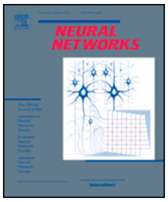
Contents lists available at ScienceDirect

Neural Networks

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



Full Length Article

MemberShield: A framework for federated learning with membership privacy

Faisal Ahmed [a,b,*], David Sánchez [b], Zouhair Haddi [a], Josep Domingo-Ferrer [b]

[a] *NVISION Systems and Technologies SL, Gran Via Carles III, 124, ent. 1a, 08034, Barcelona, Catalonia, Spain*

[b] *Universitat Rovira i Virgili, Dept. of Computer Engineering and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Països Catalans 26, 43007 Tarragona, Catalonia, Spain*

ARTICLE INFO

Dataset link: https://github.com/faisalahm3d/MemberShield

Keywords:
Federated learning
Privacy
Membership inference attack
Deep learning models
Data protection

ABSTRACT

Federated Learning (FL) allows multiple data owners to build high-quality deep learning models collaboratively, by sharing only model updates and keeping data on their premises. Even though FL offers privacy-by-design, it is vulnerable to membership inference attacks (MIA), where an adversary tries to determine whether a sample was included in the training data. Existing defenses against MIA cannot offer meaningful privacy protection without significantly hampering the model's utility and causing a non-negligible training overhead. In this paper we analyze the underlying causes of the differences in the model behavior for member and non-member samples, which arise from model overfitting and facilitate MIAs. Accordingly, we propose MemberShield, a generalization-based defense method for MIAs that consists of: (i) one-time preprocessing of each client's training data labels that transforms one-hot encoded labels to soft labels and eventually exploits them in local training, and (ii) early stopping the training when the local model's validation accuracy does not improve on that of the global model for a number of epochs. Extensive empirical evaluations on three widely used datasets and four model architectures demonstrate that MemberShield outperforms state-of-the-art defense methods by delivering substantially better practical privacy protection against all forms of MIAs, while better preserving the target model utility. On top of that, our proposal significantly reduces training time and is straightforward to implement, by just tuning a single hyperparameter.

1. Introduction

Federated learning (FL) is a distributed machine-learning paradigm where multiple parties can jointly train deep-learning models without outsourcing their (private) data (McMahan, Moore, Ramage, Hampson, & y Arcas, 2017). FL involves a server coordinating the learning process and a set of clients representing the data owners. At each learning round, the server sends a global model to each client, who trains it on their private data (McMahan et al., 2017) and returns the updated model (rather than the actual data) to the server. The server then aggregates model updates from the different clients to generate an updated global model that is redistributed. The process continues until model convergence. FL has been successfully adopted in various applications, including self-driving cars (Nguyen et al., 2022), voice recognition (Guliani, Beaufays, & Motta, 2021; Zhang, Bosch, & Olsson, 2021), word and emoji prediction (Hard et al., 2018; Ramaswamy, Mathews, Rao, & Beaufays, 2019), human activity recognition (Bettini, Civitarese, & Presotto, 2021; Ouyang, Xie, Zhou, Huang, & Xing, 2021; Sozinov, Vlassov, & Girdzijauskas, 2018; Xiao et al., 2021), and financial fraud detection (Kanamori et al., 2022; Myalil, Rajan, Apte, & Lodha, 2021; Suzumura et al., 2019; Yang, Zhang, Ye, Li, & Xu, 2019).

Although FL is supposed to provide privacy-by-design, recent work has shown that it is still vulnerable to privacy attacks, because deep neural networks are prone to memorizing (sensitive) information from training data (Carlini, Liu, Erlingsson, Kos, & Song, 2019; Fredrikson, Jha, & Ristenpart, 2015; Hitaj, Ateniese, & Perez-Cruz, 2017). In particular, *dishonest* or *honest-but-curious* server or clients can orchestrate privacy attacks to infer sensitive information from the updates sent by clients during the federated training rounds. The membership inference attack (MIA) is such a privacy attack, where an adversary with white-box or black-box access to the model tries to identify whether a data sample is part of the training dataset.

Even though MIAs affect machine learning models in general, FL is more vulnerable due to two specific reasons. First, the adversary can be the server or other clients, which have white-box access to the

[*] Corresponding author at: Universitat Rovira i Virgili, Dept. of Computer Engineering and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Països Catalans 26, 43007 Tarragona, Catalonia, Spain.

E-mail addresses: faisal.ahmed@nvision.es (F. Ahmed), david.sanchez@urv.cat (D. Sánchez), zouhair.haddi@nvision.es (Z. Haddi), josep.domingo@urv.cat (J. Domingo-Ferrer).

https://doi.org/10.1016/j.neunet.2024.106768
Received 18 March 2024; Received in revised form 28 July 2024; Accepted 27 September 2024
Available online 1 October 2024

target model parameters, hyperparameters, and learning algorithms. Also, attacks can be performed during each training round. Second, the target model continuously updates its parameters by using the same underlying training set across multiple communication rounds, thereby allowing adversaries to observe several representations of the training data (Shejwalkar & Houmansadr, 2021).

Defenses have been proposed to prevent MIAs based on various principles, including *regularization*, *adversarial training*, *knowledge distillation*, and *differential privacy*, which we survey in Section 2. However, existing defenses have one or more of the following limitations: (i) they are explicitly designed for centralized training and fail to maintain their performance in FL settings, (ii) they require additional public datasets that are not always available in sensitive domains like healthcare, (iii) they offer privacy protection against a specific category of MIAs, and/or (iv) they require training multiple models on non-overlapping subsets of training data, which is not applicable in the FL setting due to the (usually) small size of client data. On top of that, most defenses offer privacy at the expense of utility, and they add significant training overhead. To the best of our knowledge, no defense can offer meaningful privacy protection against all forms of MIAs in FL without compromising model utility and incurring additional training or inference overheads. In fact, achieving membership privacy while maintaining the model accuracy and without imposing training overhead remains an open challenge for FL due to contradicting requirements (Jebreel, Domingo-Ferrer, Blanco-Justicia, & Sánchez, 2022).

To tackle this challenge, in this paper we propose *MemberShield*, an FL framework that offers membership privacy by generalizing the local model on the client end. Specifically, our contributions are:

- We analyze the target model behavior on member and non-member samples to identify the most distinctive features an adversary can exploit to perform MIAs. We also demonstrate that model overfitting is the underlying cause of the difference in features for member and non-member samples, which stems from training a model with one-hot encoded labels and unintended sample memorization due to excessive training.
- We propose a defense mechanism to prevent MIAs by generalizing the local model in two steps: (1) regulating prediction confidence by training the model with soft-encoded ground truth labels, and (2) mitigating training sample memorization by implementing early stopping, considering both global and local model perspectives.
- We report extensive empirical analyses on different tasks comprising diverse data modalities and deep learning architectures to evaluate the privacy and accuracy that our defense offers. In addition, we compare our defense with several state-of-the-art defenses to show its superiority at delivering both practical membership privacy and high model utility with reduced training time.

The rest of the paper is organized as follows: Section 2 gives background on MIAs and discusses related defenses for MIAs. Section 3 describes the threat model we tackle. Section 4 reports an analysis of the target model's behaviors on member and non-member samples. Section 5 presents *MemberShield*. Sections 6 and 7 report empirical results on the effectiveness of *MemberShield* in terms of model accuracy, robustness against attacks, and runtime. Additional results on the robustness analysis are reported in Appendix. Section 8 gathers the conclusions and depicts several lines of future research.

2. Related works

In a membership inference attack (MIA), the adversary tries to determine whether a sample is part of a target model's training data. For this, the adversary first learns the behavior of the target model by *shadow training* (Shokri, Stronati, Song, & Shmatikov, 2017), where shadow models are trained as proxies for the target model on a dataset that has the same distribution as the target model's training set. The adversary then extracts the shadow models' output properties, such as gradient norm, confidence, entropy, and loss, for both the training dataset and a disjoint test set of the same size. The training samples' output properties and corresponding ground truth labels are labeled as *member*, while those of the test samples are labeled as *non-member*. These labeled data constitute the *attack dataset* on which the adversary constructs an *attack model*, which is basically a binary classifier able to distinguish *member* from *non-member*. The attack model can be a *neural network (NN)* or a simple *threshold function*. For the NN, the adversary trains it on the attack dataset in a supervised manner. For the threshold function, the adversary fine-tunes class-wise threshold values of different properties to make membership decisions. Finally, the adversary extracts the target model's output properties for that sample and inputs these into the *attack model* for membership prediction.

In the FL setting, the adversary can be either the server (*i.e.*, *global adversary*) or the clients (*i.e.*, *local adversaries*). The *global adversary* performs MIAs on each client's local model when the client returns the updated model after local training. To attack a specific client's local model, the *global adversary* constructs the shadow model with the same architecture as the target client's local model and trains on a dataset that mimics the corresponding client's private data distribution. In contrast, the *local adversaries* attack the global model that the server generates by aggregating all the local models. These adversaries aim to determine whether any participating client included the target sample in their private training data. Local adversaries construct the shadow model by replicating the global model's architecture, and subsequently train that shadow model on a dataset that has a cumulative distribution of all clients' private data.

The global adversary attacking local models is more powerful than local adversaries attacking the global model. The former can infer the target sample's membership information and identify the corresponding client, whereas the latter can only determine the membership information. Also, the success rate of local adversaries' attacks is lower than that of attacks by the global adversary, since the data of multiple clients are blended into the global model.

Existing defenses against MIAs are designed for centralized learning and founded on one of the following principles: *regularization, adversarial training, knowledge distillation*, and *differential privacy*.

2.1. Regularization

Regularization prevents MIAs by alleviating model overfitting, which is the primary source of membership risk in deep learning models (Jia, Salem, Backes, Zhang, & Gong, 2019; Salem et al., 2018; Shokri et al., 2017).

Vanilla regularization methods designed to mitigate overfitting (but not specifically MIAs), such as *dropout* (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), *weight-decay (L2)* (Geman, Bienenstock, & Doursat, 1992), *confidence penalty* (Pereyra, Tucker, Chorowski, Kaiser, & Hinton, 2017) and *label smoothing* (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016), can be leveraged as defenses. However, as reported in Kaya and Dumitras (2021) and Kaya, Hong, and Dumitras (2020), these techniques offer limited performance against MIAs, despite marginally minimizing the generalization gap. Our experimental results reported in Section 7 are in line with the results of Kaya and Dumitras (2021) and Kaya et al. (2020). This is because dealing with overfitting only involves minimizing the generalization gap, which is the difference between the performance (typically loss or accuracy) on the training data and the validation data. The goal is to achieve consistent performance for training and unseen (test) data by learning only the underlying patterns from the training data, therefore ignoring any noises and outliers. However, when addressing MIA, the focus must extend beyond minimizing the generalization gap. A comprehensive approach is required that simultaneously achieves good generalization and ensures that the statistical properties of the model are similar for both training and unseen samples. At best, marginal privacy protection can be achieved when two or more regularization techniques are applied together at the expense of utility loss. In this case, finding the right combination is another big challenge. For example, if we consider enforcing dropout and L2 regularization in a single deep learning model having many layers (RestNet, VGG, DenseNet), selecting the layers where this regularization is enforced and to what extent (*i.e.*, percentage of dropout) becomes a combinatorial optimization problem.

In contrast, RelaxLoss (Chen, Yu, & Fritz, 2022) is a regularization-based technique specifically designed for mitigating MIAs. It is designed based on the strong assumption that the optimal MIA depends only on the sample loss undermining entropy and prediction confidence, which are more powerful metrics for MIAs than loss, as demonstrated in Song and Mittal (2021). Moreover, it prevents MIAs by applying *gradient ascent* or a *posterior flattening* step when the loss falls below a certain threshold. However, the latter generally happens during the last few rounds of training, which means that the early rounds of federated training will be still vulnerable for the attacker to differentiate between the member and non-member samples.

2.2. Adversarial training

Defenses based on adversarial training place a shadow attacker in the target model's training or inference process to regulate the distinguishability of prediction output for member and non-member samples. *Adversarial regularization* (Adv-Reg) (Nasr, Shokri, & Houmansadr, 2018) and *MemGuard* (Jia et al., 2019) work on this principle.

Adv-Reg incorporates a neural network-based attack model in the target model training. The attack classifier is first trained to estimate the privacy risk at each training step. Afterwards, the target model is retrained to minimize the prediction loss and mislead the attack classifier simultaneously. More specifically, the target model minimizes a weighted sum of cross-entropy and adversarial loss approximated from the attack classifier. This doubles the computational overhead to model training (because of training the attack model); the overhead gets worse for federated learning, where each client goes through several training rounds. Moreover, the privacy-accuracy trade-off of Adv-Reg is similar to that of the early stop regularization technique. The marginal privacy protection achieved is due to neural network-based attack models, which underestimate privacy risks due to challenges in choosing appropriate hyperparameters (number of hidden layers, learning rate, activation, functions) (Song & Mittal, 2021). In contrast, the accuracy drop comes from integrating adversarial loss as regularization.

MemGuard does not change the training process; instead, it blurs the target model's output scores at test time with well-designed adversarial noises in order to confuse the membership inference classifier without changing the predicted label. As a result, MemGuard can be an ideal defense in a scenario where the adversary is an external entity, and the parameters available from the model are limited to the output vector, prediction confidence, and loss. However, in FL, the adversary can be an insider (client or server) with complete white-box access to the model.

2.3. Differential privacy

Differential privacy (DP) provides membership privacy by obscuring an algorithm's output so it cannot disclose information about any particular user in the input data. It achieves this by adding noise to the computation process, which ensures that the presence or absence of any single user data has a negligible impact on the overall output. When properly implemented, DP provides statistical guarantees on the information an adversary can obtain from the output of a randomized algorithm (Dwork, Roth, et al., 2014).

The approaches to enforce DP in FL are *Local Differential Privacy (LDP)* and *Central Differential Privacy (CDP)*.

In LDP, each client either *perturbs her model update by injecting noise before sending it to the server* or *trains the local model on her private data by applying a differentially private stochastic gradient descent (DP-SGD) algorithm*. The latter approach, which adds noise to the gradients during each local training step, is the most popular one, as it is implemented in the Tensorflow-Privacy framework.[1] The problem is that the neural network's gradients are unbounded, which entails an unbounded sensitivity and noise to attain DP. To fix this problem, DP-SGD clips gradients to a threshold before adding noise, which slows down or biases the learning process. Besides, DP-SGD computes gradients on a random subsample instead of on entire private data, also making the learning process slower or biased (Blanco-Justicia, Sánchez, Domingo-Ferrer, & Muralidhar, 2022). Moreover, subsampling may cause an outlying individual's contribution to be easily noticeable, therefore delivering insufficient privacy protection for that individual (Dwork, 2011).

DP-SGD is rarely enforced in a strict way, as this would likely produce unusable models. Instead, a relaxed version ((ϵ, δ)-DP) is used, which allows the possibility of privacy breaches with a probability δ. However, even (ϵ, δ)-DP-SGD with safe enough parameters (*i.e.*, $\epsilon < 1$ and $\delta \ll 1/n$, being n the size of the sample Dwork, 2011) fails to keep the model's utility to a meaningful level (Blanco-Justicia et al., 2022). In an attempt to preserve the model's utility, the literature has broadly used DP-SGD with unsafe privacy parameters (seeTable 1), but this comes at the cost of losing any meaningful privacy guarantees that DP may provide (Blanco-Justicia et al., 2022; Jayaraman & Evans, 2019).

The lack of privacy guarantees that arises from unsafe ϵ values is formally demonstrated in Yeom, Giacomelli, Fredrikson, and Jha (2018), where the theoretical upper bound on possible leakage in DP-ML against MIA is defined in terms of the *adversary's advantage* (*i.e.*, the difference between the adversary's true positive and false positive rates). This work proves that the *adversary's advantage* against an ϵ-DP model is upper-bounded by $e^{\epsilon} - 1$. Therefore, for $\epsilon > 1$, the *adversary advantage* is greater than 1, indicating no theoretical protection guarantees (Blanco-Justicia et al., 2022). In this respect, Jayaraman and Evans (2019) further show that relaxed (ϵ, δ)-DP with weak privacy parameters has a practical leakage similar to *privacy protection by plain noise addition*, a 40-year-old technique criticized for its poor privacy/utility trade-off (Paass, 1988; Tendick, 1991). On top of that, all works use δ values very close or even equal to $1/n$, which implies that ϵ-DP is not satisfied at least for one data point. With such weak privacy parameters, the theoretical guarantees of DP are so diminished that the model should not be considered as DP-protected (Dwork, Kohli, & Mulligan, 2019).

Calibrating DP parameters is also challenging. The noise needed for each training step is approximated by using the moments accountant method (Abadi et al., 2016) for a desired ϵ according to the training data size. Since a model is trained successively for many epochs on the same dataset (or on non completely disjoint subsets), the effective ϵ grows with training epochs. As a result, each epoch requires more noise injection into the model gradients to keep the effective ϵ under control. This extra noise slows down model convergence and demands more training epochs, thereby requiring more noise per epoch to achieve the target ϵ. This creates a vicious cycle that not only slows down training, but also makes achieving the desired ϵ difficult.

The compromised privacy protection is not the only problem with LDP: even when weakly applied, it still substantially hurts the model's utility, as shown in Table 1. Also, the usual non-iidness that characterizes clients in the FL setting makes some of them more easily distinguishable, which in turn demands more noise to obscure their influence in the model.

Finally, the typical implementation of LDP focuses on protecting clients' single data points (*e.g.*, records). However, quite often, all

[1] https://github.com/tensorflow/privacy.

Table 1
DP for federated deep learning: a comparison of recent works.

Reference	Dataset	Clients	Original accuracy	DP parameters	DP accuracy
Triastcyn and Faltings (2019)	MNIST (non-iid)	100	97%	$\varepsilon = 8$; $\delta = 10^{-3}$	78%
	MNIST (non-iid)	10,000	99%	$\varepsilon = 8$; $\delta = 10^{-6}$	96%
Triastcyn and Faltings (2019)	MNIST (iid)	100	97%	$\varepsilon = 8$; $\delta = 10^{-3}$	86%
	MNIST (iid)	10,000	99%	$\varepsilon = 8$; $\delta = 10^{-6}$	97%
Triastcyn and Faltings (2019)	APTOS 2019	100	70%	$\varepsilon = 8$; $\delta = 10^{-3}$	60%
	APTOS 2019	10,000	72%	$\varepsilon = 8$; $\delta = 10^{-6}$	68%
Naseri, Hayes, and De Cristofaro (2020)	MNIST	100	98%	$\varepsilon = 3$; $\delta = 10^{-5}$	62%
Naseri et al. (2020)	MNIST	100	98%	$\varepsilon =7.5$; $\delta = 10^{-5}$	82%
Naseri et al. (2020)	CIFAR10	100	94%	$\varepsilon = 2.5$; $\delta = 10^{-5}$	67%
Naseri et al. (2020)	CIFAR10	100	94%	$\varepsilon = 7.0$; $\delta = 10^{-5}$	79%
Wei et al. (2021)	FashionMNIST	10	–	$\varepsilon = 25$; $\delta = 0.001$	85%
Wei et al. (2021)	CIFAR10	10	–	$\varepsilon = 25$; $\delta = 0.001$	41%

the clients' data points originate from a single user (*e.g.*, fitness or health measurements over time), thereby making data points highly correlated. In that case, LDP offers only *instance-level* privacy, and not *client-level* privacy. The former can barely be called DP, as masking the presence or absence of a single data point fails to ensure client privacy when that sample exhibits a strong correlation with the other points in the client dataset (Blanco-Justicia et al., 2022; Triastcyn & Faltings, 2019). From the above discussion, we can argue that the typical implementation of LDP in FL fails to offer the theoretical privacy guarantees. Under these circumstances, an empirical privacy assessment should be conducted *ex post*, which is seldom done by related works.

The second approach for enforcing DP in FL, Central Differential Privacy, is more in line with the DP guarantee, as it ensures *client-level* privacy protection. In CDP, the server modifies the aggregation function to clip the L2 norms of clients' updates and injects Gaussian noise into the global model produced from the combination of local models. However, clients must blindly trust the server and send their updates in clear form. This makes clients vulnerable to MIAs orchestrated by *honest but curious* servers, which are the most common and powerful privacy attackers in the FL setting (Blanco-Justicia et al., 2022; Naseri et al., 2020).

2.4. Knowledge distillation

Knowledge distillation (KD) transfers knowledge from a complex (in terms of parameters) model – known as teacher – to another comparatively simple model – called student –. The technique was initially proposed for model compression. Distillation for membership privacy (DMP) (Shejwalkar & Houmansadr, 2021), knowledge cross distillation (KCD) (Chourasia et al., 2021), self-ensemble architecture (SELENA) (Tang et al., 2022), and private aggregation of teacher ensembles (PATE) (Papernot, Abadi, Erlingsson, Goodfellow, & Talwar, 2016) are based on the KD principle.

DMP is vanilla KD, except it trains the student model by using the output of a teacher model obtained on an unlabeled reference dataset instead of the same (private) dataset as the teacher model. KD first trains the teacher (private) model on sensitive personal data and applies it to a public reference dataset to obtain a softmax output for each sample in the reference dataset. The student model is then trained on the public reference set, but using the softmax output of each sample, which is finally employed as the public model (the client sends this model to the server in FL). However, such public datasets are unavailable in many domains like healthcare, thereby making DMP hardly applicable.

KCD overcomes DMP's dependency on the public reference dataset by using a portion of the private set as a reference set. It divides the private training dataset into n equal disjoint subsets and trains n teacher models where the ith model is trained on all subsets except the ith subset. Finally, KCD trains the student model on the entire private dataset, generating the soft labels for the ith subsets by exploiting the ith teacher model. However, in FL, the clients' dataset size is generally small and heterogeneous. Splitting such a dataset into multiple subsets is impractical since the model trained on each subset will be underfitted and distinct. Distilling knowledge from these models produces a public (student) model with barely any utility. On this ground, KCD is not applicable in FL, even though it performs well in a centralized setting.

SELENA overcomes the dependency on reference public datasets by exploiting the self-distillation strategy, where the same private dataset distills the knowledge. SELENA randomly divides the private training set into overlapping subsets and trains a model on each. The prediction for each sample in the original training set is the aggregation of the predictions of models that did not contain that sample in their training data. However, training samples and their corresponding soft labels generated by combined prediction comprise the reference set on which the protected model is trained. Its applicability to FL is also questionable due to the same reason as KCD.

PATE combines KD and DP in local model training. It first trains multiple private (teacher) models on disjoint subsets of sensitive data and then trains a public (student) model on a public dataset labeled by the noise-added ensemble of teacher decisions. However, its applicability in FL is hampered by its requirements of a *public dataset* and *multiple teacher models*. Whereas the first requirement cannot always be met in sensitive domains like healthcare, the second requirement can hardly be satisfied for FL clients with small local datasets. Moreover, PATE enforces LDP on the softmax output of the teacher models, which only ensures *instance-level* privacy at the cost of a significant utility loss (as discussed in Section 2.3).

FCCL (Huang, Ye, & Du, 2022), FedMD (Li & Wang, 2019), and FedDF (Lin, Kong, Stich, & Jaggi, 2020) also utilize the KD strategy and are specifically designed to address three primary challenges in FL: *data heterogeneity*, *model heterogeneity*, and *catastrophic forgetting*. However, these approaches are not effective at protecting against MIAs in FL due to the following reasons.

In FedMD, each client first trains its personalized local model on a public dataset (known to all clients) and then fine-tunes it on the private data. Subsequently, the client computes the class score on the public dataset and transmits the results to the central server. Upon receiving the class scores from all the clients, the server takes their average and broadcasts the results to the participating clients. Each client then updates its local model to approach the mean class score by exploiting the public and private datasets. However, usable public datasets are barely available in privacy-sensitive domains such as healthcare and finance. Furthermore, preparing the public dataset for FedMD requires careful deliberation and prior knowledge of clients' private data to ensure that the public data includes samples from all classes present in the private data. This dependency on public datasets makes FedDM challenging to apply in FL settings, where the properties of all clients' private data must be considered.

FedDF performs ensemble KD on an unlabeled public dataset at the server end to address the heterogeneity of clients' models without considering data privacy issues. The clients return the local models to the server in plain form without privacy safeguards, similar to vanilla KD. As a result, a malicious or honest-but-curious server can effectively conduct privacy attacks. Therefore, FedDF is not suitable for preserving clients' data privacy.

FCCL exploits KD to address the catastrophic forgetting problem in federated continual learning. The KD strategy in FCCL helps the local models retain the knowledge gained in previous training rounds from other clients while updating them on the private data. FCCL incorporates dual-domain distillation loss with cross-entropy loss as the regularization term. Dual-domain distillation loss comprises two components: inter-domain distillation loss and intra-domain distillation loss. The former helps to learn continuously from other clients in the federation, while the latter encourages learning from itself. However, after updating the model on private data by using an optimization algorithm that minimizes the overall training loss, the updated local model is returned to the server without adopting any privacy protection mechanism. Therefore, the local models are vulnerable to privacy attacks either by the server or the clients, which makes FCCL ineffective for privacy-preserving FL.

3. Federated learning threat model

In FL, a server S coordinates the joint learning process of a deep neural network ω by K clients $\{C_1, C_2, C_3, \dots, C_K\}$. These clients possess private datasets $\{D_1, D_2, D_3 \dots, D_K\}$ that they aim to utilize for training ω at their premises. During each training round $t \in [1, T]$, the server S sends the current global model ω^{t-1} to all clients. Upon receiving ω^{t-1}, each client C_k updates ω^{t-1} using their respective private dataset D_k by employing the stochastic gradient descent (SGD) algorithm to generate the local model ω_k^t. Subsequently, all clients return their local models (rather than their own data) back to S, which aggregates them to produce an improved global model ω^t. The most usual aggregation algorithm is federated averaging (FedAvg) (McMahan et al., 2017):

$$\omega^t = \sum_{k=1}^{K} \frac{|D_k|}{|D|} \omega_k^t,$$

where $|D_k|$ represents the dataset size of client C_k, and $|D| = \sum_{k=1}^{K} |D_k|$, represents the total dataset size across all clients. After repeating the process for T rounds, the final global model ω^T, trained on all clients' private datasets is produced.

In this work, we consider both the server S and the clients $\{C_1, C_2, C_3, \dots, C_K\}$ as being *honest but curious* adversaries who strictly follow the FL protocol but perform MIAs on local (the server is the global adversary) and global models (clients are the local adversaries) at each training round. We consider a black-box attack scenario where the attacker only queries the model and obtains corresponding prediction vectors or labels.

Like the previous defenses (Nasr et al., 2018; Song & Mittal, 2021; Tang et al., 2022), we assume strong global and local adversaries who know a non-member dataset and a subset of training (member) samples as prior knowledge from which they can train NN-based attack classifiers or learn (loss or entropy) threshold functions to identify other member samples. More specifically, the global adversary has data from each client training set to develop client-wise attack models, while local adversaries develop attack models considering data from all clients in the federation.

4. Analysis of model behavior for member and non-member samples

To design an effective defense against MIAs, we first analyze the differences in model behavior for member and non-member samples that adversaries may exploit to perform successful attacks.

4.1. Theoretical analysis

To understand the sources of membership inference attacks from an analytical viewpoint, let us consider an FL classification task where each local model is trained by minimizing the cross-entropy loss between the ground truth distribution y and an estimated probability distribution p over C classes for local training samples. In practice, the ground truth distribution y is a one-hot encoding of class labels, where the probability is $y_a = 1$ for ground truth class label a, and $y_c = 0$ for class label $c \neq a$. During training, the local model's last layer output o is first fed into the softmax function to estimate p as follows:

$$p_i = \frac{e^{o_i}}{\sum_{j=1}^{n} e^{o_j}}, \quad i \in \{1, 2, \dots, C\}.$$

Then, the cross entropy loss between p and y is computed as

$$\ell = -\sum_{c=1}^{C} y_c \log p_c.$$

After that, the gradient of loss w.r.t. the ith neuron output o_i in the classification layer is computed as

$$\begin{aligned} \delta_i &= \frac{\partial \ell}{\partial o_i} = -\sum_c y_c \frac{\partial \log p_c}{\partial o_i} = -\sum_c y_c \frac{1}{p_c} \frac{\partial p_c}{\partial o_i} \\ &= -y_i(1 - p_i) - \sum_{c \neq i} y_c \frac{1}{p_c}(-p_c p_i) \\ &= p_i \left(\sum_c y_c \right) - y_i = p_i - y_i, \end{aligned} \tag{1}$$

where $\sum_c y_c = 1$, as y is a vector with only one non-zero element, which is 1. It is noteworthy that δ_i is bounded between -1 and 1. On the one hand, δ_i is consistently within the interval $[-1, 0]$ when the ith neuron corresponds to the ground truth class a, that is, when $y_a = 1$. Conversely, in instances where the ith neuron corresponds to an incorrect class ($y_c = 0$ and $c \neq a$), δ_i consistently takes values within the interval $[0, 1]$.

The gradient δ_i quantifies the model's error in estimating the probability for the ith class, which is expected to be 0 to enhance performance. Eq. (1) indicates that the cross-entropy loss for a sample x with ground truth class a touches the minimum point when $p_c = 1$ for $c = a$, and 0 otherwise.

This minimum is not attainable for finite o_i, but is approached if $o_a \gg o_c$ for all $c \neq a$ —if the output corresponding to the ground-truth class is much greater than all other outputs. This causes two problems. First, it may result in overfitting: if the model learns to assign the entire probability to the ground truth label for each training example, generalization will be affected (Hinton, Vinyals, & Dean, 2015; Pereyra et al., 2017; Szegedy et al., 2016). Second, it favors the differences between the output connected to the ground truth class and all the other classes to become larger, and this, combined with the bounded gradient δ_i, reduces the model's ability to adapt (Szegedy et al., 2016). *Intuitively, this occurs because of training with a one-hot encoded ground truth distribution over classes.*

Consequently, the model becomes overconfident in its prediction for training samples, placing the entire probability mass on the ground truth class, and diminishing the probabilities for the other classes to zero. Conversely, since the model does not explicitly minimize the cross entropy loss for non-training samples, it shows divergent behaviors or patterns for them. Specifically, for non-training samples, the model assigns the highest probability to the correct class, and leaves at least some probability to incorrect classes, thereby exhibiting statistical differences in prediction confidence and losses for member and non-member samples. Differences in prediction confidences also create discrepancies in prediction entropies and gradient norms, as they are directly associated with the prediction confidence: the entropy for training samples is lower than for non-training samples as the model is more confident in training samples, which indicates less uncertainty.

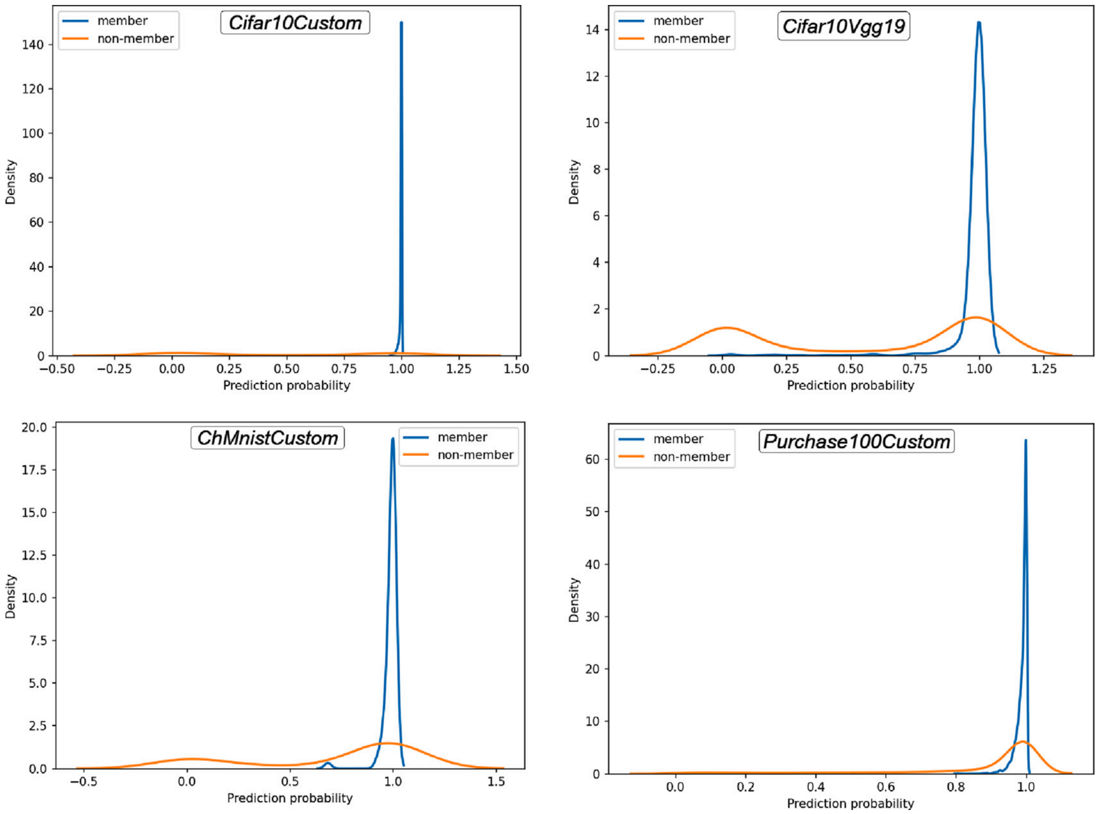


Fig. 1. Distribution of the actual class probability for member and non-member samples across four tasks (*Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom* and *Purchase100Custom*) using vanilla FL. The member and non-member distributions are represented with blue and orange colors, whereas the overlap between both distributions is represented in brown (blue+orange).

Therefore, the gradient norm leans more towards zero for training samples than for non-training samples.

Connecting the dots from the above discussion, we can conclude that *training a local model with a one-hot encoded ground truth distribution over classes makes it overfitted on the data, and eventually shows statistical differences in prediction confidence, loss, entropy, and gradient norm between training (member) and non-training (non-member) samples that an adversary can exploit to perform membership inference attacks*. Our analytical insight is in line with the previous works (Song & Mittal, 2021; Yeom et al., 2018).

4.2. Empirical analysis

To empirically validate the argument in the previous section, in the following we simulate an FL scenario for ten training rounds with five clients under non-iid data distributions across *Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom*, and *Purchase100Custom* tasks (see Section 6).

We considered the *global adversary* described in Section 3 (an *honest-but-curious* server that performs MIAs on the local models after each training round before aggregation) for this empirical study. For each client, we analyzed the behavior of the local model that gives maximum attack advantage to the global adversary on the member and non-member samples.

We first investigated the model's *prediction confidence*. Specifically, the prediction confidence of a model f on sample x is $p_a = f_a(x)$, which indicates the prediction probability of the ground truth class a for input x. Fig. 1 shows the distributions of the prediction confidence over member and non-member samples across all the tasks. At first glance, it is clear that the prediction confidence is nearly 1 for members, indicating a high certainty. In contrast, the confidence varies across a wide range for non-members, confirming our analytical findings.

We further compared the *uncertainty* of the model's output for members and non-members. We considered different uncertainty quantification techniques, including *traditional information entropy* and *modified entropy*. However, we found that the modified entropy difference between member and non-member samples was more perceptible than other uncertain measure techniques, which suggests that MIAs based on the modified entropy threshold could be more aggressive. Therefore, we quantified the uncertainty of the prediction vector p for each member and non-member sample by the modified entropy $-(1-p_a)\log p_a + \sum_{j\neq a} p_j \log(1-p_j)$, which takes into account the ground truth label a. Fig. 2 shows the modified entropy distributions as histograms, where two distributions over member and non-member samples are distinguishable across all the tasks. With a slight overlap in the lower range, the non-member histogram shows higher frequencies in the upper range, indicating a divergence from the member histogram.

Finally, we studied the gradients of loss of the local models' predictions. Gradients were computed w.r.t. the parameters of the model. Fig. 3 shows the distributions where the number of members and non-members given on y-axes falls in a particular range of gradient norm values given on x-axes. We note that the distribution of the gradient norms of the model is heavily skewed to the left for the members, *i.e.*, towards lower gradient norm values, unlike that for the non-members, which indicates that the model has memorized training samples. This minimal gradient norm on the members compared to the non-members gives the adversary an advantage to perform MIA in white-box settings, which is the case of FL.

We can conclude from the theoretical and empirical analyses above that model overfitting is the underlying cause of membership privacy leakage. It happens due to training a model with one-hot encoded

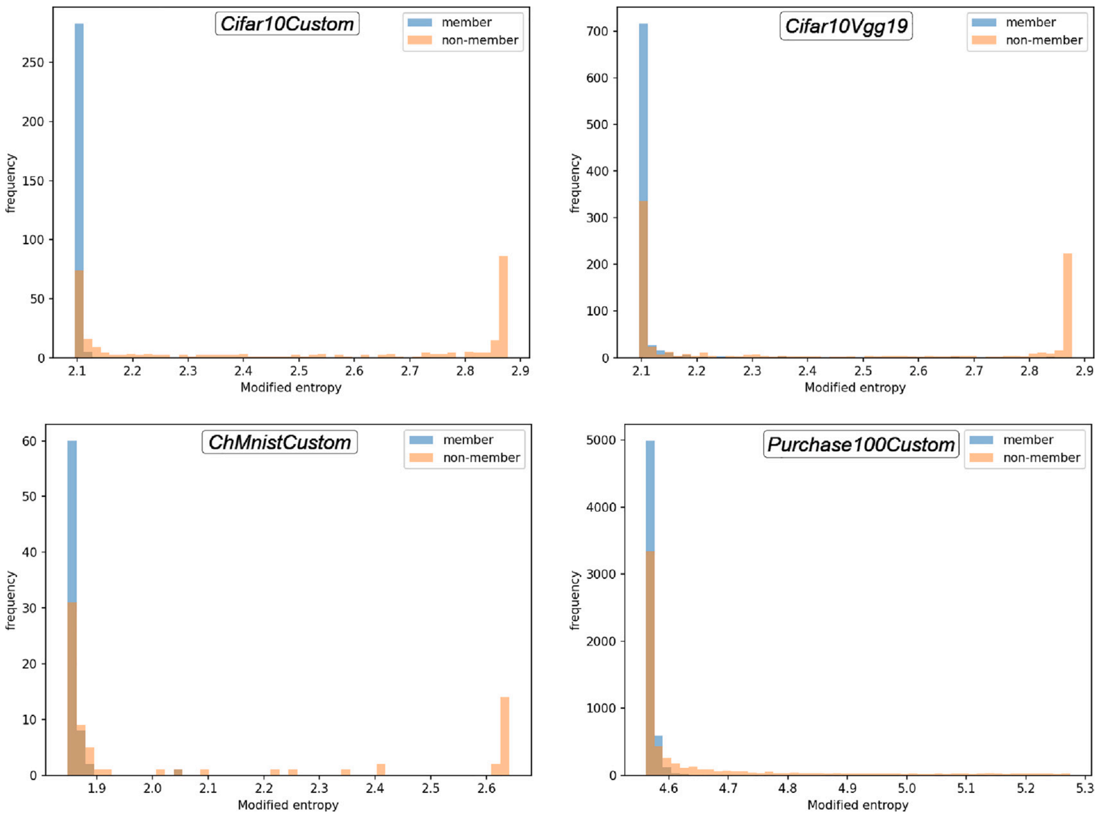


Fig. 2. Distribution of the modified entropy for member and non-member samples across four tasks (*Cifar10Custom*, *Cifar10Vgg19*, *ChMinstCustom* and *Purchase100*) using vanilla FL. The member and non-member distributions are represented with blue and orange colors, whereas the overlap between both distributions is represented in brown (blue+orange).

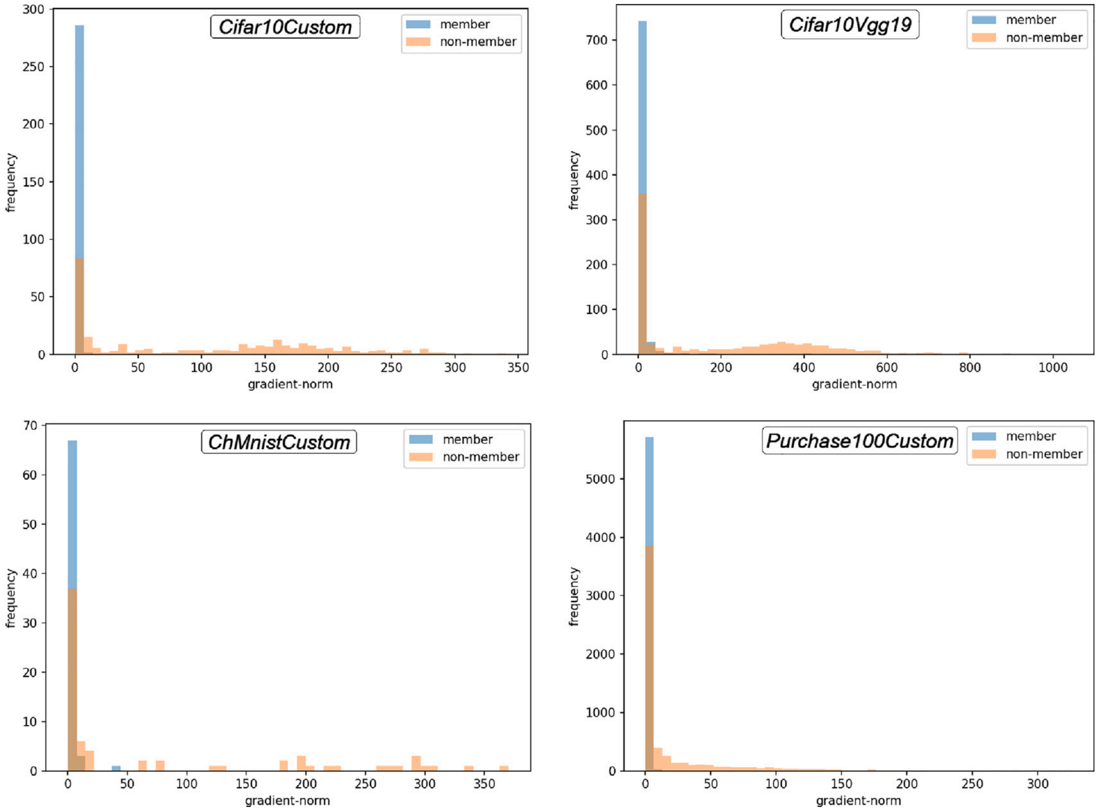


Fig. 3. Distribution of the gradient norm for member and non-member samples across four tasks (*Cifar10Custom*, *Cifar10Vgg19*, *ChMinstCustom* and *Purchase100*) using vanilla FL. The member and non-member distributions are represented with blue and orange colors, whereas the overlap between both distributions is represented in brown (blue+orange).

labels, which forces a model to place all its confidence in the ground truth class without assigning any probability to wrong classes for training samples. This eventually creates differences in the target model's behavior on member and non-member samples. The attacker uses this gap to perform successful MIAs. These findings also suggest that membership privacy can be preserved by controlling over-confidence in prediction during training, to make the prediction confidence distribution identical for member and non-member samples.

5. Our solution

This section presents our solution – called MemberShield – for preventing MIAs in FL, which is grounded on the analyses in the previous section. MemberShield aims are (i) to be suitable for FL environments, (ii) to be model- and dataset-agnostic, (iii) to be client number-independent, (iv) to provide a better privacy/utility trade-off than related works, and (v) to incur minimal computational overhead during training.

The previous section has shown that the overfitting exploited by MIAs primarily arises from the overconfidence of the model in predicting and memorizing training samples. Whereas overconfidence is a by-product of training a model with one-hot encoded ground truth labels, memorization happens due to excessive training.

Our defense overcomes these issues by generalizing the local model in two steps: *(1) regulating the prediction confidence by training the model with soft-encoded ground truth labels*, and *(2) mitigating training sample memorization by early stopping considering global and local model perspectives.*

Regulating the prediction confidence. This step encourages the model to show less confidence in the training sample prediction, which contributes to making confidence, entropy, and loss distributions for member and non-member samples less distinguishable. This step also helps the model to generalize better, by allowing the model to assign some probabilities to incorrect classes (Hinton et al., 2015; Pereyra et al., 2017).

The knowledge distillation (Hinton et al., 2015) and anti-overfitting techniques – such as confidence penalty (Pereyra et al., 2017) and label smoothing (Szegedy et al., 2016) – discussed in Section 2 work on this principle to generalize a model. They penalize overconfident predictions during training by incorporating a regularization term in the loss function that prevents a model from putting all the probability mass on a single class in the training set. However, their goal of generalization targets boosting accuracy, rather than preserving privacy. As a result, they cannot control how the model should distribute the probability mass among incorrect classes. It may be the case that only a small set of incorrect classes (rather than all incorrect classes) is assigned non-zero probability. In other words, such techniques fall short of making the output distributions of members and non-member samples indistinguishable.

Our method targets indistinguishability by training the model with soft-encoded ground truth labels instead of one-hot encoded ground truth labels. This step requires each client to preprocess her private data to transform one-hot encoded ground truth labels into soft-encoded ground truth labels as explained next.

Consider a uniform prior distribution u over levels $c \in \{1, \ldots, C\}$ where $u_c = 1/C$ for each c, independent of the training example x. We replace the one-hot encoded ground truth distribution y with a new probability distribution y' given by

$$y'_c = (1-\theta) \cdot y_c + \theta \cdot u_c \tag{2}$$

for each label $c \in \{1, \ldots, C\}$. For the new distribution, $\sum_{c=1}^{C} y'_c = 1$ and $1 > y'_a > y'_c > 0$ for the ground truth label a and all $c \neq a$. The new ground truth distribution replaces the hard 0 and 1 class labels with θ/C and $1 - \frac{C-1}{C} \cdot \theta$, respectively, thereby mirroring the prediction probability distributions for unseen data.

This soft distribution can be interpreted as a mixture of a one-hot encoded distribution y and a uniform prior distribution u with weights θ and $1-\theta$, respectively. In other words, Eq. (2) transforms a one-hot encoded ground truth distribution to a probability distribution by allocating the maximum confidence score for the ground truth class and uniformly distributing the remaining confidence score to incorrect classes. In this way, it prevents focusing on hard probabilities, without discouraging correct classification.

Mitigating training sample memorization. Due to the usual heterogeneous data distribution across the peers in FL settings, some local models may converge earlier than the expected number of federated training rounds. Specifically, a local model stops enhancing its generalization capability when its validation loss fails to exhibit improvement w.r.t. the validation loss of the global model from the preceding local training round. In fact, from that moment onwards the local model starts memorizing local training samples, thereby becoming vulnerable to membership leakage.

To prevent such unnecessary overtraining, we include an early stopping regularization into the local model training. Our early stopping criterion differs from traditional early stopping, where training stops when a model's training or validation loss does not change significantly for a predefined number of consecutive local epochs. With that approach, the local model does not consider the global perspective for stopping. In our approach, the peer evaluates the model's loss on its internal validation data after receiving the aggregated global model from the server, and starts local training by tracing the validation loss for successive epochs. More specifically, during the first local training epoch, the peer compares her local model's validation loss with the global model's loss. In successive epochs, the peer compares the local model's validation loss with that of the previous epoch. The training stops when the local validation loss does not improve for a number of consecutive epochs.

We formalize our method in Algorithm 1. At the beginning of the FL training (lines 1–2), all peers preprocess their local training (X_{train}, Y_{train}) and validation (X_{val}, Y_{val}) data to transform one-hot encoded ground truth labels to soft encoded ground truth labels using Algorithm 2. It is worth noting that each peer performs this step only once during the entire FL training. During each training round t, upon receiving the current global model w_{t-1}, each peer calculates the validation loss $loss_{global}$ of w_{t-1} on their internal validation set (X_{val}, Y_{val}) using Algorithm 3 (line 5). Subsequently, the peer starts training w_{t-1} on their private dataset X_{train} with the corresponding soft labels Y_{train} (lines 7–21). Throughout the local training process, the peer compares the validation loss of epoch e with that of epoch $(e-1)$ except for the first epoch, where the comparison is made against $loss_{global}$ (lines 12–17). The peer stops local training when there is no improvement in the model's validation loss for $count_{max}$ consecutive epochs (lines 18–20) and returns the updated local model, w_k^t, to the server.

6. Experimental setup

In this section, we evaluate the effectiveness of MemberShield against a variety of membership inference attacks and compare its performance with related defenses across various datasets of diverse modalities and classification models.

We simulated an FL setting for each classification task for ten training rounds with five clients under non-iid data distributions. The experiments were conducted on an Intel Xeon Silver 4114 processor with 64 GB of RAM and an NVIDIA A10 GPU with 24 GB of VRAM operating on Ubuntu 22.04.3 LTS. Python version 3.10.12, CUDA version 12.2, and TensorFlow version 2.12.0 were utilized to construct the deep learning models.

We used the TensorFlow-Privacy version 0.8.12 implementations of the moments accountant method, the DP-SGD training algorithm,

Algorithm 1 MemberShield: Local model training in FL with membership privacy

```
Input: Global model weight $\omega^{t-1}$ at global round $t-1$
Output: Local model weight $\omega_k^t$ trained on client $C_k$ at global round $t$
$X_{\text{train}}, Y_{\text{train}} \leftarrow \texttt{SoftLabel}(X_{\text{train}}, Y_{\text{train}}, \theta)$
$X_{\text{val}}, Y_{\text{val}} \leftarrow \texttt{SoftLabel}(X_{\text{val}}, Y_{\text{val}}, \theta)$
$B_{\text{train}} \leftarrow$ (split $(X_{\text{train}}, Y_{\text{train}})$ into $m$ batches of size $B$)
$w \leftarrow \omega^{t-1}$
$loss_{\text{global}} \leftarrow \texttt{Evaluate}(w, X_{\text{val}}, Y_{\text{val}}, B)$
initialize $count_{\text{max}}$
for each local epoch $e \in [1, E]$ do
    for each batch $b \in B_{\text{train}}$ do
        $w \leftarrow w - \eta \cdot \Delta\ell(w; b)$
    end for
    $loss_{\text{local}} \leftarrow \texttt{Evaluate}(w, X_{\text{val}}, Y_{\text{val}}, B)$
    if $loss_{\text{local}} \geq loss_{\text{global}}$ then
        $count \leftarrow count + 1$
    else
        $loss_{\text{global}} \leftarrow loss_{\text{local}}$
        $count \leftarrow 0$
    end if
    if $count \geq count_{\text{max}}$ then
        break
    end if
end for
$\omega_k^t \leftarrow w$
```

Algorithm 2 Generating soft labels

```
Function SoftLabel$(X, Y, \theta)$:
$Y' \leftarrow \{\}$
$X' \leftarrow \{\}$
for each training sample $(x, y) \in (X, Y)$ do
    for each class label $c$ in $[0, C-1]$ do
        $y'_c = (1-\theta) \cdot y_c + \frac{\theta}{C}$
    end for
    $Y' \leftarrow Y' \cup y'$
    $X' \leftarrow X' \cup x$
end for
return $X'$, $Y'$
```

Algorithm 3 Evaluate model on dataset

```
Function Evaluate$(w, X, Y, B)$:
$\beta \leftarrow$ (split $(X, Y)$ into $n$ batches of size $B$)
$loss \leftarrow 0$
for each batch $b \in \beta$ do
    $loss \leftarrow loss + \ell(w; b)$
end for
$loss \leftarrow \frac{loss}{n}$
return $loss$
```

and MIAs. To reproduce the results, our code and data are available at https://github.com/faisalahm3d/MemberShield.

6.1. Datasets

We used three well-differentiated datasets that have been widely employed for evaluating MIAs and their defenses:

- *CIFAR10* is a benchmark dataset for image classification tasks containing 60,000 color images of 32 × 32 pixels clustered into ten classes. The dataset has train and test splits, comprising 500 and 100 images per class. We normalized the image pixel value to a mean of 0 and a standard deviation of 1.
- *CH-MNIST* contains 5000 grayscale images of eight tissue types collected from colorectal cancer patients. We employed the preprocessed version from Kaggle, where each image is 64 × 64 pixels and normalized to $[-1, 1]$.
- *Purchase100* includes customers' shopping records released by the Kaggle Acuaire Valued Shopper Challenge. We used the preprocessed version of the dataset provided by Shokri et al. (2017), which contains 197,324 records. Each record has 600 binary features indicating whether the user purchases a specific item. The records are grouped into 100 classes representing the customers' shopping styles.

We distributed each dataset across the five clients of the simulated FL setting by adapting the Dirichlet distribution (Minka, 2000) with the hyper-parameters $\alpha = 1$ as in Hsu, Qi, and Brown (2019, 2020) and Jebreel and Domingo-Ferrer (2023). The α value controls the degree of non-IIDness, where $\alpha = \infty$ indicates identical local data distribution, and smaller α corresponds to proportionally larger non-IIDness (Gong et al., 2022). We further split each client's data into train and test with an 80:20 ratio with stratification, in order to preserve the same IID-distribution and the percentage of samples for each class in both splits.

6.2. Target models

For the CIFAR-10 dataset, we employed two distinct neural network architectures: a 19-layer VGG model and a custom two-layer convolutional neural network. The latter incorporates two convolutional layers with filter sizes $[20, 50]$ and kernel sizes $[(5, 5), (5, 5)]$. Each convolutional layer passes through a 2D max polling layer. After the convolutional blocks, a fully connected layer comprising 500 neurons is appended. Activation functions applied to neurons within each layer are rectified linear units (ReLU), except for the classification layer, which employs the softmax activation function.

For the Purchase100 dataset, we adopted the same architecture as in Nasr et al. (2018): a four-layer fully connected network with layer size $[1024, 512, 256, 128]$. We used the Tanh activation function and initialized the weights and bias with a random normal distribution with a mean 0 and a standard deviation 0.01.

For CH-MNIST datasets, we used a convolutional neural network consisting of three convolutional layers with filter sizes $[128, 64, 64]$ and corresponding kernel sizes $[(5, 5), (3, 3), (3, 3)]$. After each convolutional layer, a 2D max pooling layer with a pool size of (2,2) was applied. Convolutional layers were followed by three fully connected layers of sizes $[256, 64, 32]$. We used the ReLU activation function and initialized parameters with a random normal distribution having mean 0 and standard deviation 0.01.

In the remainder of the paper, we refer to the classification task of the *CIFAR10*, *CH-MNIST*, and *Purchase100* datasets with the custom models as *Cifar10Custom*, *ChMnistCustom*, and *Purchase100Custom*, respectively. Similarly, *CIFAR10* classification with the VGG19 model will be called *Cifar10Vgg19*.

We used the stochastic gradient descent algorithm for each local model training with a learning rate 0.001, momentum 0.99, batch size 200, and no weight decay, irrespective of datasets and model. In each FL round across all tasks, the clients trained their local model for 50 epochs, except for *Purchase100Custom*, which was trained for 10 local epochs because the model architecture for this task is shallow, consisting of only fully connected layers, and the dataset is a tabular record that requires much fewer training steps to learn.

For all the experiments with MemberShield, we set the weights for one-hot encoded labels (θ) to 0.7 for *Cifar10Vgg10* and *ChMnistCustom*, and 0.8 for *Purchase100Custom* and *Cifar10Custom* while generating soft labels. This hyperparameter needs to be tuned depending on the

Table 2
For vanilla FL, six baseline defenses, and MemberShield on the *Cifar10Custom* task: the global adversary's auR and maximum advantage (Adv) against each client local model; the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
Vanilla FL (No defense)	–	–	0.84	0.71	0.81	0.58	0.83	0.63	0.84	0.64	0.86	0.69	0.89	0.64	1973.62
Early stopping	–	–	0.61	0.19	0.60	0.17	0.58	0.13	0.58	0.14	0.59	0.18	0.69	0.61	1215.48
AoF	0.25	–	0.88	0.74	0.82	0.56	0.86	0.62	0.87	0.67	0.91	0.82	0.89	0.65	2117.64
	0.50	–	0.88	0.73	0.83	0.54	0.86	0.62	0.86	0.63	0.92	0.78	0.89	0.65	2080.81
	0.75	–	0.89	0.68	0.82	0.50	0.84	0.54	0.85	0.59	0.92	0.75	0.87	0.65	2015.21
	–	L2	0.87	0.66	0.71	0.33	0.81	0.52	0.83	0.59	0.90	0.77	0.66	0.61	2021.10
	0.25	L2	0.85	0.60	0.67	0.26	0.78	0.43	0.80	0.49	0.88	0.71	0.72	0.66	2058.35
	0.50	L2	0.83	0.56	0.67	0.25	0.75	0.41	0.78	0.45	0.87	0.65	0.75	0.68	2083.54
	0.75	L2	0.81	0.52	0.60	0.16	0.70	0.32	0.73	0.38	0.86	0.64	0.70	0.65	2058.80
DP	0.1	–	0.52	0.09	0.51	0.02	0.52	0.04	0.52	0.04	0.54	0.14	0.09	0.09	3068.95
	0.5	–	0.53	0.06	0.51	0.03	0.52	0.04	0.52	0.04	0.54	0.10	0.12	0.12	3066.95
	1	–	0.53	0.08	0.51	0.02	0.52	0.03	0.52	0.04	0.57	0.14	0.11	0.11	3044.48
	2	–	0.53	0.07	0.52	0.03	0.52	0.04	0.52	0.06	0.56	0.15	0.15	0.15	3022.81
	4	–	0.52	0.06	0.51	0.03	0.52	0.04	0.53	0.05	0.56	0.14	0.19	0.19	3028.14
	8	–	0.52	0.10	0.51	0.03	0.52	0.05	0.51	0.03	0.55	0.11	0.19	0.19	3019.60
	16	–	0.53	0.07	0.51	0.03	0.52	0.05	0.52	0.06	0.53	0.16	0.20	0.20	3014.77
	100	–	0.53	0.07	0.51	0.03	0.51	0.05	0.53	0.05	0.53	0.11	0.23	0.23	3029.29
	1000	–	0.53	0.07	0.51	0.02	0.52	0.05	0.52	0.07	0.52	0.10	0.26	0.26	3024.98
KD ($\alpha = 0.5, T = 10$)	–	–	0.85	0.59	0.75	0.39	0.81	0.50	0.86	0.62	0.84	0.55	0.48	0.38	4757.52
RelaxLoss ($\alpha = 0.4$)	–	–	0.80	0.53	0.79	0.49	0.80	0.50	0.81	0.54	0.81	0.50	0.85	0.56	1918.79
Adv-Reg ($\lambda = 3$)	–	–	0.70	0.33	0.62	0.18	0.65	0.23	0.62	0.20	0.74	0.39	0.18	0.17	3595.63
MemberShield (Our defense)	–	–	0.56	0.11	0.60	0.16	0.58	0.13	0.56	0.11	0.57	0.14	0.72	0.66	1780.29

complexity of the model and dataset. We experimented with different values in the range $[0.6, \dots, 0.9]$ for all the tasks using grid search and chose the value that gives the best privacy-utility trade-off. However, we observed no substantial difference in the privacy-utility trade-off for the values in this range. To enforce the early stopping step, we considered stable validation losses for 3 consecutive epochs for the *Purchase100Custom* task and 10 epochs for the rest of the tasks. These values are widely used to enforce early stopping regularization in simple and complex learning tasks, respectively (Goodfellow, Bengio, & Courville, 2016). Since local models for *Purchase100Custom* were trained for 10 epochs and monitoring 10 epochs would mean no early stopping, we monitored validation loss of only 3 epochs for this task. For other tasks, models were trained for 50 epochs and 10 epochs were monitored.

6.3. Evaluation metrics

Defenses were evaluated under the following three perspectives:

- *Utility* was measured in terms of the final global model's accuracy on the test set.
- *Defense effectiveness* (*i.e.*, privacy protection) was evaluated as the maximum attacker's advantage (*i.e.*, the difference between the adversary's true and false positive rates in MIA) and the highest area under the receiver operating characteristic curve (auR) obtained throughout all attacks across all FL rounds.
- The *training overhead* incurred by the defenses was measured in terms of the overall time needed to complete the FL process.

7. Empirical results

Tables 2, 4, 6, and 8 report the global adversary's maximum advantage against each client local model, as well as the global model's training accuracy, test accuracy, and training time for vanilla FL (without any defense mechanism), MemberShield, and six defenses in the literature on the *Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom*, and *Purchase100Custom* tasks, respectively. Similarly, Tables 3, 5, 7, and 9 report the local adversaries' maximum advantage against the global model for the same defenses and tasks. Additional results with concentration parameter $\beta \in \{0.5, 0.1\}$ for the Dirichlet distribution are reported in Appendix.

7.1. Comparison with vanilla FL

For vanilla FL, we see the advantages of global and local adversaries are large (around 0.5 or over 0.5) on all the tasks except *Purchase100Custom*, which indicates a privacy vulnerability. The advantages of local adversaries are comparatively lower than those of global adversaries as MIAs are performed against the global model where the local models are blended, and the contribution of client data tends to be unnoticeable due to it being hidden in the crowd.

In contrast, MemberShield effectively mitigates the membership privacy risks, yielding substantial reductions in the advantage of global and local adversaries across all the tasks. Such a high level of privacy protection comes from MemberShield's two main steps, which are carefully designed to suppress the distinction between the target model's behavior on member and non-member samples.

On top of that, our method incurs no or negligible utility drop and minimizes the training time. The test accuracy drop is highest on the *ChMnistCustom* task (from 73% to 67%) while it is boosted (from 64% to 66%) on the *Cifar10Custom* task. This decrease is lower than with other defenses, and it is more than compensated by the privacy gain.

The comparatively smaller drop incurred by MemberShield comes from its focus on preventing overfitting, which is a standard technique meant to improve model generalization. While preventing overfitting, MemberShield does not hamper the model's learning process and prediction decision learned from the labels (because the soft labels and one-hot encoded label express the same decision for a sample). The impact of generalization that MemberShield brings can be seen in Table 2, where our method improved on the accuracy of vanilla FL.

The reduction in training time stems from integrating the improved version of the early stopping step in local model training that considers local and global model accuracy.

7.2. Comparison with DP

For DP, we trained the local model with the DP-SGD algorithm, where the required noise multiplier σ for each client was estimated using the moment accountant method by supplying the expected training epochs, training data size, target ϵ, and δ. We considered values of $\epsilon \in [0.1, 0.5, 1, 2, 4, 16, 100, 1000]$, which include *safe* values (*i.e.*, $\epsilon \leq 1$, which

Table 3
For vanilla FL, six baseline defenses, and MemberShield on the *Cifar10Custom* task: local adversaries' auR and maximum advantage (Adv) against the global model.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5	
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
Vanilla FL (No defense)	–	–	0.69	0.38	0.74	0.43	0.73	0.41	0.77	0.49	0.77	0.51
Early stopping	–	–	0.53	0.08	0.57	0.12	0.55	0.08	0.58	0.13	0.56	0.13
AoF	0.25	–	0.72	0.37	0.76	0.47	0.73	0.40	0.80	0.56	0.83	0.60
	0.50	–	0.69	0.36	0.76	0.44	0.72	0.38	0.81	0.52	0.82	0.60
	0.75	–	0.66	0.28	0.73	0.37	0.67	0.31	0.79	0.46	0.79	0.52
	X	L2	0.56	0.13	0.55	0.09	0.55	0.10	0.58	0.13	0.59	0.18
	0.25	L2	0.57	0.13	0.56	0.09	0.55	0.08	0.58	0.13	0.60	0.17
	0.5	L2	0.55	0.10	0.55	0.10	0.55	0.10	0.59	0.14	0.60	0.18
	0.75	L2	0.55	0.12	0.54	0.08	0.55	0.08	0.56	0.11	0.59	0.16
DP	0.1	–	0.52	0.06	0.51	0.02	0.52	0.04	0.52	0.05	0.53	0.13
	0.5	–	0.53	0.06	0.51	0.03	0.52	0.05	0.51	0.05	0.55	0.10
	1	–	0.52	0.07	0.51	0.03	0.51	0.04	0.52	0.05	0.54	0.10
	2	–	0.53	0.07	0.52	0.04	0.52	0.04	0.52	0.06	0.57	0.11
	4	–	0.52	0.07	0.51	0.02	0.52	0.03	0.52	0.05	0.54	0.11
	8	–	0.53	0.08	0.52	0.03	0.52	0.05	0.53	0.05	0.57	0.10
	16	–	0.52	0.07	0.51	0.04	0.52	0.04	0.51	0.04	0.54	0.11
	100	–	0.55	0.08	0.52	0.03	0.51	0.03	0.52	0.05	0.54	0.11
	1000	–	0.53	0.08	0.51	0.02	0.51	0.04	0.52	0.04	0.54	0.11
KD ($\alpha = 0.5, T = 10$)	–	–	0.56	0.11	0.61	0.16	0.61	0.17	0.66	0.24	0.60	0.20
RelaxLoss ($\alpha = 0.4$)	–	–	0.66	0.28	0.78	0.46	0.79	0.47	0.80	0.52	0.73	0.43
Adv-Reg ($\lambda = 3$)	–	–	0.55	0.10	0.53	0.05	0.52	0.05	0.55	0.10	0.57	0.12
MemberShield (Our defense)	–	–	0.53	0.07	0.57	0.11	0.54	0.07	0.56	0.12	0.55	0.11

Table 4
For vanilla FL, six baseline defenses, and MemberShield on the *Cifar10Vgg19* task: the global adversary's auR and maximum advantage (Adv) against each client local model; the global model's training accuracy (Tr Ac), test accuracy (Te Ac) accuracy, and training time.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
Vanilla FL (No defense)	–	–	0.79	0.56	0.74	0.43	0.77	0.49	0.75	0.47	0.78	0.53	0.93	0.74	4154.52
Early stopping	–	–	0.59	0.19	0.62	0.19	0.59	0.15	0.67	0.28	0.65	0.27	0.84	0.72	2554.60
AoF	0.25	–	0.51	0.05	0.51	0.02	0.51	0.03	0.51	0.05	0.52	0.08	0.10	0.01	3994.83
	0.50	–	0.52	0.05	0.51	0.03	0.51	0.04	0.52	0.03	0.53	0.11	0.10	0.10	3985.51
	0.75	–	0.51	0.08	0.51	0.02	0.51	0.03	0.51	0.04	0.53	0.08	0.10	0.10	3988.30
	–	L2	0.51	0.06	0.51	0.02	0.57	0.13	0.51	0.04	0.54	0.10	0.10	0.10	4056.82
	0.25	L2	0.52	0.05	0.51	0.02	0.52	0.03	0.51	0.05	0.53	0.12	0.10	0.10	4079.61
	0.50	L2	0.52	0.06	0.51	0.02	0.51	0.04	0.51	0.05	0.54	0.10	0.10	0.10	4074.44
	0.75	L2	0.51	0.07	0.51	0.02	0.51	0.03	0.52	0.05	0.54	0.09	0.10	0.10	4077.73
KD ($\alpha = 0.5, T = 10$)	–	–	0.63	0.22	0.75	0.42	0.70	0.36	0.74	0.45	0.53	0.09	0.12	0.12	10 137.03
RelaxLoss ($\alpha = 0.4$)	–	–	0.79	0.49	0.73	0.40	0.76	0.46	0.77	0.50	0.74	0.49	0.90	0.71	3984.64
Adv-Reg ($\lambda = 3$)	–	–	0.69	0.29	0.71	0.34	0.68	0.28	0.66	0.25	0.65	0.28	0.31	0.25	6329.56
MemberShield (Our defense)	–	–	0.56	0.12	0.67	0.28	0.65	0.25	0.61	0.21	0.58	0.20	0.81	0.71	3172.35

Table 5
For vanilla FL, six baseline defenses, and MemberShield on the *Cifar10Vgg19* task: local adversaries' auR and maximum advantage (Adv) against the global model.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5	
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
Vanilla FL (No defense)	–	–	0.70	0.38	0.73	0.40	0.71	0.38	0.73	0.43	0.72	0.45
Early stopping	–	–	0.70	0.38	0.73	0.42	0.72	0.39	0.74	0.44	0.72	0.42
AoF	0.25	–	0.53	0.05	0.51	0.02	0.51	0.04	0.51	0.04	0.55	0.10
	0.50	–	0.52	0.05	0.51	0.03	0.51	0.04	0.52	0.03	0.53	0.11
	0.75	–	0.53	0.06	0.51	0.03	0.51	0.04	0.51	0.04	0.55	0.12
	–	L2	0.52	0.06	0.51	0.02	0.51	0.03	0.51	0.04	0.55	0.09
	0.25	L2	0.52	0.07	0.51	0.02	0.51	0.03	0.51	0.05	0.54	0.10
	0.5	L2	0.52	0.07	0.51	0.02	0.51	0.03	0.51	0.04	0.55	0.10
	0.75	L2	0.52	0.05	0.51	0.02	0.51	0.03	0.51	0.04	0.55	0.10
KD ($\alpha = 0.5, T = 10$)	–	–	0.50	0.06	0.51	0.02	0.51	0.03	0.50	0.07	0.57	0.13
RelaxLoss ($\alpha = 0.4$)	–	–	0.64	0.27	0.77	0.45	0.76	0.45	0.78	0.48	0.70	0.38
Adv-Reg ($\lambda = 3$)	–	–	0.53	0.07	0.61	0.16	0.59	0.13	0.62	0.18	0.59	0.17
MemberShield (Our defense)	–	–	0.53	0.06	0.61	0.19	0.56	0.12	0.61	0.21	0.58	0.20

still retain DP guarantees Dwork et al., 2019), *weak* values commonly used in the ML literature (*i.e.*, ϵ between 2 and 8 Blanco-Justicia et al., 2022), and *unsafe* values that would provide no real privacy guarantees (*i.e.*, $\epsilon \geq 10$). The δ parameter was kept constant at 10^{-6}, which is safe enough for all datasets (*i.e.*, δ should be well below $1/n$, where n denotes the number of training samples a client holds), and at least one order of magnitude smaller than the values used in the federated learning literature (Blanco-Justicia et al., 2022).

At first glance, it is clear that DP substantially reduces the attacker's advantage compared to the baseline results for all ϵ values. The reduction is in line with that achieved by MemberShield. However, unlike MemberShield, DP hampers the utility of the model very significantly,

Table 6
For vanilla FL, six baseline defenses, and MemberShield on the *ChMnistCustom* task: the global adversary's auR and maximum advantage against each client local model; the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
Vanilla FL (No defense)	–	–	0.77	0.46	0.78	0.48	0.79	0.51	0.77	0.48	0.79	0.50	0.87	0.73	402.49
Early stopping	–	–	0.61	0.34	0.66	0.28	0.67	0.30	0.64	0.22	0.67	0.31	0.76	0.66	384.01
AoF	0.25	–	0.86	0.61	0.83	0.53	0.85	0.56	0.84	0.53	0.85	0.55	0.83	0.64	399.65
	0.50	–	0.76	0.45	0.74	0.37	0.79	0.44	0.75	0.41	0.82	0.50	0.73	0.60	410.01
	0.75	–	0.58	0.24	0.53	0.09	0.51	0.14	0.51	0.05	0.57	0.14	0.13	0.13	403.76
	–	L2	0.80	0.54	0.76	0.43	0.74	0.43	0.73	0.41	0.73	0.46	0.61	0.56	396.17
	0.25	L2	0.85	0.54	0.75	0.39	0.76	0.45	0.76	0.43	0.78	0.46	0.47	0.44	398.23
	0.50	L2	0.63	0.31	0.56	0.15	0.60	0.23	0.58	0.15	0.62	0.21	0.43	0.43	407.99
	0.75	L2	0.58	0.25	0.53	0.09	0.52	0.12	0.52	0.09	0.57	0.14	0.13	0.13	402.26
DP	0.1	–	0.57	0.27	0.56	0.12	0.56	0.16	0.55	0.14	0.56	0.15	0.14	0.14	776.41
	0.5	–	0.59	0.23	0.55	0.14	0.56	0.13	0.54	0.11	0.53	0.17	0.10	0.12	769.97
	1	–	0.55	0.24	0.55	0.13	0.53	0.16	0.55	0.14	0.54	0.20	0.16	0.15	758.43
	2	–	0.58	0.24	0.56	0.14	0.55	0.12	0.56	0.11	0.56	0.14	0.18	0.18	762.24
	4	–	0.53	0.24	0.57	0.13	0.54	0.15	0.55	0.12	0.56	0.17	0.23	0.23	781.89
	8	–	0.56	0.25	0.57	0.16	0.53	0.12	0.54	0.13	0.55	0.19	0.25	0.25	789.08
	16	–	0.57	0.23	0.57	0.16	0.56	0.13	0.55	0.12	0.57	0.17	0.37	0.35	793.20
	100	–	0.57	0.20	0.55	0.13	0.56	0.12	0.53	0.14	0.56	0.22	0.48	0.46	797.58
	1000	–	0.56	0.18	0.57	0.15	0.61	0.23	0.55	0.12	0.58	0.19	0.51	0.48	797.95
KD ($\alpha = 0.5, T = 10$)	–	–	0.65	0.24	0.58	0.14	0.66	0.27	0.57	0.15	0.57	0.17	0.13	0.13	1075.54
RelaxLoss ($\alpha = 0.4$)	–	–	0.69	0.38	0.71	0.36	0.75	0.38	0.73	0.34	0.71	0.37	0.76	0.64	395.83
Adv-Reg ($\lambda = 3$)	–	–	0.71	0.32	0.64	0.21	0.73	0.43	0.65	0.29	0.71	0.39	0.87	0.72	615.50
MemberShield (Our defense)	–	–	0.59	0.20	0.58	0.16	0.58	0.16	0.59	0.16	0.58	0.21	0.73	0.67	315.95

Table 7
For vanilla FL, six baseline defenses, and MemberShield on the *ChMnistCustom* task: local adversaries' auR and maximum advantage (Adv) against the global model.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5	
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
Vanilla FL (No defense)	–	–	0.68	0.38	0.72	0.35	0.79	0.48	0.75	0.39	0.75	0.43
Early stopping	–	–	0.56	0.27	0.59	0.18	0.65	0.28	0.61	0.20	0.61	0.23
AoF	0.25	–	0.76	0.46	0.76	0.41	0.84	0.56	0.81	0.49	0.84	0.59
	0.50	–	0.61	0.27	0.63	0.20	0.75	0.39	0.70	0.31	0.75	0.45
	0.75	–	0.55	0.21	0.54	0.12	0.52	0.10	0.51	0.08	0.55	0.11
	–	L2	0.57	0.18	0.56	0.12	0.58	0.16	0.55	0.12	0.56	0.17
	0.25	L2	0.59	0.30	0.55	0.15	0.60	0.14	0.58	0.13	0.59	0.19
	0.50	L2	0.57	0.27	0.56	0.15	0.56	0.13	0.54	0.12	0.58	0.17
	0.75	L2	0.53	0.20	0.52	0.11	0.51	0.12	0.52	0.09	0.55	0.10
DP	0.1	–	0.53	0.21	0.55	0.11	0.56	0.17	0.54	0.13	0.57	0.16
	0.5	–	0.57	0.28	0.56	0.13	0.53	0.14	0.54	0.12	0.56	0.17
	1	–	0.59	0.21	0.55	0.11	0.55	0.15	0.53	0.16	0.56	0.17
	2	–	0.58	0.31	0.55	0.13	0.53	0.23	0.54	0.13	0.55	0.16
	4	–	0.57	0.17	0.57	0.15	0.55	0.12	0.54	0.13	0.54	0.21
	8	–	0.57	0.25	0.58	0.13	0.58	0.15	0.54	0.14	0.59	0.19
	16	–	0.63	0.30	0.55	0.13	0.58	0.17	0.57	0.14	0.55	0.12
	100	–	0.59	0.25	0.57	0.15	0.57	0.15	0.55	0.10	0.54	0.14
	1000	–	0.56	0.21	0.55	0.15	0.58	0.20	0.55	0.11	0.55	0.15
KD ($\alpha = 0.5, T = 10$)	–	–	0.56	0.24	0.54	0.11	0.52	0.12	0.50	0.07	0.55	0.12
RelaxLoss ($\alpha = 0.4$)	–	–	0.68	0.30	0.66	0.26	0.72	0.36	0.69	0.29	0.72	0.36
Adv-Reg ($\lambda = 3$)	–	–	0.64	0.23	0.64	0.26	0.67	0.32	0.68	0.30	0.69	0.43
MemberShield (Our defense)	–	–	0.55	0.23	0.55	0.14	0.58	0.15	0.57	0.12	0.57	0.14

reaching accuracies near the random guess even for the weakest privacy parameters (that is, larger ϵ values). The situation is even worse for a complex model such as VGG19, for which the training loss gets so large that the program encounters runtime errors. This indicates that, even though the mildest distortion introduced by DP is sufficient to achieve practical defense against MIAs, its randomness severely harms the learning process. In contrast, MemberShield introduces no distortion in the learning process, but regulates it to prevent overfitting, which not only does not significantly harm the accuracy of the model, but sometimes even improves it. To make things worse for DP, the training time with this defense is significantly high. This comes from clipping the model gradient for each sample in the training set before adding noise, which hinders the parallel processing of batches on the GPU. Renouncing parallel batch processing explains the 1.5 to 2 times higher training duration with respect to vanilla FL.

In contrast, MemberShield only incurs a slight training overhead due to the one-time label preprocessing it introduces to generate soft labels, which is more than compensated by the time saved by the early training stopping. As a result, the total training time with our defense is less than that of vanilla FL training.

7.3. Comparison with KD

For KD, we implemented distillation for membership privacy (DMP) (Shejwalkar & Houmansadr, 2021) with a self-distillation approach similar to that of Chen et al. (2022), Tang et al. (2022), and clients with no external public data. The unavailability of external data is realistic in many domains (such as healthcare), for which the training data are private/proprietary. Instead, we exploited the same private dataset to train the student (protected) model, but with the soft label generated from the teacher's prediction that is finally sent to the server. Following

Table 8
For vanilla FL, six baseline defenses, and MemberShield on the *Purchase100Custom* task: the global adversary's auR and maximum advantage (Adv) against each client local model; the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
Vanilla FL (No defense)	–	–	0.62	0.27	0.62	0.26	0.64	0.31	0.63	0.27	0.65	0.32	0.96	0.91	1903.93
Early stopping	–	–	0.62	0.22	0.59	0.21	0.60	0.21	0.60	0.23	0.65	0.28	0.98	0.93	567.90
AoF	0.25	–	0.71	0.36	0.60	0.20	0.63	0.24	0.61	0.21	0.65	0.28	0.85	0.80	2013.05
	0.50	–	0.65	0.24	0.60	0.18	0.63	0.24	0.61	0.20	0.65	0.27	0.78	0.75	1998.05
	0.75	–	0.59	0.16	0.56	0.09	0.59	0.14	0.57	0.11	0.59	0.15	0.68	0.66	2027.18
	–	L2	0.72	0.41	0.65	0.29	0.67	0.35	0.66	0.31	0.69	0.38	0.86	0.82	2107.30
	0.25	L2	0.69	0.36	0.65	0.26	0.67	0.33	0.64	0.26	0.70	0.36	0.85	0.79	1925.89
	0.50	L2	0.54	0.08	0.55	0.09	0.59	0.16	0.57	0.13	0.59	0.16	0.83	0.78	2207.19
	0.75	L2	0.63	0.20	0.60	0.17	0.64	0.22	0.61	0.18	0.64	0.23	0.78	0.74	2125.60
DP	0.10	–	0.51	0.03	0.51	0.02	0.51	0.03	0.51	0.02	0.51	0.03	0.01	0.01	5475.10
	0.50	–	0.51	0.04	0.51	0.02	0.51	0.03	0.51	0.02	0.51	0.03	0.01	0.01	5411.33
	1	–	0.52	0.04	0.51	0.02	0.51	0.03	0.51	0.02	0.51	0.03	0.01	0.01	5302.76
	2	–	0.52	0.04	0.51	0.03	0.51	0.03	0.51	0.02	0.51	0.02	0.01	0.01	5411.04
	4	–	0.51	0.04	0.51	0.02	0.51	0.03	0.51	0.02	0.51	0.03	0.01	0.01	5277.44
	8	–	0.51	0.04	0.52	0.03	0.51	0.03	0.51	0.02	0.51	0.03	0.01	0.01	5451.01
	16	–	0.52	0.04	0.52	0.02	0.51	0.03	0.51	0.02	0.51	0.03	0.02	0.01	5328.21
	100	–	0.52	0.06	0.51	0.03	0.52	0.03	0.51	0.02	0.52	0.04	0.02	0.02	5607.28
	1000	–	0.54	0.08	0.52	0.03	0.52	0.04	0.51	0.03	0.54	0.07	0.02	0.02	5485.50
KD ($\alpha = 0.5, T = 3$)	–	–	0.64	0.29	0.62	0.25	0.64	0.31	0.63	0.28	0.65	0.32	0.96	0.92	4288.17
RelaxLoss ($\alpha = 0.8$)	–	–	0.60	0.15	0.53	0.05	0.54	0.07	0.53	0.05	0.56	0.11	0.8	0.79	1871.70
Adv-Reg ($\lambda = 3$)	–	–	0.63	0.23	0.60	0.19	0.64	0.29	0.61	0.22	0.63	0.29	0.94	0.89	4126.56
MemberShield (Our defense)	–	–	0.55	0.09	0.55	0.09	0.56	0.12	0.55	0.10	0.57	0.12	0.89	0.87	1851.09

Table 9
For vanilla FL, six baseline defenses, and MemberShield on the *Purchase100Custom* task: local adversaries' auR and maximum advantage (Adv) against the global model.

Method	Parameters		Client-1		Client-2		Client-3		Client-4		Client-5	
	Dr/Ep	Reg	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
Vanilla FL (No defense)	–	–	0.55	0.12	0.58	0.17	0.62	0.27	0.62	0.23	0.61	0.27
Early stopping	–	–	0.54	0.08	0.58	0.16	0.57	0.14	0.58	0.19	0.57	0.18
AoF	0.25	–	0.53	0.07	0.54	0.08	0.57	0.13	0.56	0.11	0.58	0.14
	0.50	–	0.53	0.05	0.53	0.06	0.56	0.11	0.55	0.09	0.57	0.11
	0.75	–	0.52	0.05	0.52	0.04	0.54	0.07	0.53	0.05	0.55	0.08
	–	L2	0.54	0.09	0.55	0.09	0.59	0.16	0.57	0.13	0.59	0.16
	0.25	L2	0.54	0.09	0.55	0.09	0.58	0.15	0.57	0.13	0.59	0.16
	0.50	L2	0.68	0.31	0.64	0.25	0.67	0.31	0.65	0.26	0.69	0.34
	0.75	L2	0.54	0.07	0.54	0.07	0.57	0.13	0.56	0.10	0.57	0.13
DP	0.1	–	0.51	0.04	0.51	0.03	0.51	0.03	0.51	0.02	0.51	0.03
	0.5	–	0.51	0.04	0.51	0.02	0.51	0.03	0.51	0.02	0.51	0.03
	1	–	0.54	0.06	0.51	0.02	0.51	0.02	0.51	0.02	0.51	0.03
	2	–	0.52	0.04	0.51	0.02	0.51	0.02	0.51	0.02	0.51	0.03
	4	–	0.51	0.04	0.51	0.03	0.51	0.02	0.51	0.02	0.51	0.03
	8	–	0.52	0.05	0.51	0.02	0.51	0.02	0.51	0.02	0.51	0.03
	16	–	0.52	0.05	0.51	0.03	0.51	0.02	0.51	0.02	0.51	0.03
	100	–	0.52	0.04	0.51	0.03	0.52	0.03	0.51	0.02	0.52	0.04
	1000	–	0.51	0.04	0.51	0.03	0.52	0.04	0.52	0.03	0.52	0.04
KD ($\alpha = 0.5, T = 3$)	–	–	0.55	0.14	0.58	0.17	0.62	0.28	0.61	0.24	0.62	0.28
RelaxLoss ($\alpha = 0.8$)	–	–	0.52	0.04	0.52	0.04	0.53	0.06	0.53	0.05	0.54	0.07
Adv-Reg ($\lambda = 3$)	–	–	0.56	0.11	0.57	0.15	0.62	0.24	0.60	0.19	0.61	0.23
MemberShield (Our defense)	–	–	0.52	0.05	0.53	0.06	0.56	0.11	0.55	0.08	0.55	0.09

the previous study (Chen et al., 2022) and practical standards (Hinton et al., 2015), we set the hyperparameter $\alpha = 0.5$, which balances the distillation and cross-entropy losses. The temperature hyperparameter T, which controls the softness of the teacher's prediction, was set to 3 for the *Purchase100Custom* task, and to 10 for the rest of the tasks, as they show the best performance after experimenting with different values in the range $[1, \dots, 12]$.

Empirical results show that *KD preserves neither membership privacy nor model utility in FL settings*. Since the teacher networks are trained without defense, they show a large gap between the training and test losses, and they generate high-confidence soft labels, which are later transferred to the student models. Hence, the loss and entropy difference between member and non-member samples become almost identical in the student and teacher models (*i.e.*, the baseline without any defense). Consequently, KD fails to preserve membership privacy of the training samples.

On the other hand, the unprotected teacher model learned from non-iid data distributions across the clients produces diverse degrees of soft labels across the local data. Moreover, soft labels differ for samples of the same class in client data. This introduces uncontrolled randomness in the local models' learning, thereby leading to local models residing far from each other within the solution space. Consequently, these divergent local models fail to deliver a meaningful global model upon aggregation, causing overall utility loss of the final learned model. Table 10 illustrates this scenario within the 7th training round for the *ChMnistCustom* task. We can observe that the local models overfit to different degrees on their soft labels, yielding membership information disclosure and useless global models upon aggregation.

In contrast, MemberShield employs uniform soft labels (precisely designed at the beginning of training to imitate the output prediction of

Table 10
Local and global models' characteristics on the *ChMnistCustom* task during the 7th training round with KD.

Local models							Global model					
Client	Tr loss	Te loss	Tr Ac	Te Ac	auR	Adv	Tr loss	Te loss	Tr Ac	Te Ac	auR	Adv
1	0.57	1.25	0.87	0.52	0.72	0.42	3.23	3.02	0.10	0.13	0.52	0.13
2	0.22	1.07	0.95	0.60	0.70	0.41	3.07	3.20	0.12	0.14	0.51	0.08
3	0.17	1.75	0.97	0.56	0.77	0.46	3.10	3.10	0.12	0.12	0.50	0.08
4	6.07	5.94	0.12	0.12	0.50	0.07	3.08	3.02	0.12	0.12	0.50	0.07
5	0.23	1.14	0.98	0.60	0.77	0.49	3.20	2.97	0.12	0.12	0.54	0.11

non-member samples) for training samples across the clients, and further limits overtraining through early stopping regularization. Therefore, it effectively preserves membership privacy without damaging model utility.

7.4. Comparison with Adv-Reg

Avg-Reg was implemented by following the settings in its original proposal (Nasr et al., 2018). The inference attack model incorporated in the target model training was designed with three separate, fully connected sub-networks for all the tasks. One network of layer sizes $[100, 1024, 512, 64]$ operates on the target model's prediction vector, while another network of layer sizes $[100, 512, 64]$ operates on one-hot coded labels. The third (shared) network operates on the concatenation of the output of the first two networks and has layer sizes $[256, 64, 1]$. The ReLU activation was incorporated in each layer except the classification layer, which uses sigmoid activation to express the membership probability of the target sample. The inference attack model is trained with a subset of the training set and a hold-out validation set of equal sample size (Jia et al., 2019). The adversarial regularization factor λ was set to 3 for all tasks.

Like KD, Avg-Reg fails to provide adequate privacy protection or substantive utility. Moreover, it introduces considerable training overhead due to the need to train the additional attack model at each local epoch. Its inadequacy for privacy protection stems from its reliance on neural network-based attack models as regularization terms while training the protected local models, thereby disregarding more resilient metrics-based attacks. We observe from the empirical results that adversaries gain the highest advantage when they predominantly exploit confidence thresholds. In contrast, the adversary obtains much less advantage when MemberShield is employed.

In terms of utility, the performance of Avg-Reg depends on the task. On the one hand, it reaches the same accuracy as vanilla FL for simple tasks such as *Purchase100Custom* and *ChMnistCustom*. However, its performance diminishes when dealing with more complex tasks like *Cifar10Custom* and *Cifar10Vgg19*, resulting in accuracies as low as 17% and 25%, which are meaningless. Meaningless utility can be explained by the adversarial regularization term, which detrimentally impacts on the local model utility. Besides, the amount of regularization depends on the neural network-based attack models incorporated in the local training at the clients' premises. These attack models are diverse across clients due to diverse sets of parameters, hyperparameters, and training datasets. As a result, the clients produce quite different local models. This scenario becomes exacerbated by the non-iid data distributions among clients. Consequently, the global model derived from aggregating these local models lacks insights gleaned from local data, yielding meaningless utility.

This scenario is illustrated in Table 11, where the training and test accuracies of local and global models are reported during the eighth round of FL on the CIFAR10 dataset with a custom model. We can see that the local models fit the local datasets to different degrees. However, when local models are combined into the global model, the knowledge derived from the local datasets gets lost, which explains the poor performance of Adv-Reg on complex tasks.

7.5. Comparison with RelaxLoss

RelaxLoss used the implementation provided by the authors of Jia et al. (2019) for FL settings, employing the same loss threshold values for enforcing gradient ascent or posterior flattening steps as in the original paper, which are 0.4, 0.4, and 0.8 for CIFAR10, CH-MINST, and Purchase100 datasets, respectively.

As with the previous methods, RelaxLoss fails to defend membership privacy while diminishing the model's utility. This is clear from the almost equal global adversary's MIA advantages against local models when trained with RelaxLoss versus trained without any defense across all the classification models. In fact, not even the global model, where local models are blended and client samples can be expected to remain undetected due to being hidden in the crowd, succeeds in providing client privacy. All other baseline defenses meet this expectation, thereby offering much more protection against local adversaries than global adversaries across all the tasks. On this ground, RelaxLoss performs the worst in protecting privacy among all the compared methods.

The first reason behind such high privacy breaches are RelaxLoss's main two steps for protecting privacy: *enforcing gradient ascent* and *posterior flattening* triggered when training loss falls below a predefined threshold, which happens during the last few rounds of FL. The early round of federated training is vanilla FL, where the adversary can easily differentiate between the member and non-member samples. Secondly, it is designed with the strong assumption that an optimal attack depends only on the loss value underestimating entropy and confidence thresholding attacks, which are more powerful than a loss thresholding attack (Song & Mittal, 2021). During the posterior flattening step, the method retains the prediction confidence for the ground truth class while redistributing the remaining probability equally among all non-ground-truth classes. Consequently, it suppresses only the loss difference in member and non-member samples without impeding the model's tendency to be excessively confident in target classes. This results in highly confident predictions for member instances and less confidence for non-member instances. The adversaries can exploit this distinction to execute MIA with a high success rate. Consequently, even though this defense performs well in centralized machine learning when a model is trained for many epochs, it fails to maintain such performance in the FL setting.

On top of that, the gradient ascent step prevents model convergence, bringing some randomness to the learning process by diverting the gradient direction, which eventually prevents the model from converging in global minima. This randomness adds to the heterogeneous nature of clients' data and worsens the scenario. As a result, the model loses its utility by a significant magnitude.

7.6. Comparison with early stopping

To implement the early stopping baseline, we terminated local training if validation loss did not improve for consecutive 3 epochs for *Purchase100Custom* and 10 epochs for the rest of the tasks. We then selected the model with the best validation accuracy for evaluation.

Table 11
Local and global models' characteristics on *Cifar10Custom* task with the 8th training round with Avd-Reg.

Local models							Global model					
Client	Tr loss	Te loss	Tr Ac	Te Ac	auR	Adv	Tr loss	Te loss	Tr Ac	Te Ac	auR	Adv
1	5.32	5.95	0.67	0.38	0.70	0.33	7.41	7.92	0.12	0.13	0.50	0.04
2	8.01	8.04	0.10	0.10	0.50	0.01	7.53	7.64	0.12	0.11	0.51	0.02
3	8.15	8.04	0.34	0.29	0.56	0.09	7.54	7.54	0.12	0.12	0.51	0.02
4	7.47	7.48	0.37	0.30	0.57	0.12	7.51	8.00	0.13	0.11	0.51	0.03
5	5.26	5.44	0.70	0.37	0.71	0.34	7.48	7.31	0.15	0.12	0.52	0.06

Table 12
Performance of an adaptive membership inference attack initiated by the *global adversary* against the *local models* trained on different tasks.

Task	Client-1		Client-2		Client-3		Client-4		Client-5	
	auR	Adv	auR	Adv	auR	Adv	auR	Adv	auR	Adv
Cifar10Custom	0.57	0.14	0.71	0.34	0.61	0.21	0.59	0.18	0.61	0.27
Cifar10Vgg19	0.60	0.20	0.63	0.21	0.58	0.13	0.60	0.16	0.59	0.16
ChMnistCustom	0.57	0.27	0.58	0.15	0.60	0.21	0.59	0.16	0.57	0.17
Purchase100Custom	0.57	0.12	0.61	0.16	0.58	0.13	0.58	0.14	0.56	0.12

Early stopping fell short of safeguarding privacy for simple datasets and models, exhibiting much lower privacy protection than MemberShield. On the other hand, even though it offered the same level of privacy protection as MemberShield when dealing with complex datasets and models, it failed to achieve the same test accuracy as our defense.

This indicates that early stopping does not allow the model to explore the entire solution space, potentially missing out essential features or patterns that could contribute to better accuracy. Specifically, one-hot encoded hard labels create a non-smooth loss surface due to discrete label transitions, leading to sharp changes in the loss function. The non-smoothness of the loss surface poses challenges for the gradient descent algorithm to navigate abrupt changes in the loss surface (Szegedy et al., 2016), making early stopping more likely to get stuck in suboptimal regions. In contrast, MemberShield employs soft-encoded labels that yield a smooth loss surface, which facilitates more stable optimization. It also removes local minima in the solution space, making it easier for gradient-based optimization algorithms to converge to an optimal solution during training (Szegedy et al., 2016). This smoothness also prevents the model from overfitting to noisy or irrelevant details in the training data (Hinton et al., 2015).

7.7. Comparison with anti-overfitting techniques

We implemented standard anti-overfitting (AoF) baselines with different dropout rates $(0.25, 0.50, 0.75)$, L2 regularization, and a combination of both. Following previous defense baselines (Chen et al., 2022; Tang et al., 2022), we enforced them in the last two and three fully connected layers for the *Cifar10Vgg19* and *ChMnistCustom* tasks, respectively. For the other two tasks (*Cifar10Custom*, *Purchase100Custom*), we incorporated them in the last fully connected layer.

Dropout offers a very poor privacy-utility trade-off: higher dropout generally ensures better privacy protection but substantially hurts the model utility and vice versa. In contrast, L2 regularization, when enforced alone, does not safeguard membership privacy but incurs utility loss. The combination of a high dropout rate and L2 regularization delivers acceptable privacy protection for simple tasks like *Purschase100Custom*, but this comes at the expense of model utility. The final global model experienced the lowest utility drop from 91% to 74% for this task. However, the combination offered no privacy protection for complex tasks but caused utility loss. Note that any form of regularization destroys the utility of well-established and highly optimized models like VGG19.

The main reason behind the accuracy drop is that both dropout and L2 change the learning process, parameters and internal structure, and both of them restrict the model learning capacity. L2 alters the learning rule to multiplicatively shrink the weight vector by a constant factor on each step just before performing the usual gradient update (Goodfellow et al., 2016). Dropout reduces the capacity of the model by effectively removing some units from the network by multiplying their output value by zero. Therefore, a much deeper model, a larger dataset, and many more iterations of the training algorithm are needed to offset this effect and make the regularization effective (Goodfellow et al., 2016). These requirements are impractical in FL as the clients' datasets are comparatively smaller and more heterogeneous than in centralized settings, and few iterations of the training algorithm are executed at each communication round (Goodfellow et al., 2016). As a result, no usable accuracy is retained.

On the other hand, the poor membership privacy offered by these methods can be understood due to their lack of focus on MIA protection. They prevent overfitting by reducing the generalization gap quantified by the difference between training and test loss. But they cannot suppress the distinction in other properties (confidence, entropy) of the target model for training and test samples. Properly implementing regularization in a very deep model also requires trial and error in choosing the best layers and accurate regularization amount, which is often a combinatorial optimization problem. This trial-and-error scenario worsens in FL because each local model, while enforcing regularization, must also consider the impact of other models' regularizations on the global model to achieve optimal performance, thereby resulting in improper implementation and membership privacy breaches.

7.8. Privacy protection against an adaptive attack

Going one step further, in this section, we evaluate MemberShield's effectiveness against an adaptive inference attack (Song & Mittal, 2021) where the adversary knows the defense method and trains the attack model (for a neural network-based attack) or fine-tunes threshold functions (for a metrics-based attack) by using the same soft labels and early stop regularization employed by the defense in each task.

In this experiment, we considered a global adversary who performs MIAs against individual local models after each federated training round only because, as shown in the previous section, attacks against global models showed low performance. Table 12 reports the attack performance against the five clients on four tasks. In all of them, MemberShield provides the same (or even better) membership privacy protection as against non-adaptive attacks.

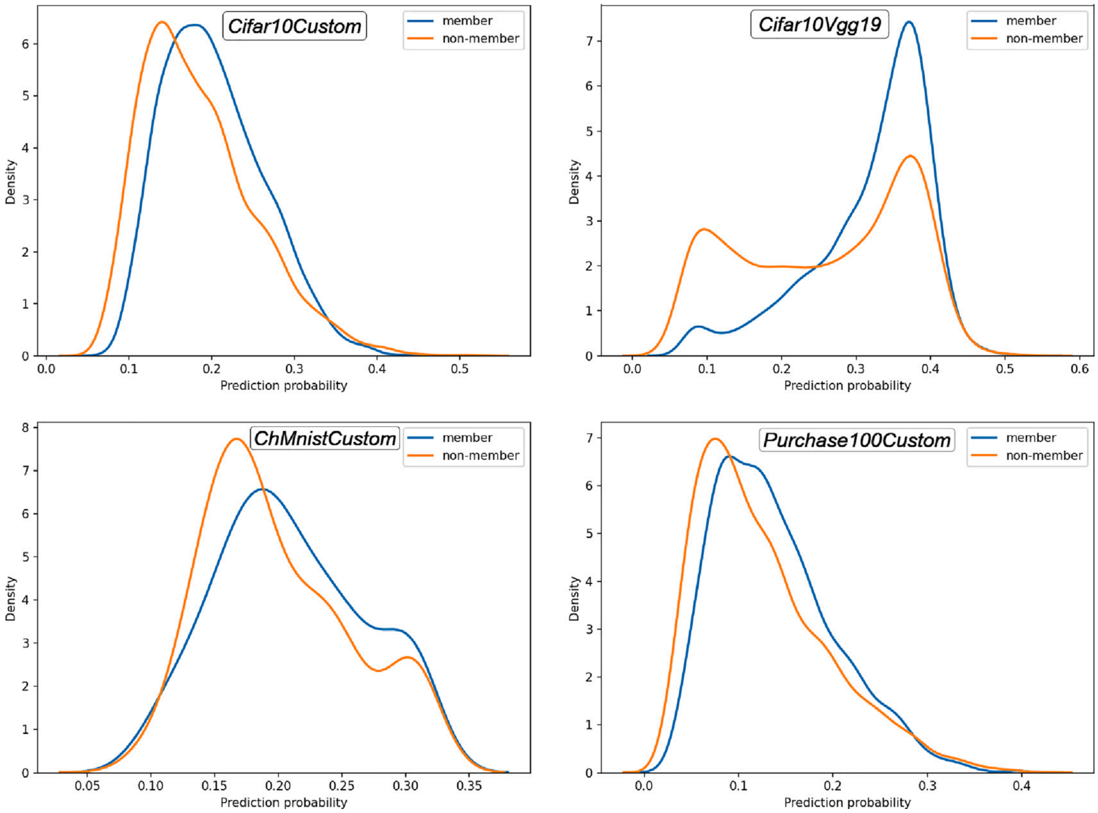


Fig. 4. Prediction probability distribution of member and non-member samples for the *Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom*, and *Purchase100Custom* tasks using MemberShield.

7.9. Statistical indistinguishability due to MemberShield

In this section, we discuss the statistical indistinguishability that MemberShield brings to different features of the target model on the member and non-member samples for effectively mitigating MIAs.

As discussed in Section 4, the most outstanding features to execute MIAs are *prediction confidence*, *modified entropy*, and *gradient norm*. In the following, we study these three features. Specifically, we examine the local model with the highest advantage for the global adversary among all models generated by the five clients during ten rounds of federated training. We then visualize the distribution of prediction confidence, modified entropy, and gradient norm for this model on both member and non-member samples.

The overlapping confidence distributions of member and non-member samples shown in Fig. 4 indicate that MemberShield effectively regulates prediction confidence for member samples during the model training to a level achievable for non-member samples. This regulation stems from the first step of MemberShield. Its impact on the modified entropy is the same as on the prediction confidence, as shown in Fig. 5. In this way, MemberShield ensures membership privacy protection against the strongest MIA exploiting the target model's output.

Gradient norm distributions are shown in Fig. 6. We can see that the gradient distribution for member samples is not heavily skewed towards zero, unlike for the unprotected vanilla FL shown in Fig. 3, indicating that MemberShield substantially minimizes unintended memorization of samples by enforcing the early stopping step during training. The distributions for member and non-member samples are evenly distributed across a wide range of gradient norm values, which makes them indistinguishable. Therefore, an adversary cannot exploit the gradient norm to perform MIAs.

We also investigated how MemberShield impacts the generalization error for defending MIAs. Fig. 7 depicts the empirical cumulative distribution function (CDF) representing the generalization error across tasks, including *Cifar10Custom*, *Cifar10VGG19*, *ChMnistCustom*, and *Purchase100Custom* using MemberShield against those without any defensive measure (vanilla FL). The y-axis represents the cumulative fraction of classes for which the target model's generalization error is below the respective value on the x-axis. A line closer to $x = 0$ indicates a lower generalization error. The curves suggest that MemberShield notably reduces generalization errors across all tasks, effectively improving membership privacy.

8. Conclusions and future work

We have proposed a privacy-preserving FL framework, *MemberShield*, which safeguards MIAs by generalizing the local models. It trains the local model by exploiting soft-encoded ground truth labels of samples instead of standard one-hot encoded labels to regulate over-confident prediction. It also incorporates an enhanced early stopping step in local training to prevent unintended memorization. Extensive performance analysis shows that *MemberShield offers much better practical membership privacy than previous defenses except DP, and better or similar global model accuracy than all previous defenses (similar to vanilla FL) and much better than DP, while reducing the computational cost of model training with respect to all previous defenses.*

As future work, we plan to examine the robustness of *MemberShield* against active MIAs, where adversaries manipulate the model's training process to retrieve additional information. We also plan to investigate its performance when incorporating defenses against security attacks, such as those proposed in Jebreel and Domingo-Ferrer (2023), Jebreel et al. (2022) and Jebreel, Domingo-Ferrer, Sánchez, and Blanco-Justicia (2024).

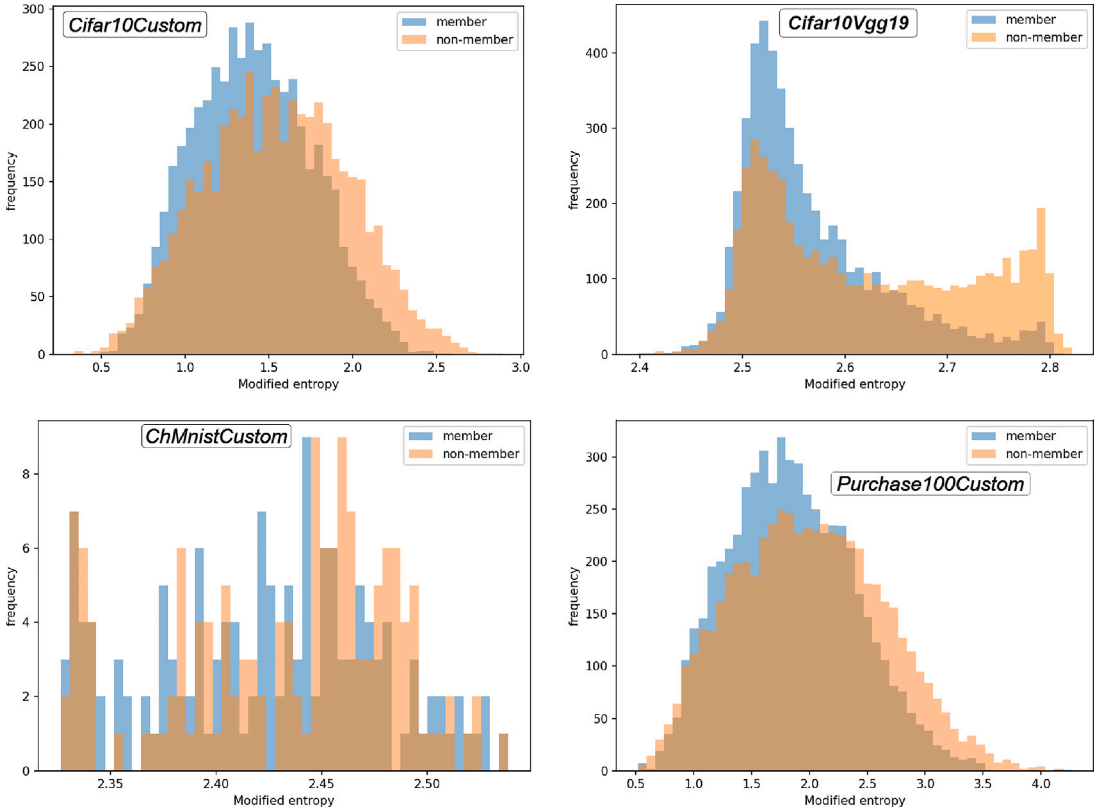


Fig. 5. Modified entropy distribution of member and non-member samples for the *Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom*, and *Purchase100Custom* tasks using MemberShield.

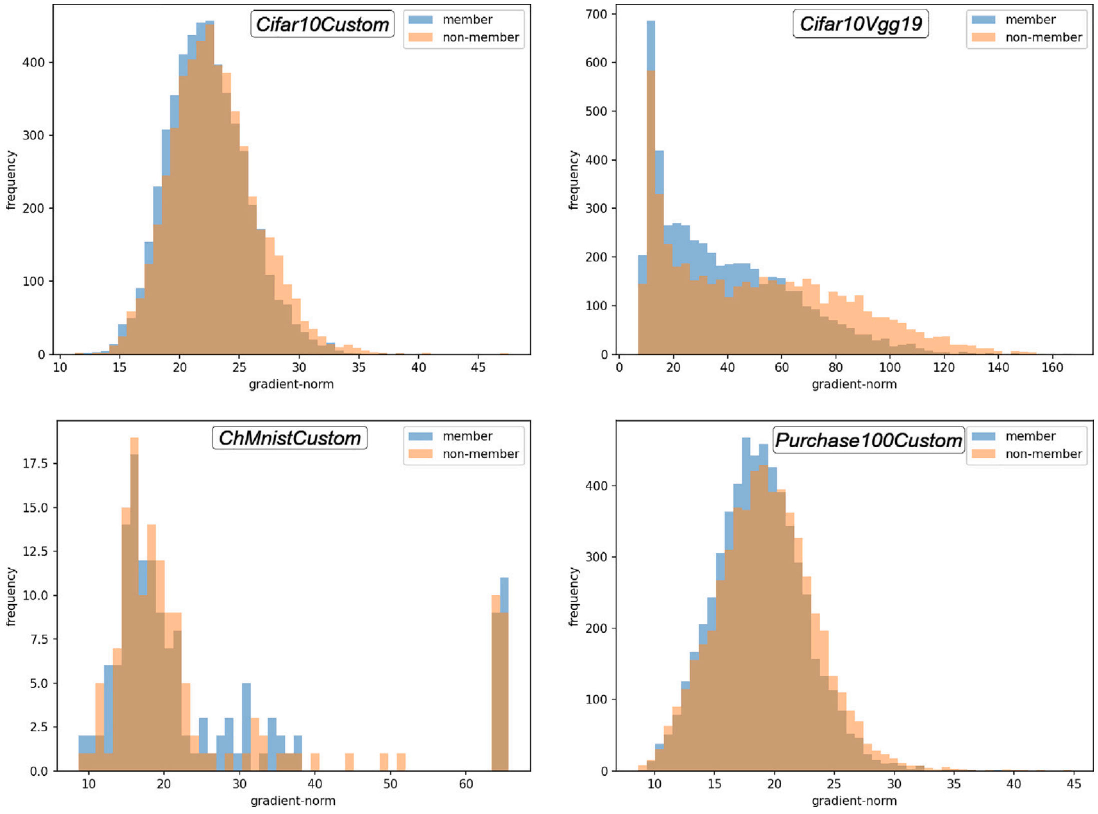


Fig. 6. Gradient norm distribution of member and non-member samples for the *Cifar10Custom*, *Cifar10Vgg19*, *ChMnistCustom*, and *Purchase100Custom* tasks using MemberShield.

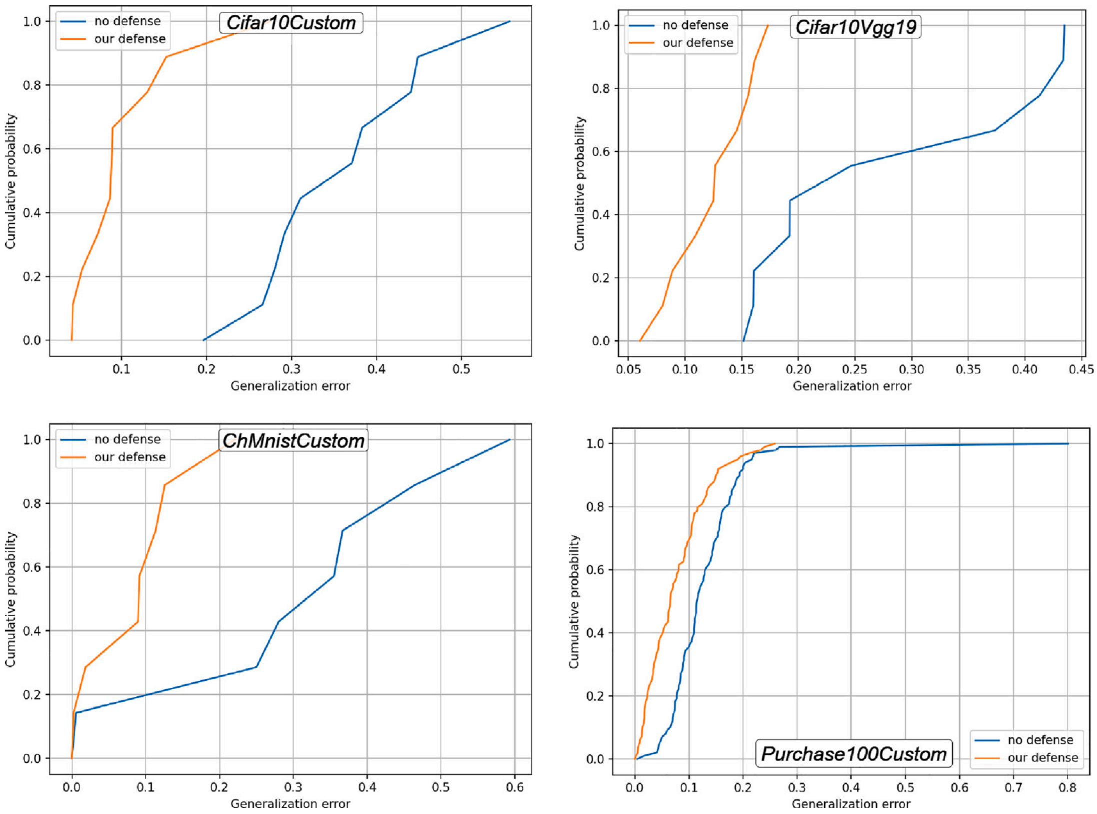


Fig. 7. Empirical CDF of the generalization error for the *Cifar10Custom*, *Cifar10VGG19*, *ChMnistCustom*, and *Purchase100Custom* tasks using our defense (MemberShield) and without defense (Vanilla FL).

CRediT authorship contribution statement

Faisal Ahmed: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **David Sánchez:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Zouhair Haddi:** Funding acquisition. **Josep Domingo-Ferrer:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Josep Domingo-Ferrer and Zouhair Haddi reports financial support was provided by European Commission. Josep Domingo-Ferrer and David Sanchez reports financial support was provided by Ministry of Science Technology and Innovations. Josep Domingo-Ferrer and David Sanchez reports financial support was provided by Cybersecurity National Institute. Josep Domingo-Ferrer and David Sanchez reports financial support was provided by Government of Catalonia Agency for Administration of University and Research Grants. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://github.com/faisalahm3d/MemberShield.

Acknowledgments

This research was partially supported by the European Commission (projects HORIZON-101073222 "BosomShield" and H2020-871042 "SoBigData++"), MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe" under grant PID2021-123637NB-I00 "CURLING", the European Union's NextGenerationEU/PRTR via INCIB (project "HERMES" and INCIBE-URV Cybersecurity Chair), and the Government of Catalonia, Spain (ICREA Acadèmia Prizes to J. Domingo-Ferrer and to D. Sánchez, and grant 2021 SGR 00115).

Appendix. Robustness of MemberShield on heterogeneous data

We also evaluated MemberShield's performance under various degrees of local data heterogeneity among clients, focusing specifically on the heterogeneity caused by label distribution skewness. To simulate this local data heterogeneity, we partitioned the datasets using the Dirichlet distribution with concentration parameter $\beta \in \{0.5, 0.1\}$. The other hyperparameters were kept constant during these experiments. The experimental results are presented in Tables A.1 through A.8.

As label distribution skewness increases (indicated by a smaller β value), model performance drops, and the membership privacy breaches from both local and global models rise significantly across all tasks, as shown in Tables A.1 through A.8 for vanilla FL. This accuracy drop stems from unevenly distributed data samples from different classes across the clients, which forces local models to learn different representations. These representations are biased towards the classes with the majority of samples. As a result, aggregating these local models leads to inconsistent updates of the global model. Additionally, the local models tend to overfit on samples from the majority class, resulting in different behaviors for member and non-member samples. This overfitting contributes to higher membership privacy breaches for the training samples.

In contrast, MemberShield consistently protects against MIAs while maintaining nearly the same global model accuracy as vanilla FL and, in some cases, performing even better. Additionally, MemberShield consistently achieves faster training times than vanilla FL. Although

Table A.1
Vanilla FL and MemberShield performance on the *Cifar10Custom* task: the global adversary's auR and maximum advantage (Adv) against each client local model, the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac↑	Te Ac↑	Time ↓
0.5	Vanilla FL	0.90	0.74	0.96	0.82	0.97	0.84	0.93	0.78	0.88	0.59	0.73	0.59	2128.94
	MemberShield	0.57	0.13	0.58	0.14	0.59	0.15	0.56	0.11	0.55	0.12	0.67	0.62	1893.04
0.1	Vanilla FL	0.96	0.81	0.91	0.75	0.76	0.42	0.96	0.82	0.83	0.52	0.61	0.48	2014.56
	MemberShield	0.58	0.14	0.59	0.15	0.57	0.15	0.50	0.18	0.60	0.17	0.56	0.53	1884.83

Table A.2
Vanilla FL and MemberShield performance on the *Cifar10Custom* task: local adversaries' auR and maximum advantage (Adv) against the global model. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5	
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
0.5	Vanilla FL	0.78	0.47	0.82	0.50	0.89	0.65	0.77	0.42	0.73	0.36
	MemberShield	0.53	0.07	0.56	0.10	0.55	0.09	0.55	0.10	0.55	0.11
0.1	Vanilla FL	0.85	0.52	0.87	0.67	0.72	0.34	0.94	0.77	0.79	0.49
	MemberShield	0.55	0.10	0.56	0.11	0.56	0.15	0.54	0.21	0.58	0.13

Table A.3
Vanilla FL and MemberShield performance on the *Cifar10Vgg19* task: the global adversay's auR and maximum advantage (Adv) against each client local model, the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac↑	Time ↓
0.5	Vanilla FL	0.83	0.56	0.93	0.74	0.91	0.79	0.79	0.41	0.77	0.42	0.76	0.61	4187.30
	MemberShield	0.57	0.11	0.59	0.16	0.57	0.14	0.55	0.25	0.64	0.23	0.65	0.63	3201.01
0.1	Vanilla FL	0.97	0.81	0.92	0.80	0.77	0.44	0.97	0.84	0.88	0.61	0.49	0.36	4098.33
	MemberShield	0.58	0.15	0.64	0.23	0.63	0.24	0.53	0.25	0.64	0.23	0.37	0.35	3196.22

Table A.4
Vanilla FL and MemberShield performance on the *Cifar10Vgg19* task: local adversaries' auR and maximum advantage (Adv) against the global model. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5	
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
0.5	Vanilla FL	0.81	0.53	0.93	0.72	0.92	0.79	0.79	0.41	0.78	0.43
	MemberShield	0.55	0.09	0.53	0.07	0.56	0.14	0.54	0.18	0.56	0.12
0.1	Vanilla FL	0.95	0.80	0.91	0.80	0.74	0.37	0.97	0.84	0.87	0.58
	MemberShield	0.57	0.13	0.62	0.21	0.59	0.15	0.53	0.18	0.61	0.20

Table A.5
Vanilla FL and MemberShield performance on the *ChMnistCustom* task: the global adversary's auR and maximum advantage (Adv) against each client local model, the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
0.5	Vanilla FL	0.90	0.62	0.87	0.63	0.80	0.50	0.87	0.60	0.82	0.49	0.67	0.52	411.02
	MemberShield	0.57	0.17	0.55	0.09	0.59	0.15	0.53	0.20	0.55	0.10	0.56	0.50	359.45
0.1	Vanilla FL	0.94	0.76	0.88	0.70	0.71	0.34	0.88	0.64	0.83	0.53	0.59	0.34	413.32
	MemberShield	0.59	0.26	0.56	0.10	0.62	0.19	0.62	0.23	0.57	0.13	0.31	0.31	348.93

Table A.6
Vanilla FL and MemberShield performance on the *ChMnistCustom* task: local adversaries' auR and maximum advantage (Adv) against the global model. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5	
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
0.5	Vanilla FL	0.89	0.66	0.88	0.63	0.70	0.36	0.85	0.60	0.84	0.50
	MemberShield	0.57	0.28	0.55	0.10	0.59	0.13	0.56	0.17	0.54	0.10
0.1	Vanilla FL	0.91	0.68	0.86	0.66	0.75	0.42	0.83	0.53	0.76	0.40
	MemberShield	0.59	0.28	0.55	0.11	0.59	0.14	0.59	0.20	0.55	0.11

Table A.7
Vanilla FL and MemberShield performance on the *Purchase100Custom* task: the global adversary's auR and maximum advantage (Adv) against each client local model, the global model's training accuracy (Tr Ac), test accuracy (Te Ac), and training time. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5		Global model		
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	Tr Ac ↑	Te Ac ↑	Time ↓
0.5	Vanilla FL	0.83	0.51	0.81	0.47	0.81	0.47	0.81	0.45	0.83	0.49	0.85	0.76	1897.09
	MemberShield	0.57	0.12	0.54	0.08	0.55	0.09	0.57	0.11	0.55	0.08	0.81	0.78	1232.11
0.1	Vanilla FL	0.96	0.79	0.93	0.71	0.94	0.73	0.96	0.79	0.97	0.84	0.72	0.64	1903.43
	MemberShield	0.56	0.11	0.60	0.16	0.58	0.13	0.56	0.11	0.57	0.14	0.64	0.61	1123.84

Table A.8
Vanilla FL and MemberShield performance on the *Purchase100Custom* task: local adversaries' auR and maximum advantage (Adv) against the global model. It can be seen that MemberShield offers the best performance in all columns.

Label skew (β)	Method	Client-1		Client-2		Client-3		Client-4		Client-5	
		auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓	auR ↓	Adv ↓
0.5	Vanilla FL	0.79	0.45	0.78	0.43	0.77	0.41	0.78	0.41	0.80	0.45
	MemberShield	0.54	0.07	0.54	0.07	0.54	0.07	0.54	0.09	0.54	0.07
0.1	Vanilla FL	0.93	0.70	0.90	0.64	0.90	0.63	0.92	0.70	0.94	0.72
	MemberShield	0.56	0.11	0.60	0.16	0.58	0.13	0.56	0.11	0.57	0.14

there is an accuracy drop with increased label distribution skewness, this is justified because the primary design objective of MemberShield is to provide data privacy protection while preserving the same level of accuracy that vanilla FL offers.

MemberShield does not include mechanisms to address the data heterogeneity problem. However, it is flexible enough to incorporate existing methods, such as FedDF (Lin et al., 2020) and FCCL (Huang et al., 2022), to tackle data heterogeneity.

To incorporate FedDF, clients must update their local models on private data by using the local training techniques proposed in MemberShield and return the protected models to the server. The server then trains a student model with the same architecture as the clients' models, distills the ensemble knowledge of the local models by using an unlabeled public dataset, and eventually broadcasts the student model to clients for the next round of private training.

To integrate FCCL with MemberShield, clients should first pre-train local models on private data by using the soft labels recommended by MemberShield. During local training with MemberShield, clients must incorporate dual-domain distillation (intra-domain distillation loss and inter-domain distillation loss) along with cross-entropy loss calculated on the soft-encoded labels of private data. After receiving all the protected local models from the clients, the server should perform cross-correlation learning on each model instead of aggregating them with Fed-Avg, and then broadcast the updated models to the clients for the next round of federated training.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., et al. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308–318).

Bettini, C., Civitarese, G., & Presotto, R. (2021). Personalized semi-supervised federated learning for human activity recognition. arXiv preprint arXiv:2104.08094.

Blanco-Justicia, A., Sánchez, D., Domingo-Ferrer, J., & Muralidhar, K. (2022). A critical review on the use (and misuse) of differential privacy in machine learning. *ACM Computing Surveys*, [ISSN: 0360-0300] *55*(8), http://dx.doi.org/10.1145/3547139.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., & Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USeNIX security symposium (USeNIX security 19)* (pp. 267–284).

Chen, D., Yu, N., & Fritz, M. (2022). Relaxloss: Defending membership inference attacks without losing utility. arXiv preprint arXiv:2207.05801.

Chourasia, R., Enkhtaivan, B., Ito, K., Mori, J., Teranishi, I., & Tsuchida, H. (2021). Knowledge cross-distillation for membership privacy. arXiv preprint arXiv:2111.01363.

Dwork, C. (2011). A firm foundation for private data analysis. *Communications of the ACM*, *54*(1), 86–95.

Dwork, C., Kohli, N., & Mulligan, D. (2019). Differential privacy in practice: Expose your epsilons!. *Journal of Privacy and Confidentiality*, *9*(2).

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, *9*(3–4), 211–407.

Fredrikson, M., Jha, S., & Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (pp. 1322–1333).

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*(1), 1–58.

Gong, X., Sharma, A., Karanam, S., Wu, Z., Chen, T., Doermann, D., et al. (2022). Preserving privacy in federated learning with ensemble cross-domain knowledge distillation. *Vol. 36*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 11891–11899).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Guliani, D., Beaufays, F., & Motta, G. (2021). Training speech recognition models with federated learning: A quality/cost framework. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing* ICASSP, (pp. 3080–3084). IEEE.

Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., et al. (2018). Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hitaj, B., Ateniese, G., & Perez-Cruz, F. (2017). Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (pp. 603–618).

Hsu, T.-M. H., Qi, H., & Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335.

Hsu, T.-M. H., Qi, H., & Brown, M. (2020). Federated visual classification with real-world data distribution. In *Computer vision–ECCV 2020: 16th European conference, glasgow, UK, August 23–28, 2020, proceedings, part x 16* (pp. 76–92). Springer.

Huang, W., Ye, M., & Du, B. (2022). Learn from others and be yourself in heterogeneous federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10143–10153).

Jayaraman, B., & Evans, D. (2019). Evaluating differentially private machine learning in practice. In *28th USeNIX security symposium (USeNIX security 19)* (pp. 1895–1912).

Jebreel, N. M., & Domingo-Ferrer, J. (2023). FL-Defender: Combating targeted attacks in federated learning. *Knowledge-Based Systems*, *260*, Article 110178.

Jebreel, N. M., Domingo-Ferrer, J., Blanco-Justicia, A., & Sánchez, D. (2022). Enhanced security and privacy via fragmented federated learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Jebreel, N. M., Domingo-Ferrer, J., Sánchez, D., & Blanco-Justicia, A. (2024). LFighter: Defending against the label-flipping attack in federated learning. *Neural Networks*, *170*, 111–126.

Jia, J., Salem, A., Backes, M., Zhang, Y., & Gong, N. Z. (2019). Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security* (pp. 259–274).

Kanamori, S., Abe, T., Ito, T., Emura, K., Wang, L., Yamamoto, S., et al. (2022). Privacy-preserving federated learning for detecting fraudulent financial transactions in Japanese banks. *Journal of Information Processing*, *30*, 789–795.

Kaya, Y., & Dumitras, T. (2021). When does data augmentation help with membership inference attacks? In *International conference on machine learning* (pp. 5345–5355). PMLR.

Kaya, Y., Hong, S., & Dumitras, T. (2020). On the effectiveness of regularization against membership inference attacks. arXiv preprint arXiv:2006.05336.

Li, D., & Wang, J. (2019). Fedmd: Heterogenous federated learning via model distillation. arXiv preprint arXiv:1910.03581.

Lin, T., Kong, L., Stich, S. U., & Jaggi, M. (2020). Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, *33*, 2351–2363.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Minka, T. (2000). *Estimating a Dirichlet distribution*: *Technical report*, MIT.

Myalil, D., Rajan, M., Apte, M., & Lodha, S. (2021). Robust collaborative fraudulent transaction detection using federated learning. In *2021 20th IEEE international conference on machine learning and applications* ICMLA, (pp. 373–378). IEEE.

Naseri, M., Hayes, J., & De Cristofaro, E. (2020). Local and central differential privacy for robustness and privacy in federated learning. arXiv preprint arXiv:2009.03561.

Nasr, M., Shokri, R., & Houmansadr, A. (2018). Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security* (pp. 634–646).

Nguyen, A., Do, T., Tran, M., Nguyen, B. X., Duong, C., Phan, T., et al. (2022). Deep federated learning for autonomous driving. In *2022 IEEE intelligent vehicles symposium* IV, (pp. 1824–1830). IEEE.

Ouyang, X., Xie, Z., Zhou, J., Huang, J., & Xing, G. (2021). Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th annual international conference on mobile systems, applications, and services* (pp. 54–66).

Paass, G. (1988). Disclosure risk and disclosure avoidance for microdata. *Journal of Business & Economic Statistics*, *6*(4), 487–500.

Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., & Talwar, K. (2016). Semi-supervised knowledge transfer for deep learning from private training data. arXiv preprint arXiv:1610.05755.

Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., & Hinton, G. (2017). Regularizing neural networks by penalizing confident output distributions. arXiv preprint arXiv:1701.06548.

Ramaswamy, S., Mathews, R., Rao, K., & Beaufays, F. (2019). Federated learning for emoji prediction in a mobile keyboard. arXiv preprint arXiv:1906.04329.

Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., & Backes, M. (2018). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246.

Shejwalkar, V., & Houmansadr, A. (2021). Membership privacy for machine learning models through knowledge transfer. *Vol. 35*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 9549–9557).

Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy* SP, (pp. 3–18). IEEE.

Song, L., & Mittal, P. (2021). Systematic evaluation of privacy risks of machine learning models. In *30th USeNIX security symposium (USeNIX security 21)* (pp. 2615–2632).

Sozinov, K., Vlassov, V., & Girdzijauskas, S. (2018). Human activity recognition using federated learning. In *2018 IEEE intl conf on parallel & distributed processing with applications, ubiquitous computing & communications, big data & cloud computing, social computing & networking, sustainable computing & communications (ISPA/iUCC/bDCloud/socialCom/sustainCom)* (pp. 1103–1111). IEEE.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929–1958.

Suzumura, T., Zhou, Y., Baracaldo, N., Ye, G., Houck, K., Kawahara, R., et al. (2019). Towards federated graph learning for collaborative financial crimes detection. arXiv preprint arXiv:1909.12946.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818–2826).

Tang, X., Mahloujifar, S., Song, L., Shejwalkar, V., Nasr, M., Houmansadr, A., et al. (2022). Mitigating membership inference attacks by $\{Self-Distillation\}$ through a novel ensemble architecture. In *31st USeNIX security symposium (USeNIX security 22)* (pp. 1433–1450).

Tendick, P. (1991). Optimal noise addition for preserving confidentiality in multivariate data. *Journal of Statistical Planning and Inference*, *27*(3), 341–353.

Triastcyn, A., & Faltings, B. (2019). Federated learning with bayesian differential privacy. In *2019 IEEE international conference on big data (big data)* (pp. 2587–2596). IEEE.

Wei, K., Li, J., Ma, C., Ding, M., Chen, C., Jin, S., et al. (2021). Low-latency federated learning over wireless channels with differential privacy. *IEEE Journal on Selected Areas in Communications*, *40*(1), 290–307.

Xiao, Z., Xu, X., Xing, H., Song, F., Wang, X., & Zhao, B. (2021). A federated learning system with enhanced feature extraction for human activity recognition. *Knowledge-Based Systems*, *229*, Article 107338.

Yang, W., Zhang, Y., Ye, K., Li, L., & Xu, C.-Z. (2019). Ffd: A federated learning based method for credit card fraud detection. In *Big data–bigData 2019: 8th international congress, held as part of the services conference federation, SCF 2019, san diego, CA, USA, June 25–30, 2019, proceedings 8* (pp. 18–32). Springer.

Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium* CSF, (pp. 268–282). IEEE.

Zhang, H., Bosch, J., & Olsson, H. H. (2021). End-to-end federated learning for autonomous driving vehicles. In *2021 international joint conference on neural networks* IJCNN, (pp. 1–8). IEEE.