dimensions in half. S = 0 means the output of the corresponding block will be double of the size of the input mini-batch. S = 1 indicates that the output dimension is identical to the input dimension.

The experiment involves six different types of architectures for DNNs. Each type is selected to measure the varying performance of the proposed scheme according to the architecture of the DNNs. All residual blocks in type 1 have identical kernel sizes as well as identical feature map dimensions. Residual blocks in type 3, 4, 5, and 6 have varying kernel sizes while type 2, 5, and 6 include blocks that reduce and increases feature maps.

We simulated our attack on both FedSgd and FedAvg protocols. For FedSgd, we simulated FL with $|I| = 5$ using aforementioned multiple DNNs and resnet-20 as collaborative neural networks (He et al., 2016). Every FL scheme is trained for 300 epochs with batch sizes of 32 and a learning rate of $1e - 4$. After training multiple FL schemes, we simulated the attack algorithm suggested in the previous section. The attack simulation runs the attack algorithm to recover 100 private data instances.

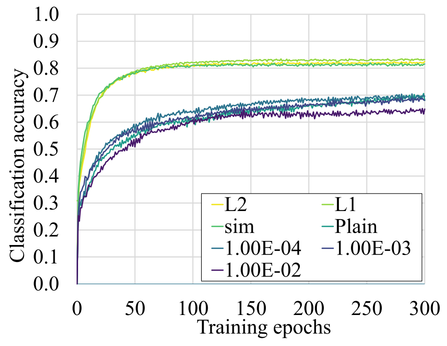
For the FedAvg algorithm, we simulated FL with $|I| = 3$ and randomly selected 30,000 private data instances for each device from the CIFAR-100 dataset with allowing duplicates. The simulation incorporates same DNN, but we used AlexNet (Krizhevsky et al., 2012) as our collaborative neural network architecture. Every FL scheme is trained for 300 epochs with a batch size of 64 and a learning rate of $1e - 3$. The training conditions and hyperparameters are selected to be in compliance with the attack algorithm. Moreover, we implemented the *passive local attacker* for FedAvg introduced in (Melis et al., 2019). Every FL scheme is implemented by Pytorch on workstation i7 with four RTX 2080 TI GPUs.

In this section, we also evaluate the performance of the FL by comparing accuracy and attack success rates. The accuracy denotes model performance on the test data whereas the attack success rate indicates how the model is robust to an attack. As aforementioned, we simulated the attack algorithm for both FedSgd and FedAvg protocol. For the FedSgd protocol, the attack success rate is measured by comparing the similarity of original data and the data resulting from 100 attack simulations. The similarity is measured in terms of accuracy predicted by a pre-trained classifier that can classify the original dataset with high accuracy. The high prediction accuracy of a pre-trained classifier means that the attack algorithm successfully reconstructed data that contains similar visual features to the original data. Thus high attack success rate means the model is not defending well against the attack. The low attack success rate, on the other hand, shows that the attack simulation has failed to generate a data sample that has visually analogous features. For the FedAvg algorithm, we simulated the passive local attacker. Furthermore, the attack model is a binary classifier that predicts the membership of the sample, and the attack success rate is directly obtained by the accuracy of the attack model. The lower accuracy of the attack algorithm indicates that the attack model could not determine whether a given sample is included in the private training dataset. On the other hand, a higher attack success rate indicates that the attack model is certain whether a given sample is a member of the training data.

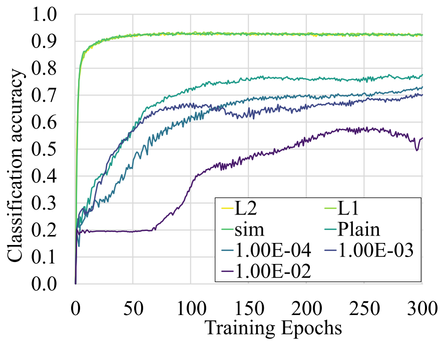
For both FedSgd and FedAvg, we trained the models with differential privacy. The differential privacy model injects a predefined amount of noise into the updates. For FedAvg, the noise is added to the parameter update while for the FedSgd model, noise is added to the gradient of the data. We trained the differential privacy models for the same training epochs and same batch size to make an exact and fair comparison with the proposed schemes. Also, we simulated attacks on differentially private models to compare the attack success rate for Differential private models and the proposed DNN.

4.1. *Performance analysis: accuracy and attack success rate on FedSgd*

We first compare the accuracy and attack success rate of the FL with differential privacy on the FedSgd protocol. Table 2 shows the accuracy and attack success rate of plain and vari-



(a) CIFAR10



(b) SVHN

Fig. 5 – Test accuracy per epoch for plain and differential privacy, and proposed scheme with type 1 digestive neural networks on CIFAR10 and SVHN dataset.

Table 2 – Accuracy and attack success rate of plain and differential privacy models.

Scheme	CIFAR 10		SVHN	
	Accuracy	Attack success rate	Accuracy	Attack success rate
Plain model	70.620	19.0	77.631	20.0
δ^2=1e-4	69.620	18.0	73.087	17.0
δ^2=1e-3	68.130	17.0	70.200	15.0
δ^2=1e-2	64.280	17.0	54.141	26.0

Table 3 – Performance of proposed scheme with L2 distance metric.

Digestive type	α	Accuracy	attack success rate
Type 1	1e-1	82.448	10.0
Type 2	1e-1	78.320	11.0
Type 3	1e-1	82.416	11.0
Type 4	1e-1	85.794	10.0
Type 5	1e-1	78.080	9.0
Type 6	1e-1	77.894	9.0
Type 1	1e-2	81.476	8.0
Type 2	1e-2	77.262	8.0
Type 3	1e-2	83.704	10.0
Type 4	1e-2	82.728	7.0
Type 5	1e-2	78.682	8.0
Type 6	1e-2	77.740	9.0
Type 1	1e-3	83.018	14.0
Type 2	1e-3	78.218	10.0
Type 3	1e-3	83.334	8.0
Type 4	1e-3	82.778	9.0
Type 5	1e-3	78.098	11.0
Type 6	1e-3	77.248	11.0
Type 1	1e-4	81.960	12.0
Type 2	1e-4	78.914	15.0
Type 3	1e-4	82.934	12.0
Type 4	1e-4	82.438	13.0
Type 5	1e-4	77.822	17.0
Type 6	1e-4	78.240	17.0

ous differential privacy schemes on the CIFAR-10 and SVHN datasets. The plain scheme (with no differential privacy) is an FL scheme without any defensive mechanisms whereas differential privacy schemes use an additive Gaussian noise with variance δ^2 to ensure data privacy. As differential privacy schemes add noise to the gradient updates, the accuracy decreases as the level of noise increases. Instead, the attack success rate decreases as the noise level increases, meaning that reconstruction of data through attack simulation was unsuccessful as noise level increases.

Fig. 5 illustrates the performance of each scheme over training epochs. We compared plain scheme, differential privacy scheme, and three proposed schemes with distance metrics of L1, L2, and cosine similarity. All proposed schemes in the graph had type 1 DNN and a threshold of α=1e-1. The graph shows that the proposed schemes, regardless of different distance metrics, achieved high accuracy than the rest of the schemes. This is because the DNN is not only responsible for the digestion of the data but also contributes to the improvement in accuracy. Also, the proposed scheme showed fast convergence on both CIFAR-10 and SVHN datasets, while plain and differential privacy schemes showed slow convergence. The graph also shows the downside of differential privacy schemes which is the drop in the accuracy of the differential privacy schemes as compared to the plain model.

Table 3 shows the accuracy and attack success rate of the proposed scheme, including various DNNs. The simulation is conducted to analyze the effects of various DNN types introduced in Table 1 as well as various α values on FedSgd protocol. All schemes in the table had L2 as a distance metric.

Comparing Tables 2 and 3, it is clear that the proposed scheme achieved higher accuracy and lower attack success rate than the differential privacy schemes. Moreover, the proposed scheme had 7% to 15% higher accuracy than the plain scheme. This is because the digestive neural network not only contributes to the digestion of the input data but also contributes to reducing the classification loss. Also, the proposed scheme had a lower attack success rate, the attack success rate of 10.0 indicates that the reconstructed data do not have any feature similar to the original data. Furthermore, the pre-

Table 4 – Accuracy and attack success rate comparisons over various α values on CIFAR-10 dataset.

Digestive type	α	L1		L2		sim	
		Acc	Att acc	Acc	Att acc	Acc	Att acc
Type 1	1e-1	83.158	16.0	82.448	10.0	81.532	8.0
Type 1	1e-2	81.064	13.0	81.476	8.0	81.672	9.0
Type 1	1e-3	81.002	13.0	83.018	14.0	81.716	13.0
Type 1	1e-4	81.060	9.0	81.960	12.0	82.912	15.0
Type 2	1e-1	79.240	4.0	78.320	11.0	78.604	14.0
Type 2	1e-2	78.606	14.0	77.262	8.0	78.834	12.0
Type 2	1e-3	78.438	6.0	78.218	10.0	78.034	4.0
Type 2	1e-4	78.594	11.0	78.914	15.0	78.418	14.0
Type 4	1e-1	82.218	10.0	85.794	10.0	82.808	8.0
Type 4	1e-2	82.724	9.0	82.728	7.0	82.636	7.0
Type 4	1e-3	82.756	10.0	82.778	9.0	81.326	11.0
Type 4	1e-4	81.950	6.0	82.438	13.0	82.486	13.0

trained classifier for measuring attack success rate makes a random guess to the reconstructed data.

Table 3 implies that the types of architecture and threshold α determines both accuracy and attack success rate. The reduction of α correlates with the attack success rate. This is because the α determines the training of $\mathcal{L}_d$. A small value of α puts less burden on models than high α where in case of higher α, the models have to maintain $\mathcal{L}_d$ up to α. After $\mathcal{L}_d$ reaches α, models train DNN only to minimize the classification error. Thus, models with smaller α achieve higher accuracy as they can train their DNN for assisting classification more pervasively than the schemes with high α. Conversely, the models with high α need to contribute themselves more to maintain $\mathcal{L}_d$ up to α. Although this degrades the classification accuracy, it makes the scheme more robust to the inference attack on FedSgd.

The types of architecture also have a varying impact on the accuracy and attack success rate of the model. The reduction of dimensions has a negative effect on accuracy. Simulation of type 2 only shows degradation in classification accuracy, but not in the attack success rate. On the other hand, type 4, an architecture with larger kernels showed a relatively lower attack success rate than other models while not showing a decrease in the classification accuracy. Effects resulting from a reduction in dimensions and large kernels are independent as simulations of type 6 show effects from both configurations.

From the observation, we have narrowed down to conduct another simulation that only includes type 1, 2, and 4 for the rest of the experiments. The selected types show distinctive effects than other types. Table 4 shows the classification accuracy and the attack success rate of the proposed scheme with various types, α, and distance metrics. We can see that types and α show similar effects even when the distance metrics are different. Also, effects from α and effects of architecture types are independently affecting the accuracy. The accuracy degradation resulting from α is seldom observable in the proposed schemes with type 4. Since type 4 has wider kernel sizes, the accuracy degradation from the smaller α has been counterbalanced by the architecture. Table 5 shows an identical experiment conducted on SVHN dataset. Unlike CIFAR-10, effects resulting from using type 4 have not been observed. Moreover, the attack success rate of the type 4 digestive neural network on the SVHN dataset is almost similar to the proposed schemes with the type of digestive neural networks.

4.2. *Performance analysis: accuracy and attack success rate on FedAvg*

Table 6 shows the performance and the attack success rate of plain FedAvg protocol and the attack success rate. The absolute accuracy of the plain FedAvg is low since the neural network for the FL is AlexNet. The Alexnet without batch normalization tends to significantly suffer from overfitting problems. We can see that the accuracy of differential private models is slightly degraded from the plain model as noise added to the parameter update affects the global accuracy of the model.

The attack success rate is more important as it shows how differential privacy defends the attack on FL. The attack success rate on the plain scheme is 98.74% while the attack on differential privacy models is around 74%. The attack success rate drops significantly. The differential privacy with the highest variance of $1e-1$ reduces the attack success rate significantly. However, the differential privacy model with the variance of $1e-1$ is not desired as classification accuracy is significantly low.

Table 7 shows the classification accuracy and the attack accuracy of the proposed scheme with various DNN on FedAvg protocol. We used the same DNNs defined in Table 1. Along with the previous simulations, we focused on type 1, 2, and 4 DNNs that have a particular behavior associated with their structure. With various α values, the proposed DNN converted the input mini-batch into the digested mini-batch. It can be seen that the DNN does not affect the classification accuracy significantly as DNN is also trained for reducing the classification error. Also, the collaborative neural network trains itself to increase classification accuracy. Similarly, with the application of FedSgd protocol, the proposed DNN converts the underlying features in original data into different features. The attack success rate substantiates the fact that the DNN completely converts the image feature. Furthermore, the adversary trains the attack model using half of the samples of the target device. However, most of the experiments had an attack success rate of less than 10%. This shows that in most cases, the attack model failed to identify the other half of the data

Table 5 – Accuracy and attack success rate comparisons over various α values on SVHN dataset.

Digestive type	α	L1		L2		sim	
		Acc	Att acc	Acc	Att acc	Acc	Att acc
Type 1	1e-1	92.307	15.0	92.338	14.0	92.411	9.0
Type 1	1e-2	92.273	13.0	92.745	11.0	92.384	16.0
Type 1	1e-3	92.331	17.0	92.189	8.0	92.086	17.0
Type 1	1e-4	92.382	6.0	92.974	20.0	92.441	7.0
Type 2	1e-1	88.692	18.0	88.843	7.0	89.368	14.0
Type 2	1e-2	88.989	9.0	88.949	17.0	88.408	15.0
Type 2	1e-3	87.735	11.0	88.916	16.0	90.209	10.0
Type 2	1e-4	89.593	14.0	88.794	10.0	78.172	16.0
Type 4	1e-1	92.205	14.0	91.927	19.0	92.437	16.0
Type 4	1e-2	92.492	16.0	92.059	15.0	92.430	12.0
Type 4	1e-3	92.402	17.0	92.740	9.0	92.494	14.0
Type 4	1e-4	92.957	7.0	92.501	9.0	91.901	12.0

Table 6 – Accuracy and attack success rate of passive local attacker on plain model and differentially private models on CIFAR-100 dataset.

Scheme	Accuracy	attack success rate
Plain model	26.80	98.74
$\delta^2 = 1e-4$	26.84	74.15
$\delta^2 = 1e-3$	22.46	72.27
$\delta^2 = 1e-2$	22.07	76.64
$\delta^2 = 1e-1$	10.01	49.37

that was transformed through the DNN. This shows that even though the attack algorithm extracts information from the target's collaborative neural network, the attack model failed to extract crucial information for identifying the digested data using the original private data.

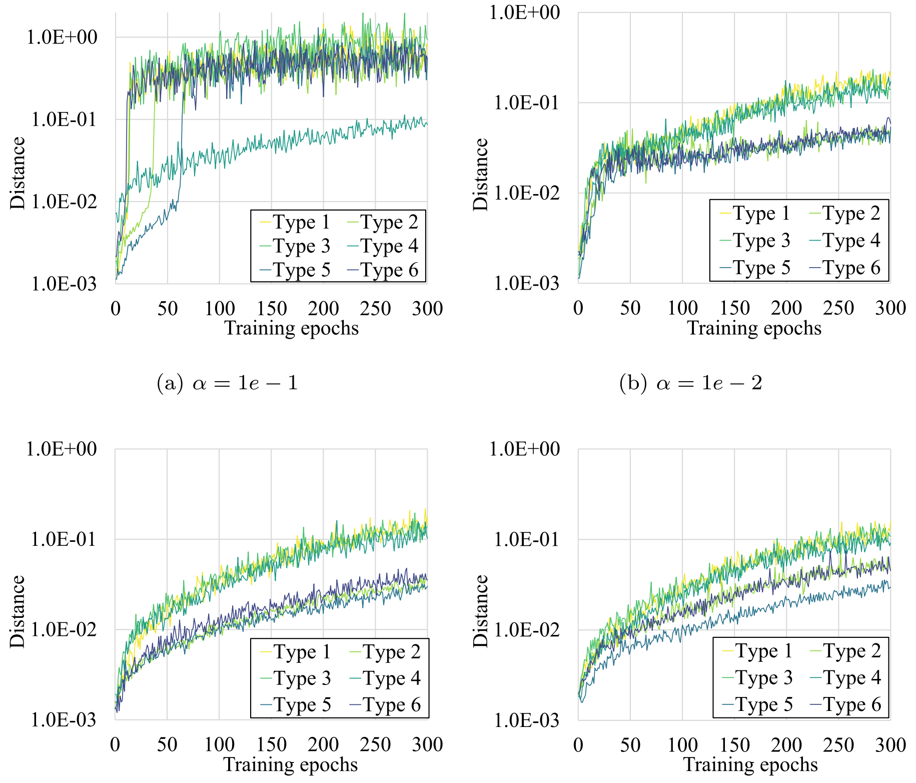
4.3. *Analysis on distance loss*

Distance loss is one of the most crucial components of our proposed scheme. Throughout the training, the goal was to maintain the distance loss higher than α. Fig. 6 shows the change in the average distance losses of five models throughout the training epochs. Architecture types and α values affect the formation of distance loss. In $\alpha = 1e-1$, the model updates the digestive loss rapidly up to $1e-1$. Exceeding that point, the influence of distance loss in digestive loss is neglected. We have anticipated that the distance loss will decrease after exceeding α, though it did not decrease after it exceeded its requirements. This is because of the sequential nature of the training protocol. DNNs and collaborative neural networks are trained sequentially. Thus, although the DNN rapidly increases distance loss, the collaborative network adapts itself to the DNN and vice versa. Thus after exceeding the α, distance loss never decreases as the collaborative network has already adapted itself. A similar rapid increment of distance loss is observed in simulations with $\alpha = 1e-2$. Type 4 showed a moderate increment in the distance loss. The slow increment may contribute to the highest accuracy as collaborative neural networks could adapt themselves to digested feature domain.

When α is too small, such as $1e-4$, a rapid surge in the distance loss is not observed. Fig. 6 (d) is an example where initial distance loss is already bigger than α. In such cases, the digestive network is only minimizing classification loss whereas the distance is automatically slightly increased as the training continues. This implies that increasing classification accuracy

Table 7 – Accuracy and attack success rate of passive local attacker on FedAvg with proposed DNN.

Digestive type	α	L1		L2		sim	
		Acc	Att acc	Acc	Att acc	Acc	Att acc
Type 1	1e-1	18.73	37.58	23.02	5.755	21.31	12.20
Type 1	1e-2	24.73	49.37	23.75	9.237	23.53	11.65
Type 1	1e-3	21.74	6.349	24.25	6.448	23.64	14.933
Type 1	1e-4	22.52	2.235	23.77	6.369	21.28	5.657
Type 2	1e-1	23.33	62.05	26.35	13.865	29.56	2.987
Type 2	1e-2	21.53	3.125	19.35	8.979	31.07	13.92
Type 2	1e-3	20.28	9.494	22.87	3.679	23.36	21.34
Type 2	1e-4	22.96	4.451	28.79	8.030	25.32	9.237
Type 4	1e-1	25.68	8.4454	25.90	5.894	22.56	2.690
Type 4	1e-2	25.25	14.750	24.62	12.50	22.90	7.183
Type 4	1e-3	27.44	8.069	23.75	9.138	25.02	8.930
Type 4	1e-4	21.98	26.167	25.08	5.835	23.42	12.26



(a) $\alpha = 1e - 1$ (b) $\alpha = 1e - 2$

(c) $\alpha = 1e - 3$ (d) $\alpha = 1e - 4$

Fig. 6 – Distance loss per epoch on proposed scheme with L2 distance metric.

and increasing the distance of gradients are not inconsistent. That is, minimizing the distance loss does not increase the classification loss. Moreover, training for increasing classification accuracy assists in an increase in distance loss.

From the same graph, different effects caused by various types of DNN are illustrated. The distance loss incremented by the collaborative neural network is significantly affected by the feature map sizes. Architecture types, including feature map reduction, not only suffered from classification accuracy but also showed a slow increment in the distance loss.

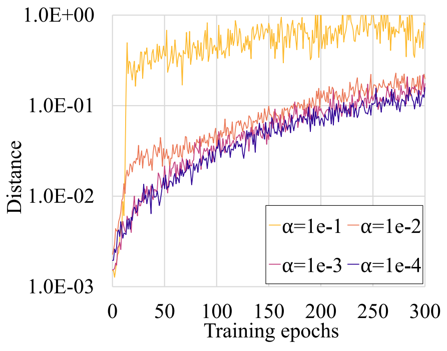
Fig. 7 shows how the distance loss is shaped with respect to different *alpha* values under different distance metrics. In every scheme, simulation with $alpha = 1e - 1$ showed a rapid increment of distance in the early training period. Since distance metrics are all different, $\alpha = 1e - 1$ does not have an identical impact on the model as well. The $\alpha = 1e - 1$ already is too small to be a threshold for the proposed schemes with cosine similarity. Thus distance losses for all experiments on the similarity are at a similar level as they do not have to update their DNN based on the distance loss.

The proposed DNN on FL scheme offers a better FL environment for an FL service provider. FL service provider who incorporates differential privacy within its FL scheme has to decide the adequate level of noise for securing the whole scheme. The accuracy degradation due to the presence of noise is inevitable for the FL service provider and every mobile device. Under the FL scheme with the proposed DNN, the FL service provider can select the level of privacy by selecting the proper α. The level of α has little effect on the classification accuracy. That is, the service provider can choose the desired α values with little concern for the performance of the FL scheme.

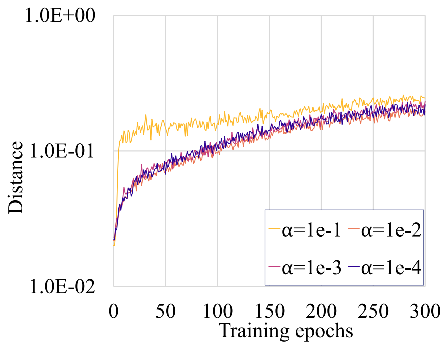
4.4. *Attack simulations on digestive neural networks*

The aforementioned experiments have substantiated that the proposed scheme can defend against the membership inference attack more effectively than the differential privacy scheme. Here we investigate in further detail, the details of the relationship between the attack algorithm and the proposed scheme. Table 8 shows the process of the attack initiated on the plain, differential privacy, and our proposed schemes with various types, threshold, and distance metrics. The attack algorithm generates an initial random image and repetitively updates the image so that the image can have more similar gradients with the target gradients.

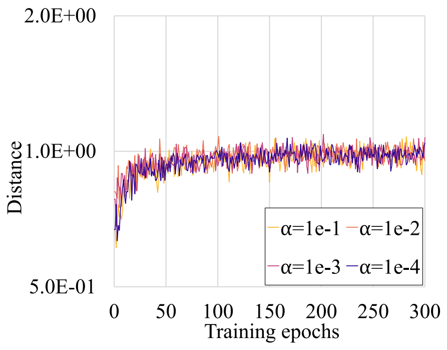
The attack on the plain scheme has successfully recovered the image. Moreover, the recovered image retains similar visual features with the ground truth. On the other hand, an attack on differential privacy with $\delta^2 = 1e - 4$ also showed that the attack algorithm has recovered the image that contains relatively similar features to the ground truth. While the attacks on the proposed scheme have reconstructed images that do not contain any representative features that are similar to the ground truth. This confirms that our proposed DNN is operating as we designed. The DNN pushes the input minibatch so that it can have a dissimilar gradient with the original



(a) Type 1 L2



(b) Type 1 L1



(c) Type 1 sim

Fig. 7 – Distance loss with respect to various α and training epochs in different distance metrics.

gradients. Pushing the gradients also alters the visual features of the original image into totally disparate features. This is why our proposed scheme is more secure than the differential privacy scheme. Even the attack algorithm successfully recovers the image using gradients, the resulting image already has totally different features from the original data. The attacker may successfully obtain data; however, will not be able to extract any private information based on the analysis of totally different features than what the attacker is looking for.

Table 9 shows visualization of the digested data from the original data. The digested data is the output of the DNN. The collaborative neural networks receive this mini-batch from the DNN and send gradients to the server based on the mini-batch data. Thus, a successful membership inference attack on the gradient will reconstruct these data. Thus the attacker hardly will gain any meaningful information from the mini-batch data. More importantly, the digestion of data is conducted locally. Thus unless the attacker can sneak into the system of a participating mobile device, recovering ground truth from digested data would not be possible.

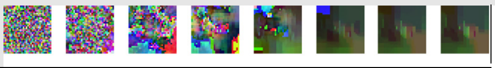
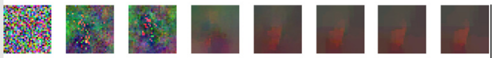
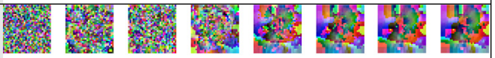
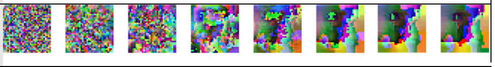
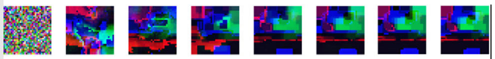
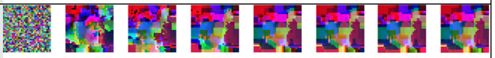
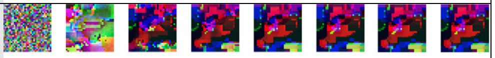
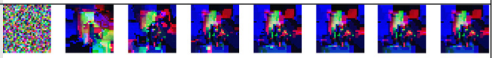
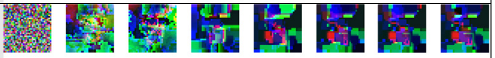
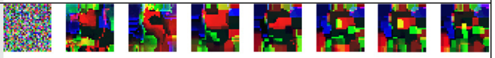
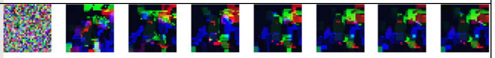
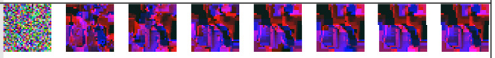
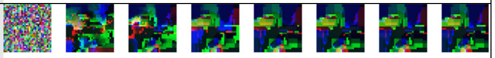
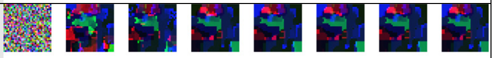
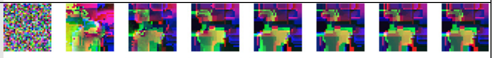
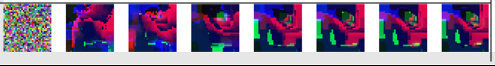
4.5. *Train-ability of the digested data*

We had a curiosity about how our proposed scheme could achieve higher accuracy on the CIFAR-10 and SVHN datasets with the DNN. To find what caused this, we incorporated t-distributed Stochastic Neighbor Embedding (t-SNE) to see how the model is perceiving its data. The t-SNE is a dimensionality reduction algorithm that converts multi-dimensional data into two-dimensional data without losing spatial relationships within high-dimensional data. The algorithm often forms clusters by placing data points with high similarity together and placing data points of low similarity far apart.

We analyzed each model by applying t-SNE to the output of the last convolutional layer of each model. This illustrates the visualization of the learned representation of input data. Since all models have identical architecture and hyper-parameters with identical epochs of training, the learned representation can reveal the learnability of a given dataset. To quantify the learned representation without any bias, we conducted experiments only on the test data of the CIFAR-10 dataset.

Fig. 8 illustrates a visual depiction of the reduced dimension by t-SNE. We analyzed plain, differential private schemes with two proposed schemes with the highest accuracy. A dot in the figure represents a data point in a dataset, and the color denotes its class. In the figure, plain and differential privacy with $\delta^2 = 1e-2$, $\delta^2 = 1e-3$ do not show clear clusters. We can consider noise added to the update in the differential privacy scheme as noise added to the data. This shows that data is too sophisticated to differentiate using convolutional layers' learned representations. On the other hand, clusters are vividly visible on t-SNE analysis in the proposed schemes. This indicates that feature extraction of the digested data is easier and more easily separable than the differential privacy schemes. The DNN actually converts the CIFAR-10 dataset, so that the convolutional layers in collaborative neural networks

Table 8 – Progress of attack simulation.

Scheme	Attack iteration	Ground Truth
Plain scheme		
DP $\delta^2 = 10^{-4}$		
DP $\delta^2 = 10^{-3}$		
DP $\delta^2 = 10^{-2}$		
Type 1 L2 α=1e-1		
Type 1 L1 α=1e-1		
Type 1 sim α=1e-1		
Type 1 L2 α=1e-2		
Type 1 L1 α=1e-2		
Type 1 sim α=1e-2		
Type 1 L2 α=1e-3		
Type 1 L1 α=1e-3		
Type 1 sim α=1e-3		
Type 1 L2 α=1e-4		
Type 1 L1 α=1e-4		
Type 1 sim α=1e-4		

can extract features effectively. The digested data may seem like random noise, however, it actually converted the dataset better so that the collaborative network can take advantage of it.

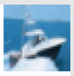
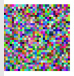
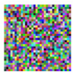
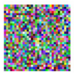
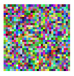
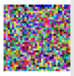
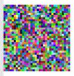
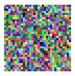
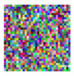
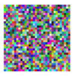
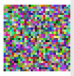
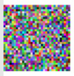
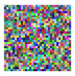
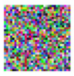
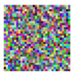
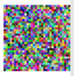
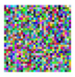
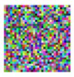
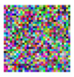
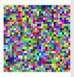
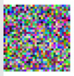
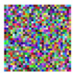
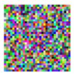
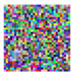
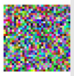
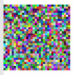
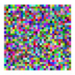
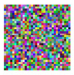
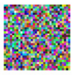
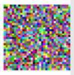
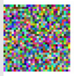
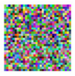
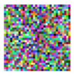
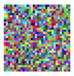
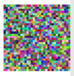
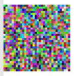
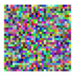
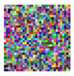
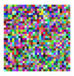
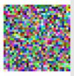
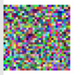
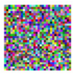
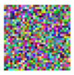
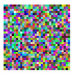
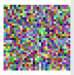
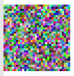
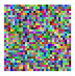
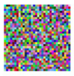
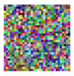
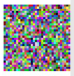
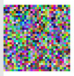
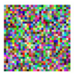
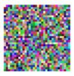
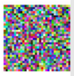
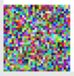
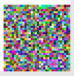
5. Related works

In this section, we outline the existing work pertaining to inference attacks on ML and FL, and privacy preservation techniques in FL.

5.1. *Membership inference attack on machine learning models*

Membership inference attack, or inference attack, can target various ML algorithms over various environments. Cai *et* al. proposed an inference attack on basic classifiers trained on social network graph data (Cai et al., 2018; Li et al., 2021). For an inference attack on deep learning models, Shokri *et* al. proposed a scheme to identify training data by inspecting a model (Shokri et al., 2017). The authors leveraged a typical behavior of an overfitted model. When a sample from training data is given, an overfitted model shows higher confidence than a generalized model. By assessing this different behavior, the attack model determines if a given datum is included in the training data of a victim model. To train the attack model, multiple shadow models were incorporated to simulate the behavior of the target model. Unlike the initial inference attacks that required multiple shadow models, the data transferring attack by Salem *et* al. found out that the inference attack can be accomplished by only one shadow model (Salem et al., 2019). The authors proposed three different approaches with an attack model that does not require any training for inference. They reported that statistical analysis on the posteriors is sufficient to explain if a sample is included in the training data. Another similar work (Yeom et al., 2018) showed that the overfitting of the target model is a sufficient condition, not a necessary condition for inference attack. The authors simulated the attack on a real-world dataset. De-

Table 9 – Digested Image of $\alpha = 10$.

β	Acc	Atk acc					
type1 L2 1e-1	82.448	10.0					
type1 L1 1e-1	83.158	16.0					
type1 sim 1e-1	81.532	8.0					
type1 L2 1e-2	81.476	8.0					
type1 L1 1e-2	81.064	13.0					
type1 sim 1e-2	81.672	9.0					
type1 L2 1e-3	83.018	14.0					
type1 L1 1e-3	81.002	13.0					
type1 sim 1e-3	81.716	13.0					
type1 L2 1e-4	81.960	12.0					
type1 L1 1e-4	81.060	9.0					
type1 sim 1e-4	82.912	15.0					

viating from using the shadow model, Lenio *et* al. proposed an attack based on white-box assumptions (Leino and Fredrikson, 2020). They identified that an overfitted model only uses certain features to make an inference.

5.2. *Membership inference attack on FL*

FL is more vulnerable to inference attacks than other ML algorithms. In non-collaborative learning, training is processed securely. However, in FL, the training process is exposed through a communication channel, allowing inference attacks to exploit the training process. Nasr *et* al. defined possible adversaries against parameter sharing FL scheme (Nasr et al., 2019). The authors defined an active and a passive attacker who exploits the communication protocol of the victim FL. The passive attacker only observes the communication channel and snatch parameter update to launch an inference attack against a participant of the FL. The active attacker actively participates in the FL protocol to induce the victim to reveal more information. However, the attack model for the passive attacker needs a substantial amount of data and extensive training. Melis *et* al. proposed an alternative approach of defining privacy leakage in the FedAvg scheme (Melis et al., 2019). Feature analysis through deep neural network examine features that are insignificant for classification objective. The attack successfully showed the extraction of private information. The proposed attack demonstrated its effectiveness on an ethnicity classifier based on the LFW dataset. The model successfully identified the presence of glasses or sunglasses in the training data.

Generative Adversarial Networks(GAN) are also leveraged to extract private information from the model (Hitaj et al., 2017; Zhang et al., 2020b). Hitaj *et* al. demonstrated the feasibility of constructing a GAN-based attack model. The attack model maliciously participates in the training protocol to train the generative model effectively. However, there is no guaran-

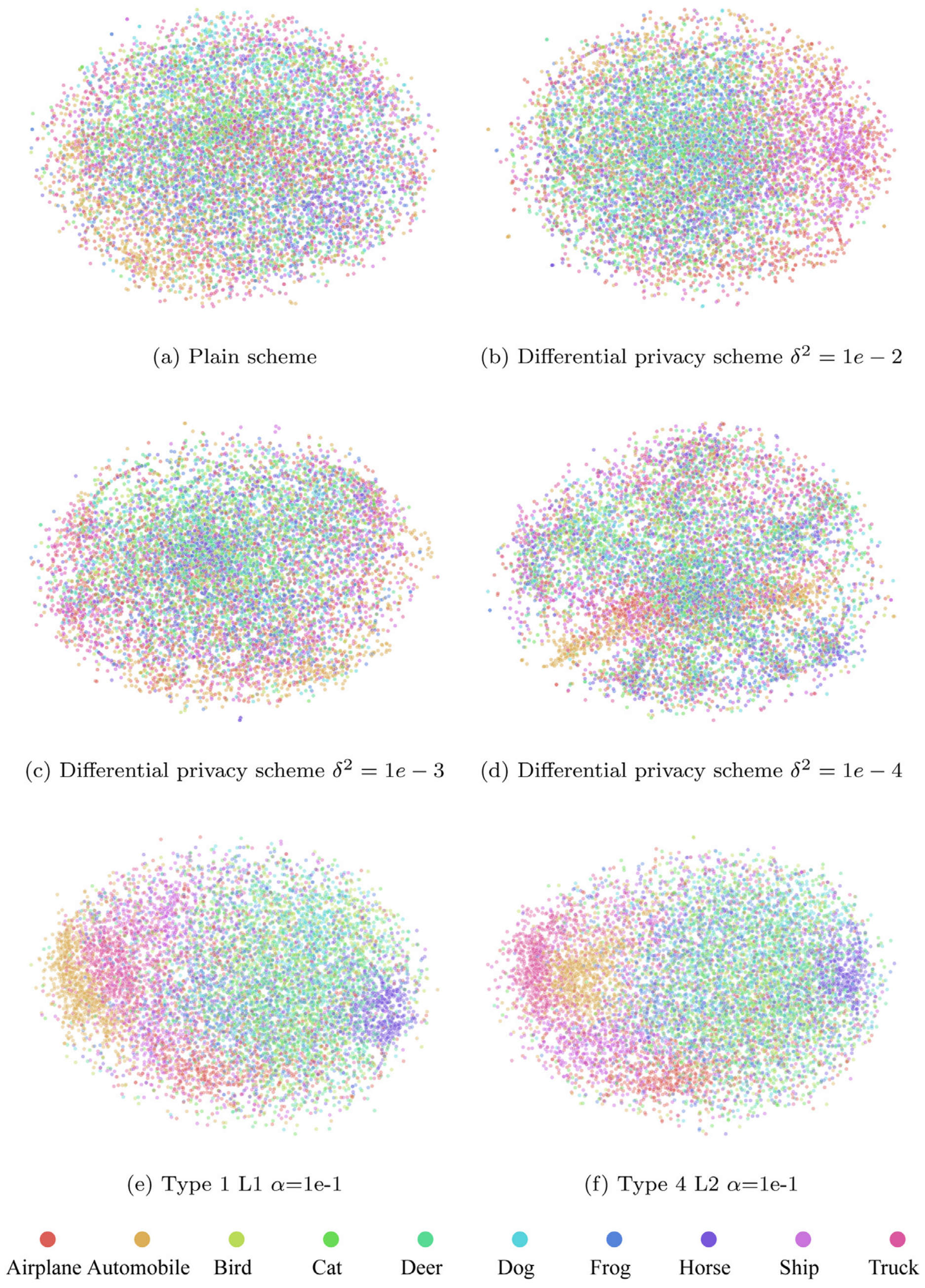


Fig. 8 – t-SNE analysis.

tee that the generated samples are identical to the original samples in the training data. Similarly, Wang *et* al. installed an additional validation step to quantify how similar the synthetic data are compared to the original data in the training set.

In FedSgd, a different approach is proposed where deep leakage from gradients exploits gradient updates from the devices (Zhu et al., 2019). In FedSgd, each gradient update from devices includes a direction to enhance the current parameter with respect to its mini-batch data. The attack model generates a random data sample and calculates the gradient of the sample with respect to the newest model distributed by the central server. The attack model continuously updates the sample so that the distance of the victim's gradient and the gradient from the sample can decrease. This induces the sample to be identical to the original sample in the victim's training dataset. Furthermore, some works focused on increasing the stability of the attack (Geiping et al., 2020; Zhao et al., 2020). Instead of Euclidean distance by deep leakage from gradients, inverting gradients used cosine similarity to match directions.

5.3. *Privacy protection mechanisms on FL*

Differential privacy has been widely used to dissuade privacy breaches in FL. Differential privacy is implemented on FL as an added noise to its updates (Agarwal et al., 2018; Geyer et al., 2017; Jayaraman et al., 2018; Wei et al., 2020). In (Wei et al., 2020), the authors showed that FL models could converge in the presence of differential privacy. However, most of the schemes suffer from performance degradation.

Alternatively, cryptographic approaches are utilized to secure FL. To this end, homomorphic encryption has been applied to FL to enhance privacy (Dong et al., 2020; Fang et al., 2021; Liu et al., 2019; Phong et al., 2017; Zhang et al., 2020a; Zhang et al., 2020). The advantage of homomorphic encryption is that arithmetic calculation is possible on encrypted data. However, it requires significant computational resources. A number of studies tried to reduce the computation cost of homomorphic encryption. For instance, in (Zhang et al., 2020a), the authors utilized the quantization of parameters to reduce overhead. Similarly, ElGamel homomorphic encryption is usually used for lightweight computation on limited network bandwidth in the Internet of Things (IoT) environment (Fang et al., 2021), and the Chinese remainder theorem is also used for reducing the size of gradient update (Zhang et al., 2020).

As aggregation is one of the most crucial components in FL, it is essentially important to secure aggregation. To date, different works have focused on the aggregation security of FL (Fang et al., 2020; Li et al., 2020; Tran et al., 2021). Fang *et* al. incorporated multi-party computation to enable aggregation of average encrypted weight updates in the FedAvg algorithm (Fang et al., 2020). Similarly, Chain-PPFL proposed a token-based system with chaining participants to counter the honest-but-curious server (Li et al., 2020).

6. Conclusion

In this paper, We proposed DNN and its training protocol in a collaborative setting for secure and effective FL training. The proposed scheme testified its performance by showing higher classification accuracy while a lower attack success rate than differential privacy models. The paper has substantiated the scalability of the proposed scheme as it shows high performance on both FedAvg and FedSgd protocols. Also, the paper demonstrated the experimental analysis on the successful performance of the proposed scheme on FL. In the future, we plan to identify the relationship between distance loss and classification loss in order to clarify the relationship between classification accuracy and attack success rate.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Hongkyu Lee: Writing - original draft, Software, Validation, Investigation. **Jeehyeong Kim:** Methodology, Visualization. **Seyoung Ahn:** Software, Data curation. **Rasheed Hussain:** Writing - review & editing. **Sunghyun Cho:** Resources, Funding acquisition. **Junggab Son:** Supervision, Project administration, Conceptualization, Methodology, Investigation, Writing - review & editing.

Acknowledgment

This work was supported by the MSIT (Ministry of Science, ICT), Korea, under the High-Potential Individuals Global Training Program (2019-0-01601) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.cose.2021.102378

REFERENCES

Agarwal N, Suresh AT, Yu FX, Kumar S, McMahan B. cpSGD: Communication-efficient and differentially-private distributed SGD, 31; 2018. p. 7575–86. doi:a

Cai Z, He Z, Guan X, Li Y. Collective data-sanitization for preventing sensitive information inference attacks in social networks. IEEE Trans. Dependable Secur. Comput. 2018;15(4):577–90. doi:10.1109/TDSC.2016.2613521.

Dong Y, Chen X, Shen L, Wang D. Eastfly: efficient and secure ternary federated learning. Comput. Secur. 2020;94:101824. doi:10.1016/j.cose.2020.101824.

Dwork C. Differential privacy. Int. Colloq. Autom. Lang. Program. 2006;4052:1–12. doi:10.1007/11787006_1.

Fang C, Guo Y, Hu Y, Ma B, Feng L, Yin A. Privacy-preserving and communication-efficient federated learning in internet of things. Comput. Secur. 2021;103:102199. doi:10.1016/j.cose.2021.102199.

Fang C, Guo Y, Wang N, Ju A. Highly efficient federated learning with strong privacy preservation in cloud computing. Comput. Secur. 2020;96:101889. doi:10.1016/j.cose.2020.101889.

Friedrich R. Untersuchungen über systeme integrierbarer unktionen. Mathematische Annalen 1910;69(4):449–97. doi:10.1007/bf01457637.

Geiping J, Bauermeister H, Dröge H, Moeller M. Inverting gradients - How easy is it to break privacy in federated learning?, 33; 2020. p. 16937–47.

Geyer, R. C., Klein, T., Nabi, M., 2017. Differentially private federated learning: a client level perspective. arXiv preprint arXiv:1712.07557.

He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 770–8.

Hisamoto S, Post M, Duh K. Membership inference attacks on sequence-to-sequence models: is my data in your machine translation system? Trans. Assoc. Comput. Linguist. 2020;8:49–63. doi:10.1162/tacl_a_00299.

Hitaj B, Ateniese G, Perez-Cruz F. Deep models under GAN: Information leakage from collaborative deep learning. In: Proceedings of ACM SIGSAC Conference on Computer and Communications; 2017. p. 603–18. doi:10.1145/3133956.3134012.

Jayaraman B, Wang L, Evans D, Gu Q. Distributed learning without distress: Privacy-preserving empirical risk minimization. Proceedings of Advances in Neural Information Processing Systems, 2018.

Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks, 25; 2012.

Leino K, Fredrikson M. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In: Proceedings of USENIX Security Symposium (USENIX Security 20); 2020. p. 1605–22.

Li K, Luo G, Ye Y, Li W, Ji S, Cai Z. Adversarial privacy preserving graph embedding against inference attack. IEEE Internet Things J. 2020;Early Access:1–12. doi:10.1109/JIOT.2020.3036583.

Li K, Luo G, Ye Y, Li W, Ji S, Cai Z. Adversarial privacy-preserving graph embedding against inference attack. IEEE Internet Things J. 2021;8(8):6904–15. doi:10.1109/JIOT.2020.3036583.

Li Y, Zhou Y, Jolfaei A, Yu D, Xu G, Zheng X. Privacy-preserving federated learning framework based on chained secure multi-party computing. IEEE Internet Things J. 2020. doi:10.1109/JIOT.2020.3022911. 1–1

Liang Y, Cai Z, Yu J, Han Q, Li Y. Deep learning based inference of private information using embedded sensors in smart devices. IEEE Netw. 2018;32(4):8–14. doi:10.1109/MNET.2018.1700349.

Lim WYB, Luong NC, Hoang DT, Jiao Y, Liang Y-C, Yang Q, Niyato D, Miao C. Federated learning in mobile edge networks: acomprehensive survey. IEEE Commun. Surv. Tutor. 2020;22(3):2031–63. doi:10.1109/COMST.2020.2986024.

Liu C, Chakraborty S, Verma D. Secure model fusion for distributed learning using partial homomorphic encryption. Policy-Based Auton. Data Gov. 2019:154–79. doi:10.1007/978-3-030-17277-0_9.

Maaten LVD, Hinton G. Visualizing data using t-SNE. J. Mach. Learn. Res. 2008;9(86):2579–605.

McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data, PMLR 54; 2017. p. 1273–82.

Melis L, Song C, Cristofaro ED, Shmatikov V. Exploiting unintended feature leakage in collaborative learning. In: Proceedings of IEEE Symposium on Security and Privacy (SP); 2019. p. 691–706. doi:10.1109/SP.2019.00029.

Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: Proceedings of IEEE Symposium on Security and Privacy (SP); 2019. p. 739–53. doi:10.1109/SP.2019.00065.

Phong LT, Aono Y, Hayashi T, Wang L, Moriai S. Privacy-preserving deep learning via additively homomorphic encryption, 13; 2017. p. 1333–45. doi:10.1109/TIFS.2017.2787987.

Salem A, Zhang Y, Humbert M, Berrang P, Fritz M, Backes M. ML-leaks: Model and data independent membership inference at-tacks and defenses on machine learning models. Proceedings of the Network and Distributed System Security Symposium (NDSS), 2019.

Shokri R, Shmatikov V. Privacy-preserving deep learning. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), 2015.

Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. Proceedings of the IEEE Symposium on Security and Privacy, 2017.

Tran A-T, Luong T-D, Karnjana J, Huynh V-N. An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. Neurocomputing 2021;422:245–62. doi:10.1016/j.neucom.2020.10.014.

Wei K, Li J, Ding M, Ma C, Yang HH, Farokhi F, Jin S, Quek TQS, Poor HV. Federated learning with differential privacy: Algorithms and performance analysis, 15; 2020. p. 3454–69. doi:10.1109/TIFS.2020.2988575.

Yeom S, Giacomelli I, Fredrikson M, Jha S. Privacy risk in machine learning:analyzing the connection to overfitting. In: Proceedings of IEEE Computer Security Foundations Symposium (CSF); 2018. p. 268–82. doi:10.1109/CSF.2018.00027.

Zhang C, Li S, Xia J, Wang W, Yan F, Liu Y. BatchCrypt: efficient homomorphic encryption for cross-silo federated learning. In: Proceedings of USENIX Annual Technical Conference (USENIX ATC 20); 2020. p. 493–506.

Zhang J, Zhang J, Chen J, Yu S. GAN enhanced membership inference: A passive local attack in federated learning. In: Proceedings of IEEE International Conference on Communications (ICC); 2020. p. 1–6. doi:10.1109/ICC40277.2020.9148790.

Zhang X, Fu A, Wang H, Zhou C, Chen Z. A privacy-preserving and verifiable federated learning scheme. In: ICC 2020 - 2020 IEEE International Conference on Communications (ICC); 2020. p. 1–6. doi:10.1109/ICC40277.2020.9148628.

Zhao, B., Mopuri, K. R., Bilen, H., 2020. iDLG: improved deep leakage from gradients. arXiv:2001.02610, pp. 1-5.

Zhu L, Liu Z, Han S. Deep leakage from gradients. Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2019.

Hongkyu Lee received his BSE degree in Electrical Engineering from the Hanyang University, Ansan, South Korea (2019). Since 2019, he is a master's student with the Information and Intelligent Security (IIS) Lab. at Kennesaw State University. His research interest includes privacy-preserving deep learning and trustworthy machine learning that ensures robustness and accuracy of machine learning models.

Jeehyeong Kim received the BSE and Ph.D. degrees in Computer Science and Engineering from Hanyang University, South Korea, in 2015 and 2020, respectively. He was a Post-doctoral researcher with the Information and Intelligent Security (IIS) Lab. at Kennesaw State University from August 2020 to January 2021. Currently, he is a senior researcher of Autonomous IoT Platform Research Center at Korea Electronics Technology Institute, Republic of Korea. His research interests include machine learning in cyber security and digital twin.

Seyoung Ahn received the B.E. degree in Computer Science and Engineering from Hanyang University, South Korea, in 2018. He is currently pursuing the M.S.-leading-to-Ph.D. degree in Computer Science and Engineering, Hanyang University, South Korea. Since 2018, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include federated learning and network automation with machine learning and deep learning techniques.

Rasheed Hussain is currently working as an Associate Professor and Head of the MS program in Security and Network Engineering (SNE) at Innopolis University, Russia. He is also the Head of the Networks and Blockchain Lab at Innopolis University. He received his B.S. Engineering degree in Computer Software Engineering from University of Engineering and Technology, Peshawar, Pakistan in 2007, MS and PhD degrees in Computer Science and Engineering from Hanyang University, South Korea in 2010 and 2015, respectively. He worked as a Postdoctoral Fellow at Hanyang University, South Korea from March 2015 to August 2015 and as a guest researcher and consultant at University of Amsterdam (UvA) from September 2015 till May 2016. He also worked as Assistant Professor at Innopolis University, Innopolis, Russia from June 2016 till December 2018. He serves as editorial board member for various journals including IEEE Access, IEEE Internet Initiative, Internet Technology Letters, Wiley, and serves as reviewer for most of the IEEE transactions, Springer and Elsevier Journals. He was the symposium chair for IEEE ICC CISS 2021. He also serves as technical program committee member of various conferences such as

IEEE VTC, IEEE VNC, IEEE Globecom, IEEE ICCVE, and so on. He is a certified trainer for Instructional Skills Workshop (ISW). Furthermore, he is also ACM Distinguished Speaker and Senior Member of IEEE. His research interests include Information Security and Privacy and particularly security and privacy issues in Vehicular Ad Hoc NETworks (VANETs), vehicular clouds, and vehicular social networking, applied cryptography, Internet of Things, Content-Centric Networking (CCN), Digital Twins (DT), Artificial Intelligence in cybersecurity, eXplainable AI (XAI), and blockchain.

Sunghyun Cho received his B.S., M.S., and Ph.D. in Computer Science and Engineering from Hanyang University, Korea, in 1995, 1997, and 2001, respectively. From 2001 to 2006, he was with Samsung Advanced Institute of Technology, and with Telecommunication R&D Center of Samsung Electronics, where he has been engaged in the design and standardization of MAC and network layers of WiBro/WiMAX and 4G-LTE systems. From 2006 to 2008, he was a Postdoctoral Visiting Scholar in the Department of Electrical Engineering, Stanford University. He is currently a Professor in the dept. of Computer Science and Engineering, Hanyang University.

Junggab Son received the BSE degree in computer science and engineering from Hanyang University, Ansan, South Korea (2009), and the Ph.D. degree in computer science and engineering from Hanyang University, Seoul, South Korea (2014). From 2014 to 2016, he was a Post-doctoral Research Associate with the Department of Math and Physics, North Carolina Central University. From 2016 to 2018, he was a Research Fellow and a Limited-term Assistant Professor at Kennesaw State University. Since 2018, he has been an Assistant Professor of Computer Science and a Director of Information and Intelligent Security (IIS) Lab. at Kennesaw State University.