Learn to Forget: User-Level Memorization Elimination in Federated Learning

Yang Liu[1], Zhuo Ma*[1], Ximeng Liu*[2], Jianfeng Ma[1]
[1] School of Cyber Engineering, Xidian University,
bcds2018@foxmail.com, mazhuo@mail.xidian.edu.cn, jfma@mail.xidian,edu.cn.
[2] College of Mathematics and Computer Science, Fuzhou University, snbnix@gmail.com.
∗ Corresponding author.

***Abstract*—Federated learning is a decentralized machine learning technique that evokes widespread attention in both the research field and the real-world market. However, the current privacy-preserving federated learning scheme only provides a secure way for the users to contribute their private data but never leaves a way to withdraw the contribution to model update. Such an irreversible setting potentially breaks the regulations about data protection and increases the risk of data extraction. To resolve the problem, this paper describes a novel concept for federated learning, called memorization elimination. Based on the concept, we propose Forsaken, a federated learning framework that allows the user to eliminate the memorization of its private data in the trained model. Specifically, each user in Forsaken is deployed with a trainable dummy gradient generator. After steps of training, the generator can produce dummy gradients to stimulate the neurons of a machine learning model to eliminate the memorization of the specific data. Also, we prove that the additional memorization elimination service of Forsaken does not break the common procedure of federated learning or lower its security.**

***Index Terms*—Federated Learning, Memorization Elimination, Machine Learning**

I. Introduction

Due to the improved security and high efficiency, federated learning arises to the star of decentralized learning overwhelmingly as soon as being proposed [1]. In 2019, the research team of Google announced that the federated learning technique had reached the state to solve the applied learning problems over tens of millions level real-world users, and anticipates for the usage in billion-level applications [2]. Not surprisingly, federated learning shall lead the trend of decentralized online learning in the future market. More applications that deploy machine learning as a service can save a significant amount of costs on model training by utilizing federated learning.

Despite being popular, federated learning is still faced with a variety of security and application challenges, such as defending the user data reconstruction attack [3] and adapting federated learning to some specific application scenarios [4], [5]. In fact, these widely focused challenges can be concluded as one point, i.e., how to make federated learning securely and efficiently memorize the training data. However, compared to "memorize", the reversed process, "forget", seems to be neglected because of the difficulty to extract specific memorization from a trained model. As a result, there is now neither an effective way left for the user to withdraw the uploaded private data or a uniform indicator to evaluate the "forget" state.

The recently released data protection regulations, e.g., the California Consumer Privacy Act (CCPA) in the United States [6] and the General Data Protection Regulation (GDPR) in the European Union [7], clearly rule that the user should have the right to withdraw his private data if there is no special statement in the user agreement. These rules imply that the lack of forgetting mechanism for federated learning potentially violates the regulations and overlooks the fairness of the user to control its private data freely. Moreover, the training set of the user in federated learning may contain some unintended data that are private but not really useful to improve model accuracy. The memorization of the trained model about these unintended data greatly increases the possibility of the adversary to extract the user's private information. Take the language model as an example. The adversary can recover an inadvertently inserted sequence that is fully memorized after only 10^4 attempts [8]. In principle, there is none directive way for the server to manage the unintended memorization in the trained model. However, we can utilize an indirect method to resolve the problem, that is, making the user to withdraw the memorization of the trained model about the data required to be eliminated, which is referred as memorization elimination in this paper. Based on the method, the user can lower the risk of privacy leakage as much as possible.

To achieve memorization elimination, we propose Forsaken in this paper. Forsaken mainly achieves the following two breakthroughs: 1) it provides an efficient way to implement user-level memorization elimination for federated learning; 2) it defines a new indicator to evaluate the performance of memorization elimination. For memorization elimination, Forsaken implements it through a trainable generator. When a user needs to conduct memorization elimination, the generator is invoked to generate dummy gradients by learning the state of the trained model. The dummy gradient can be accumulated to the trained model in a similar way to the gradient computed based on the normal method used in most federated learning scheme, e.g., stochastic gradient descent (SGD) [1]. Therefore, our memorization elimination dose not break the common procedure of federated learning, and can also fit the existing

gradient privacy protection methods, such as [9], [10], [11]. The difference is that the normal gradient is used to improve the overall performance of the target model but the dummy gradient is used to stimulate the neuron units of the machine learning model to eliminate the memorization of the specific data. After giving a memorization elimination method, the current dilemma is that there is none of a quantified way to evaluate its performance. For this purpose, Forsaken defines forgetting rate to experimentally evaluate memorization elimination. By combining the concept of membership inference for machine learning [12], forgetting rate can well describes the rate of the data that are successfully eliminated from the memorization of the target model.

The contributions of this paper are summarized below.

- **Revocable Federated Learning.** We propose Forsaken, a federated learning framework that supports the user to manage the trained model's memorization about its data independently. Particularly, the memorization management does not need to retrain the machine learning model or break the common procedure of federated learning.
- **Memorization Elimination.** We formally give a novel definition of memorization elimination for federated learning, which contains a quantified indicator to measure the performance of memorization elimination, called forgetting rate. The indicator is derived from a current hot research spot, membership inference, and can be easily computed for an inspector.
- **Learn to Forget.** We design a trainable generator that can stimulate the machine learning model to eliminate the memorization of the specific data by learning the state of the target model to generate dummy gradients.
- **Performance Evaluation.** We implement Forsaken on five standard machine learning datasets to evaluate its performance. The results show that when we eliminate the memorization of 10 users' 200 samples (about 1% of the whole training set), Forsaken can averagely achieve 87.49% forgetting rate, and only cause less than 5% accuracy loss of the target model on the remaining data.

II. Background

In this section, we briefly overview the essential technical backgrounds of federated learning and membership oracle.

A. Federated learning

Federated learning is a kind of online machine learning framework that protects the privacy of training data providers from the "gradient" level [3]. Compared with the conventional training method with centralized data storage, federated learning avoids direct access to the private data of users and has significant advantages in training efficiency because of the distributed architecture. A standard framework of federated learning is shown in Fig. 1. Assume that there are m users $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, each of which owns its local private data set D_i. At the beginning of federated learning, the central serve $\mathcal{S}$ and $\mathcal{U}$ agree on an identical machine learning model architecture and objective function $\mathcal{L}(\cdot)$. Then, at each

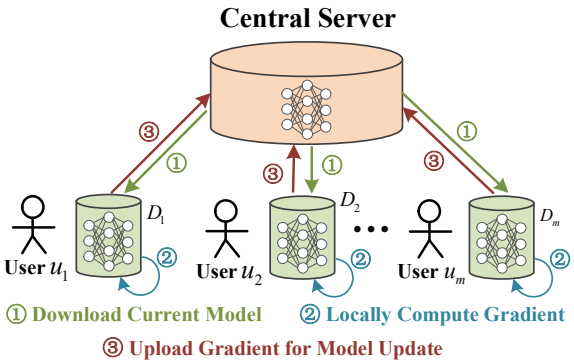


Fig. 1. A standard framework of federated learning

iteration, each user downloads the parameters of the current model from $\mathcal{S}$ and trains the model with D_i based on the gradient descent method. The model updates (i.e., gradients) obtained by each user are subsequently returned to $\mathcal{S}$, who averages the updates from all users and accumulates them to the current model according to the following equation.

$$\theta_{k+1} \leftarrow \theta_k - \eta \frac{1}{N} \sum_{i=1}^{m} n_i \cdot \nabla \mathcal{L}(f_{\theta_k}(D_i), \theta_k), \tag{1}$$

where θ_k is the parameters of the model f_{θ_k} trained for k iterations; η controls the learning rate; $\nabla \mathcal{L}(f_{\theta_k}(D_i), \theta_k)$ is the gradient uploaded by u_i at the k_{th} iteration; n_i is number of training samples owned by u_i; and $N = \sum_{i=1}^{m} n_i$. In such a way, federated learning implements distribute model training without uploading the user's private data, which greatly decreases the risk of privacy leakage.

Gradient Privacy. In federated learning, the gradient computed by each user is not completely secure. Through the generative adversarial network [13] or just a simple gradient optimizer [3], the adversary can reconstruct the private data from the gradient. To overcome the problem, the core idea of the existing methods [9], [10], [11] is as follows. First, before the gradient is sent to the server, the user masks it with the cryptographic tool, e.g., differential privacy [9] or secret sharing [10]. Then, the server (the adversary under the security model) aggregates the masked gradients and accumulates the result to the current model. During the process, the server cannot access any gradient of a specific user but obtain the final aggregated result. In this way, the success rate of such an attack can be reduced to be very slim. For security, we have to ensure that the above gradient privacy protection method can be applied to Forsaken, even when memorization elimination is invoked.

B. Membership Oracle

The goal of a membership oracle for machine learning is to determine the attribution of a certain sample, as defined in Definition 1 [14].

Definition 1 (Membership Oracle). Given a trained machine learning model f_θ and an arbitrary sample x, an ideal

membership oracle ψ outputs 1 if x belongs to the training set of f_θ; otherwise, outputs 0.

Initially, the membership oracle is intensively used as the membership inference attacker towards breaking the membership privacy of machine learning [14]. In Forsaken, we utilize it as the tool to evaluate the performance of memorization elimination. Up to now, the target-shadow method proposed by Shokri *et al.* [14] and its variants [12], [15] are still the mainstream to implement a membership oracle. Concretely, the inspector (or attacker) $\mathcal{I}$ collects a shadow data set coming from the same distribution as the training set used for the target model, and then, trains a shadow model with the shadow data set. Next, $\mathcal{I}$ uses the confidence vectors predicted by the shadow model for some members and non-members of the shadow model's training set to train a binary classifier. Taking a sample's confidence vector predicted by the target model as input, the binary classifier can infer whether the sample is a member of the training set for the target model or not, which is precisely a membership oracle. Note that the existing membership inference method can only be used in the classification task. Thus, unless stated, the machine learning models mentioned in the following paper are all classifiers by default. The research for the regression task is left for future work.

III. What is Memorization Elimination?

In this section, we introduce the meaning of memorization elimination. Further, we describe the adversary model and design goals for implementing secure memorization elimination in federated learning.

A. Motivation

Due to the excellent performance in both security and effect, federated learning reaches an unprecedented range of applications in the decentralized machine learning field. However, the mainstream of existing works about federated learning is still limited to how to make a model securely and effectively "memorize" something, e.g., enhancing the security and efficiency of gradient uploading [16] or extending the practicality of federated learning in some special scenarios [17], [18]. The reversed process, which we call *memorization elimination*, still stays in a blank stage.

Motivating Examples: Our study is first motivated by the practical requirement of federated learning. Up to now, federated learning is still a one-way trip for the user. Once a user has ever contributed its private data, no route of retreat is provided to withdraw the memorization of the trained model about these data. As mentioned before, such an irreversible setting on data memorization leaves a potential risk of violating some national data protection regulations while applying federated learning for applications..

Moreover, the lack of data forgetting mechanism increases the possibility of user data leakage. Consider a commonly discussed scenario of federated learning, training a generative sequence model on a text dataset used for automated sentence completion. Ideally, the dataset should not contain any rare-but-sensitive information about some individual users; alternatively, the trained model should not have strong memorization about this information and never emits it as sentence completion. In particular, if a user accidentally uses a sentence with the prefix "My bank password is ..." update the model, the output of the trained model would not predict the exact number in the suffix of the user's text as the most-likely completion when another user types the same prefix. Unfortunately, the research of [8] points out that it is hard for the current training environment to achieve the ideal condition, and the adversary can exploit this loophole to extract the sensitive information of the user efficiently. From the perspective of the real-world application, a general method to overcome the problem is to allow the user to check its data list used for federated learning and selectively withdraw the memorization of the trained model about these sensitive data.

Memorization Elimination: To explain what memorization elimination means for federated learning, we first illustrate what the memorization is for machine learning. Abstractly, the training process of a machine learning model is simply the memorization enhancement process. Through a series of iterative learning steps, the neurons of a machine learning model obtain some forms of memorization about the pattern of the training set; even the pattern of the training is randomized [19]. Further, a trained model intends to output what is in accordance with its memorization, i.e., strongly suggest what training data is used (see the concept of membership oracle [12], [14]). Moreover, take the classification task as an example. We can reasonably derive Definition 2 that if some samples are "totally forgotten", the model can only guess the samples' categories; in other words, the model will not specifically suggest these samples to be any category.

Definition 2. *For a p-classification machine learning task, we say that a sample x is totally forgotten by a machine learning model f_θ if the confidence vector $(y_1, y_2, ..., y_p) = f_\theta(x)$ satisfies $y_1 \approx y_2 \approx ... \approx y_p \approx \frac{1}{p}$.*

Inspired by the above illustration, the memorization elimination for federated learning can be roughly understood as forcing a trained model to forget the pattern of the specific private data owned by the user. Referring to [8], we treat the private data required to be withdrawn as out-of-scope training data, which are usually a tiny part of the whole training data. Federated learning is not intended to make the model memorize any such data as soon as the memorization elimination operation is called. Based on the principle, we present the following definition, k-elimination, i.e., conducting memorization elimination at the k_{th} iteration.

Definition 3 (k-Elimination). *Given a machine learning model $f_{\theta_{k-1}}$ trained after $k-1$ iterations, a membership oracle ψ, a training set D and an elimination dataset $D' \subset D$, k-elimination is perfectly performed if there exists an elimination function φ that can output $f_{\theta_k} = \varphi(f_{\theta_{k-1}})$ where for each $x \in D'$, we have $\psi(f_{\theta k-1}(x)) = 1$ and $\psi(f_{\theta k}(x)) = 0$;*

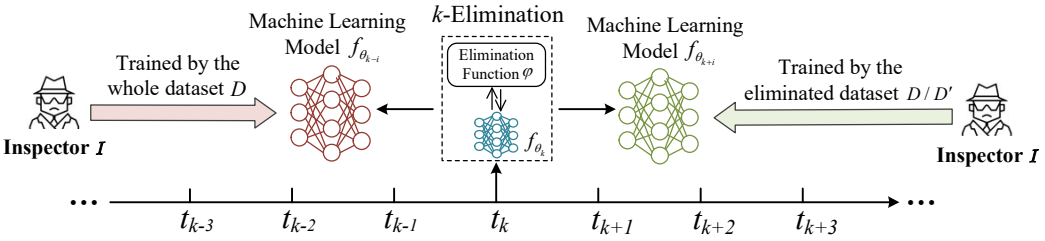


Fig. 2. k-elimination: Express the training steps of federated learning as a time sequence. k-elimination implements that before t_k, the inspector $\mathcal{I}$ regards the model $f_{\theta_{k-i}}$ as being trained by the whole dataset D; after t_k, $\mathcal{I}$ regards the model $f_{\theta_{k+i}}$ as being trained by the eliminated dataset D/D'.

for each $x \in D$, $f_{\theta_k}(x) = f_{\theta_{k-1}}(x)$.

Suppose that there is an inspector $\mathcal{I}$ who owns an ideal membership oracle. As shown in Fig. 2, Definition 3 rules that from the perspective of $\mathcal{I}$, the machine learning models obtained after the k_{th} iteration are trained by D/D'; the models obtained before the iteration are not affected, i.e., protecting the backward privacy of the eliminated data. We say that backward privacy is enough because it ensures the final output model of federated learning does not contain the contributions of the eliminated data, Furthermore, ideal k-elimination should have good directivity, that is, it forces the model to forget the pattern of specific data after a particular iteration but does not influence the performance on the remaining data.

B. Adversary Model

The adversary model of Forsaken is inherited from the standard federated learning scheme [10], [20]. The details are given as follows.

Learning Scenario. Prior to introducing the adversary model, we describe the entities of the learning scenario in Forsaken. As illustrated in Fig. 1, there are total $m \geq 2$ users and one central server $\mathcal{S}$ that are agreed on the same training objective in Forsaken. For generality, the data of all users are assumed to be non-IID distributed (not independent and identically distributed), which is consistent with the setting of standard federated learning schemes.

Adversary Description. We assume that the adversary in Forsaken is secure under the following security assumptions.

1) The adversary basically follows the standard *curious-but-honest* model [21], and can be the server or any user. The *curious-but-honest* adversary honestly conducts the predetermined protocol steps but never misses an opportunity to infer the honest user's data from the received legitimate messages.
2) The adversary is restricted from polynomial-time computation capacity. Moreover, we assume that there are secure channels between the server and the user for transmitting model updates[1].
3) The device of the user (e.g., smartphone) provides confidentiality guarantees for the storage of the local private data (both training data and the model) and is physically secure towards the adversary. Such local isolation is the premised condition for the security of federated learning.

[1]This paper focuses on memorization elimination of federated learning. For secure channel construction, please refer to [9], [10].

C. Design Goal.

Forsaken is designed to achieve the following two goals.

GOAL 1 (SECURITY INHERITANCE). *Forsaken does not break the common procedure of federated learning, and meanwhile, ensures that there is no polynomial-time algorithm for a curious-but-honest adversary $\mathcal{A}$ to infer any data of the user.*

The first goal is to avoid user privacy leakage, which is also the goal of the conventional federated learning scheme. In other words, we have to guarantee that Forsaken is at least as secure as other federated learning schemes in the *curious-but-honest* model. According to our adversary model, the attack surface of $\mathcal{A}$ can only be concentrated on the user gradient. Therefore, to achieve this goal, Forsaken should be compatible with the existing secure aggregation schemes to protect the security of the gradient. Besides the above goal, the other goal of Forsaken is to provide an additional memorization elimination service for the user, which is defined below.

GOAL 2 (MEMORIZATION ELIMINATION). *An arbitrary user u_0, no matter honest or curious-but-honest, can choose to conduct k-elimination on his data at any iteration of federated learning, and the additional operation of k-elimination does not influence the learning process of other users.*

The second goal is to implement secure memorization elimination based on the k-elimination. Remark that the elimination function given in Definition 3 is considered to be an ideal function to achieve our goal. Nevertheless, considering the meaning of memorization for machine learning, it is not realistic to perfectly implement such a function. Alternatively, in the next section, we define a simple optimizer to approximate the function.

IV. MEMORIZATION ELIMINATION IN FEDERATED LEARNING

In this section, we present the design details of memorization elimination in Forsaken.

A. Measuring Memorization Elimination

Prior to introducing Forsaken, we define a quantified indicator to measure the performance of the k-elimination method, called forgetting rate (FR).

For both the human brain and machine learning, memorization is an abstract concept that is hard to be directly measured. However, a widely accepted fact is that the memorization can only be formed from the known objects (the training samples for machine learning). Even though the human brain or machine learning can identify a never seen object, the association ability is derived from the known memorization in some ways. Thus, if we successfully eliminate the memorization of some specific data, the most intuitive reflection is that these data are transformed from "known" to "unknown" as illustrated in Fig. 2,. In such a case, the eliminated data are ensured to be no longer related to the memorization of the target model anymore. Naturally, the transformation rate between "known" and "unknown", i.e., FR, can be directly utilized to evaluate the performance of memorization elimination. To compute FR, it is necessary to find a tool to identify whether a given sample is "known" or not. Definition 1 precisely describes such an ideal tool, that is, membership oracle. Nonetheless, the dilemma is that none of the existing membership inference algorithms can implement an ideal membership oracle. Therefore, FR combines both the performance of the membership oracle and the transformation rate, which can be mathematically expressed as the following equation.

$$FR = \frac{BT}{BT + BF} \times \frac{AF}{AT + AF} \times 100\%, \tag{2}$$

where the meanings of TP, FN, TN and FP are given in Table I.

To further understand FR, we can refer to a common evaluation indicator of machine learning, recall rate [22]. In machine learning, the recall rate is the fraction of the total amount of positive instances that are correctly classified as "true". Before conducting memorization elimination, the positive instances are the training samples required to be eliminated. $\frac{BT}{BT+BF}$ expresses the recall rate of the membership oracle on these instances that are correctly identified as "known". Correspondingly, $\frac{AF}{AT+AF}$ indicates the recall rate of the instances that are correctly transformed to be "unknown" after conducting memorization elimination. Literally speaking, FR gives a qualified indicator to measure how many samples are changed from the memorized set (training set) to the unknown set (testing set) after memorization elimination. Notably, as the computation of FR involves the inference of a membership oracle that cannot be owned by a normal user, it is unpractical to directly utilize FR as an objective function of memorization elimination. To overcome the problem, we define a new objective function based on Definition 2 in Section IV-C.

B. Intuition of Memorization Elimination

Our memorization elimination method is originally inspired by the nature of the human brain. Consider the laws of human memory. The most intuitive method to make the brain to forget something is to focus on other things. Similarly, to make a model eliminate the memorization about some specific data, the simplest way is to remove these data from the training set, and then, continue to train the model with the remaining dataset. Theoretically, after countless training steps, the model can gradually weaken the memorization of the pattern about the removed data. The advantage of such a natural memorization elimination method is that it causes little impact on the original performance of the target model. However, its disadvantage is also obvious, that is, suffering a great loss in efficiency and practicality. A well-trained model is designed to have a similar feature to the deep cognition behaviour of the human brain (e.g., the language ability). To make a well-trained model forget something in a natural way, the required time is usually too long to be accepted for practical applications.

TABLE I
NOTATION TABLE

Notations	Descriptions
BT	The number of eliminated training samples that are predicted to be **TRUE** by ψ **BEFORE** conducting φ.
BF	The number of eliminated training samples that are predicted to be **FALSE** by ψ **BEFORE** conducting φ.
AT	The number of eliminated training samples that are predicted to be **TRUE** by ψ **AFTER** conducting φ.
AF	The number of eliminated training samples that are predicted to be **FALSE** by ψ **AFTER** conducting φ.

$\psi \rightarrow$ the membership oracle; $\varphi \rightarrow$ the k-elimination function.

To verify our analysis, we use the natural memorization elimination method to conduct a simple experiment with the standard image classification dataset, CIFAR-10. The detailed experiment setting is as listed in Section VI. In the experiment, we first train a target model for 25 epochs according to the federated learning procedure. Then, at the 26_{th} epoch, we randomly select a user and remove it from the candidate data provider set. Next, we continue to make the remaining users train the target model for the same number of epochs as previous training. The result shown in Fig. 3 state that only a very little increase on the FR is obtained from the extra epochs of training, which is consistent to the above analysis.

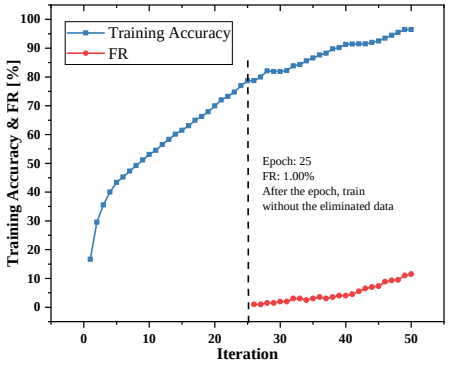


Fig. 3. Memorization elimination with the natural method for CIFAR-10. Memorization elimination occurs at epoch 25..

To overcome the defect of natural memorization elimination, it is necessary to enhance the "forgetting" strength towards the data of the selected user. To achieve this goal, we refer

to the stochastic gradient descent (SGD) method used for model updates in federated learning (mathematically expressed as Eq. 1 in Section II). From Eq. 1, it is observed that the memorization of a trained model in federated learning is formed by the gradients uploaded by the users. Therefore, a forcible method implement memorization elimination is to withdraw the gradients that the user has contributed for model update, and the process can be completed in a normal training epoch of federated learning as given in Eq. 3.

$$\theta_{k+1} \leftarrow \theta_k - \eta \frac{1}{N}[\sum_{i=1, i\neq u_0}^{m} \nabla \mathcal{L}(f_{\theta_k}(D_i), \theta_k) + \sum_{j=1}^{k} \nabla \mathcal{L}(f_{\theta_j}(D_0), \theta_j)]. \tag{3}$$

where u_0 is the user whose data memorization is eliminated at the k_{th} iteration, and $\nabla \mathcal{L}(f_{\theta_j}(D_0), \theta_j)$ is the gradient uploaded by u_0 at the j_{th} iteration. As shown in Fig. 4, using the forcible elimination method to implement k-elimination can indeed achieve a higher FR. However, since the SGD method used in federated learning is time-sequence related and irreversible, the performance loss caused by the method on the target model is also obvious. Although the above method does not obtain a considerable result, it inspires us to implement Forsaken from the gradient level.

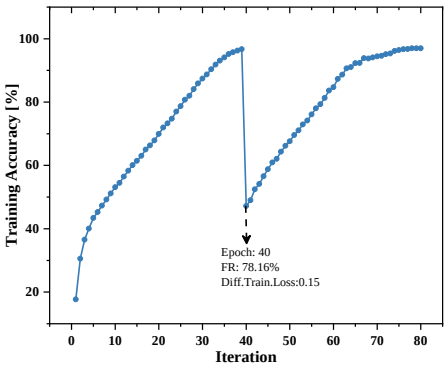


Fig. 4. Memorization elimination with the forcible method for CIFAR-10. Memorization elimination occurs at epoch 40.

C. Generator: Learn to Forget

We now present our memorization elimination methodology that relies on a trainable generator $\mathcal{G}$.

Dummy Gradient. From the forcible elimination method, we observe that gradient is a powerful component to influence the memorization in federated learning. Therefore, Forsaken also utilizes the gradient to implement memorization elimination. The difference is that the gradient used for memorization elimination in Forsaken is not computed based on the normal gradient method but specially produced by a dummy gradient generator. Initially, the dummy gradient is initialized with a series of tiny random values that have the same size as the normal gradient. By learning the state of the target model, $\mathcal{G}$ successively adapts the dummy gradient to remodel the target model towards a predefined direction to implement directive and lossless memorization elimination. The following presents the principle of $\mathcal{G}$ to generate the dummy gradient.

Learn to Forget. The design of dummy gradient generator $\mathcal{G}$ is inspired by the neurological "active forgetting [23]" mechanism of the human body. Different from the hysteresis of natural forgetting (also known as "passive forgetting"), active forgetting is vigorous and can eliminate all traces and engram cells for a given memory. The active forgetting process is as follows [23]. To eliminate the specific memorization stored in the engram cells, human body let the dopamine neurons, called forgetting cells, to produce a kind of special dopamine. The dopamine serves as the forgetting signal that stimulates the remodeling of the engram cells to accelerate memorization elimination. Based on the feedback of the engram cells, the forgetting cells learn to regularize the production of dopamine to avoid unexpected forgetting.

Protocol 1 Memorization Elimination Generator (`DummyG`)

Input: The target model f_{θ_0} and its corresponding trainable parameters θ_0; the elimination set $D' \subset D$; the user u_0 that owns D'; the maximum training epochs T.

1: u_0 uses small values to initialize the dummy gradient μ_0, and then, do the following iteration.
2: **for** $t \leftarrow 0$ to T **do**
3: Update $\theta_{t+1} \leftarrow \theta_t - \eta \frac{1}{n} \mu_t$, where n is the size of D'.
4: Compute $\Upsilon = \{y_i | y_i \leftarrow f_{t+1}(x_i), x_i \in D'\}$.
5: Optimize $\mathcal{G}$ based on the objective function $\mathcal{L}_o = \arg\min_{\mu_d}(\|\mu_d - \theta_t\|^2 + \|\Upsilon - \mathcal{P}\|^2 + \Omega)$.
6: Output a new dummy gradient $\mu_t \leftarrow \mathcal{G}(\mu_t, \Upsilon)$.
7: **end for**
8: Return the summing of dummy gradients $\sum_{i=0}^{T} \mu_t$.

Similarly, $\mathcal{G}$ plays the forgetting cell role in our memorization elimination mechanism for machine learning. To eliminate the memorization of the trained model about some specific data, $\mathcal{G}$ successively produces a certain amount of dummy gradients whose function is like the dopamine. The dummy gradients can be applied to the neuron units of the machine learning model and stimulate them to remodel themselves to eliminate the memorization for the given data. Besides, $\mathcal{G}$ can learn the state of the neuron units about memorization elimination from the feedback of the target model. According to the learning result, $\mathcal{G}$ regularizes the size of dummy gradients to avoid unexpected elimination. After several epochs of learning, the target model shall quickly lose the memorization of the specific data because of the impact of $\mathcal{G}$. The detailed construction method of $\mathcal{G}$ is illustrated below, and listed in Protocol 1 (`DummyG`).

Represent the target model whose memorization has to be eliminated as f_{θ_0}, where θ_0 is the trainable parameter set of the target model. In the training process, $\mathcal{G}$ first initializes the dummy gradient μ_0 with a series of small random values (empirically set to less than 10^{-3}). The size of μ_0 is the same as θ_0. Then, at the t_{th} iteration of training, $t \geq 0$, $\mathcal{G}$

Protocol 2 Privacy-preserving Federated Learning with Memorization Elimination (`SecForget`)

Input: The m users, each of which owns a local dataset D_j with size of n_j; the server $\mathcal{S}$; the learning rate η; the maximum epochs for memorization elimination T; a gradient privacy protection function $\phi(\cdot)$.

$\mathcal{S}$ initializes a machine learning model f_{θ_0} and determines the loss function $\mathcal{L}(\cdot)$.
for each iteration $i = 0, 1, ...$ **do**
  $\mathcal{S}$ randomly selects k users $\mathcal{U}' \subset \mathcal{U}$ and publishes f_{θ_i}, $\mathcal{L}(\cdot)$ to $\mathcal{U}'$.
  for each user $u_j \in \mathcal{U}'$ **do**
    if u_j chooses to eliminate the memorization of f_{θ_i} about its data **then**
      Ask $\mathcal{S}$ for the permission of memorization elimination.
      Compute the dummy gradient $\nabla\mathcal{L}(D_j, \theta_i) \leftarrow$`DummyG`$(f_{\theta_i}, D_j, T)$, and then, send $\nu_j \leftarrow \phi(\frac{1}{n_j} \cdot \nabla\mathcal{L}(D_j, \theta_i))$ to $\mathcal{S}$.
    else
      Compute the normal gradient with the local data $\nabla\mathcal{L}(D_j, \theta_i)$, and then, send $\nu_j \leftarrow \phi(\nabla\mathcal{L}(D_j, \theta_i))$ to $\mathcal{S}$.
    end if
  end for
  Receiving $\mathcal{V} = \{\nu_1, \nu_2, ..., \nu_k\}$, $\mathcal{S}$ uses the secure aggregation function `SecAgg` corresponding to $\phi(\cdot)$ to compute $\Lambda \leftarrow \sum_{i=1}^{m} \nabla\mathcal{L}(D_j, \theta_i) \leftarrow$`SecAgg`$(\mathcal{V}, k)$, and update $\theta_{i+1} \leftarrow \theta_i - \eta \cdot \frac{1}{N} \cdot \Lambda$, where $N = \sum_{j=1}^{k} n_j$.
end for
Return the trained model f_{θ_i}.

sends μ_t to f_{θ_t}, and f_{θ_t} uses μ_t to update θ_t in a similar way to the SGD method used in federated learning, i.e., $\theta_{t+1} \leftarrow \theta_t - \eta\frac{1}{n}\mu_t$, where η is a hyperparameter that controls the learning rate (mentioned in Eq. 1) and n is the size of D'. Next, we input each sample of the elimination set $x_i \in D'$ into $f_{\theta_{t+1}}$ and compute $\Upsilon = \{y_i | y_i \leftarrow f_{t+1}(x_i), x_i \in D'\}$. Notably, since Forsaken only focuses on the task with finite and discrete outputs as mentioned before, the output of the target model $y_i \in \Upsilon$ is limited to be a confidence vector with finite dimensions by default. Υ is feedback to $\mathcal{G}$ and used to optimize the dummy gradient by minimizing the following objective.

$$\mathcal{L}_o \leftarrow \arg\min_{\mu_d}(\|\Upsilon - \mathcal{P}\|^2 + \|\mu_d - \theta_{t+1}\|^2 + \Omega), \quad (4)$$

$\mathcal{L}_o$ is differential w.r.t. the dummy gradient μ_d can be optimized with the standard gradient based method, such as L-BFGS [24] and Adam [25]. Here, the distance $\|\Upsilon - \mathcal{P}\|^2$ rules the optimization objective of the training, where $\mathcal{P} = (\frac{1}{p}, \frac{1}{p}, ..., \frac{1}{p})$ is the confidence vector of a "perfectly" forgotten sample according to Definition 1. $\|\mu_d - \theta_{t+1}\|^2$ constrains the maximum step length and the optimization direction. $\Omega = \|\theta_{t+1} - \theta_0\|^2$ is a regularization item that punish the changes on the original model to avoid performance loss. Finally, after completing the above iteration, we can get the dummy gradient that can eliminate the memorization of the target model about D'.

D. Memorization Elimination in Federated Learning

With the dummy gradient generator, Forsaken can simply implement memorization elimination in federated learning as stated in Protocol 2 (`SecForget`).

The procedure of `SecForget` refers to the privacy-preserving federated learning scheme proposed by Google [1], [10] but provides additional memorization elimination option for each user. First, $\mathcal{S}$ initializes the trainable parameters of a machine learning model and determines its loss function $\mathcal{L}(\cdot)$. Then, at each iteration, k users are selected to participate in the model training. Each selected user u_j has two options. If choosing to eliminate the memorization of the model about its data, u_j invokes the generator defined in `DummyG` to produce the dummy gradient; otherwise, u_j uses its local data to compute the gradient according to the standard SGD method. In principle, the user is not allowed to choose memorization elimination at the first few epochs of federated learning. Furthermore, since the plaintext gradient of a single user can be used to derive the user's data [26], most of the existing federated learning schemes utilize some cryptographic tools to protect the privacy of the gradient before sending it to $\mathcal{S}$. From the design of `DummyG`, it can be observed that the dummy gradient can be treated in the same way as the normal gradient. Therefore, the mainstream gradient privacy protection methods, e.g., differential privacy [9] secret sharing [10] and homomorphic encryption [11], can be directly used in `SecForget`. Finally, $\mathcal{S}$ can invoke the secure aggregation algorithm `SecAgg` that corresponds to the gradient privacy protection function to get the summing result of all gradients and use it update the current model according to Eq. 1.

V. Theoretical Analysis

In this section, we theoretically prove that Forsaken can achieve it two design goals. Further, we discuss that Forsaken can be extended to be an attack tool to threaten the security of federated learning.

Data Privacy As defined in Goal 1, the security goal of Forsaken is to inherit the same data privacy level as the existing privacy-preserving federated learning schemes. From the procedure of `SecForget`, it can be seen that the only difference between Forsaken and other federated learning schemes is the dummy gradient produced by `DummyG`. Thus,

we can give a formal definition of the security of Forsaken as follows.

DEFINITION 4 (SECURITY OF FORSAKEN). *We say that Forsaken is as secure as the existing federated learning schemes if the dummy gradient produced by* `DummyG` *can be treated in the same way as the true gradient produced by the standard gradient method with the privacy protection function.*

Proof. According to the common gradient method used in federated learning, like SGD [27], the normal gradient given in Eq. 5, is the derivative of the loss function $\mathcal{L}$ with respect to the trainable parameters θ, where X is the training sample.

$$\nabla\mathcal{L}(f_\theta(X),\theta) \Leftarrow \frac{\partial\mathcal{L}(f_\theta(X),\theta)}{\partial\theta}. \tag{5}$$

As for Forsaken, the generator $\mathcal{G}$ in `DummyG` also uses the gradient based optimizer to generate the dummy gradient. The difference is that $\mathcal{G}$ treats the gradient as the trainable target, not the model parameters. Therefore, the dummy gradient is actually the derivative of the loss function defined in Eq. 4 with respect to the gradient, which is mathematically expressed as Eq. 6.

$$\mu \Leftarrow \frac{\partial\mathcal{L}_o(f_\theta(X),\nabla\mathcal{L}(f_\theta(X),\theta))}{\partial\nabla\mathcal{L}(f_\theta(X),\theta)}. \tag{6}$$

From the perspective of the server, both the dummy gradient and the normal gradient can be regarded as a series of numerical matrixes that have the same function and same size. Thus, the dummy gradient can also be correctly applied to the gradient privacy protection function like other federated learning schemes [9], [10]. Moreover, after applying the gradient privacy protection function, the server can only access the aggregated gradients. From the aggregation result, it is impossible to separate the gradient of a specific user in polynomial time. Further, we can derive that the dummy gradient and normal gradient in `SecForget` are computationally indistinguishable. In conclusion, Forsaken is at least as secure as the existing federated learning schemes. □

Memorization Elimination. As mentioned before, Forsaken implements memorization elimination (i.e., Goal 2) based the dummy gradient generator whose core is defined in Eq. 4. We now explain the correctness of Eq. 4 to lead the generator to generate effective dummy gradients for memorization elimination. Eq. 4 is mainly composed of two parts. The first part of Eq. 4 is based on Definition 2, which describes the output of a machine learning model for a "totally forgetten" sample. Definition 2 is reasonable because if we have no knowledge about a given sample, the only way to judge its category is guessing. A guessing output is always neutral. For example, if a machine learning model learns nothing about some specific samples in a 2-classification task, its output for these samples shall not bias towards any side, that is, approximating the guessing vector $(0.5, 0.5)$. Notably, learning nothing means having no relevant memorization and does not mean having the wrong memorization that may bias the output to the wrong side. Therefore, the first part makes the dummy gradient to change the specific sample from "memorized" to "totally forgetten". Then, the second part of Eq. 4 is used to punish the parameter change on the target model. In this way, we can minimize the performance loss on the target model caused by the memorization elimination. In the next section, we further use experiments to prove the correctness of the above setting on memorization elimination.

Extended Discussion. Similar to the membership oracle, our dummy gradient generator can also be modified from a positive tool to an attack tool that can launch a kind of inconspicuous data poison attack towards federate learning.

Usually, the data poison attack is launched by the malicious data provider [28]. By uploading the gradient computed with mislabeled data, the attacker can mislead the server to train a machine learning model that always gives wrong outputs for some specific data. However, such an attack method has a defect that it causes obvious performance loss on the trained model for other data [29]. Our dummy gradient generator provides an alternative method to overcome the defect. It can be discovered that if a malicious user slightly changes the objective function ($\mathcal{P}$ in Eq. 4) according to its requirement and successively uploads the dummy gradient to the server, the final trained model can also be misled to identify some specific data wrongly. Even worse, as stated in the experiments of Section VI, the attack causes little impact on the model performance, which makes it difficult to be detected. Although the active attack is out of the scope of this paper, we still give an intuitive way to defend the dummy gradient based poison attack in reality, i.e., observing the running time of each user. Since the dummy gradient generation process is more complicated than the normal gradient and cannot be previously computed, the server can detect the malicious user by judging whether its gradient generation time is unusually much longer than other users'.

VI. EXPERIMENT IMPLEMENTATION

In this section, comprehensive experiments are conducted to prove that Forsaken can achieve Goal 2, memorization elimination.

A. Experiment Setup.

We utilize five different machine learning datasets to conduct our experiments, namely CIFAR-10[2], CIFAR-100, MNIST[3] and News[4]. Among them, the former three datasets are standard image classification datasets. The News dataset is a commonly used text classification and clustering dataset that has a balanced class distribution. Since the raw News dataset is all string-type data that cannot be directly applied for machine learning, we preprocess it by encoding the raw data into numerical matrixes in a similar way to [12]. The detailed information about the dataset is listed in Table II. In particular, to evaluate the FR indicator of our scheme, we have to

[2]https://www.cs.toronto.edu/ kriz/cifar.html
[3]http://yann.lecun.com/exdb/mnist/
[4]http://qwone.com/ jason/20Newsgroups/

1) *Convolution:* Input image 28×28, windows size 5×5, number of output channel 10.
2) *ReLU:* Calculate ReLU for each input.
3) *MaxPooling:* Window Size $1 \times 2 \times 2$.
4) *Convolution:* Windows size 3×3, number of output channel 20.
5) *ReLU:* Calculate ReLU for each input.
6) *Fully Connected Layer:* Fully connected the incoming 2000 nodes to the outgoing 500 nodes.
7) *Fully Connected Layer:* Fully connected the incoming 500 nodes to the outgoing 10 nodes.

Fig. 5. The neural network trained for MNIST

1) *Convolution:* Input image $3 \times 32 \times 32$, windows size 3×3, number of output channel 64.
2) *BatchNormal + ReLU:* Calculate BatchNormal and ReLU for each input.
3) *MaxPooling:* Window Size $1 \times 2 \times 2$.
4) *Convolution:* Windows size 3×3, number of output channel 128.
5) *BatchNormal + ReLU:* Calculate BatchNormal and ReLU for each input.
6) *MaxPooling:* Window Size $1 \times 2 \times 2$.
7) *Convolution:* Windows size 3×3, number of output channel 256.
8) *BatchNormal + ReLU:* Calculate BatchNormal and ReLU for each input.
9) *MaxPooling:* Window Size $1 \times 2 \times 2$.
10) *Convolution:* Windows size 3×3, number of output channel 512.
11) *BatchNormal + ReLU:* Calculate BatchNormal and ReLU for each input.
12) *MaxPooling:* Window Size $1 \times 2 \times 2$.
13) *Fully Connected Layer:* Fully connected the incoming 512 nodes to the outgoing 100 nodes.

Fig. 6. The neural network trained for CIFAR100

train a membership oracle. Therefore, we train a membership oracle for each dataset according to the target-shadow method presented in [12]. Specifically, we first randomly partition all of the target dataset into two halves. One half is used to train the target model. The other is used to train the shadow model. Then, we train the target and shadow models in the white-box mode, i.e., using the same architecture and hyperparameters. Finally, with the outputs of the two models as the training and testing sets, we train an XGBoost [30] model to serve as the membership oracle. The interested readers can refer to [12] for details.

Corresponding to the five datasets, we use five neural networks with different architectures. To comprehensively evaluate the performance of Forsaken, the five neural networks are deliberately set to have obvious differences in parameter size. The neural networks used for processing MNIST, CIFAR-100 and News are given in Fig. 5, Fig. 6 and Fig. 7. For CIFAR-10, we use a previously proposed deep network architecture, called VGG-13 [31].

Moreover, the dummy gradient is initialized with the normal random generator provided by Numpy (limited to less than 10^{-3} degree). The optimizer used as the memorization elimination generator is L-BFGS [24]. All experiments are implemented with Pytorch, an open-source machine learning library of python. To simulate the federated learning process, we randomly split the training set of each dataset into a series of subsets with the same size (100 by default), each of which is assigned to a user as the local dataset. The users cooperatively

1) *Embedding:* Input word vector 1×1000, the output word embedding is 1×100.
2) *Transpose Convolution:* Windows size 5×5, number of output channel 128.
3) *ReLU:* Calculate ReLU for each input.
4) *MaxPooling:* Window Size $1 \times 5 \times 5$.
5) *Convolution:* Windows size 5×5, number of output channel 128.
6) *ReLU:* Calculate ReLU for each input.
7) *MaxPooling:* Window Size $1 \times 5 \times 5$.
8) *Convolution:* Windows size 5×5, number of output channel 256.
9) *ReLU:* Calculate ReLU for each input.
10) *MaxPooling:* Window Size $1 \times 35 \times 35$.
11) *Fully Connected Layer:* Fully connected the incoming 128 nodes to the outgoing 128 nodes.
12) *Fully Connected Layer:* Fully connected the incoming 128 nodes to the outgoing 20 nodes.

Fig. 7. The neural network trained for News. The layers in the networks are all used in the 1-dimension mode.

train the target model (or shadow model) according to the steps of Protocol 2. Note that memorization elimination is always applied for the target model in the following experiments. The default elimination size is 200. The eliminated samples are came from some randomly selected users. Each selected user contributes 20 randomly chosen samples to make up the elimination set.

TABLE II
DATASET INFORMATION

Name	No. of Instances	Features	Classes
MNIST	70000 (10000 for testing)	28×28	10
News	11314 (2262 for testing)	1000	20
CIFAR-10	60000 (10000 for testing)	32×32	10
CIFAR-100	60000 (10000 for testing)	32×32	100

B. Performance of Memorization Elimination

For memorization elimination, what the user mostly cares about is how many data are successfully forgotten by the trained model, which can be evaluated by FR, the indicator proposed in Section IV. In the FR evaluation, we emphatically observe the performance of Forsaken on the well-trained model. To achieve this, we first train couples of target models and shadow models with the prepared datasets based on the common procedure of federated learning given in Protocol 2 and carefully avoid overtraining. Then, we randomly select one user to call the memorization elimination service. Table III summarizes the FR of the selected user's data after applying the dummy gradients on the trained models in different datasets. The result shows that Forsaken can transfer more than 90% eliminated samples from the training set to the known set. However, considering the accuracy of membership inference, the FR of Forsaken is a little lower than the expected value. In addition, it can be discovered that the size of the training set or the size of the dummy gradient required to be simulated by the generator does not strongly influence the performance of Forsaken. Even for the VGG-13 that has 14.09M dummy gradient required to be simulated, Forsaken can still effectively eliminate the specific memorization.

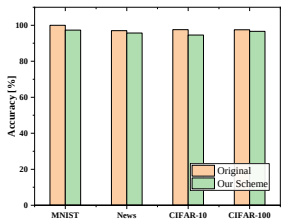
Different from the user, the central server concerns more about the performance loss of the target model after conducting memorization elimination. To evaluate this merit, we record the training accuracy and training loss of the target model before and after memorization elimination, shown in Fig. 8(a) and Fig. 8(b). Compared with two intuitive methods proposed in Section IV, the performance loss (reflected by the training accuracy and training loss in the experiments) caused by memorization elimination in Forsaken is negligible. Taken together, the above experiment results indicate that Forsaken can basically satisfy our design goal for memorization elimination.

TABLE III
FORGETTING RATE OF FORSAKEN ON DIFFERENT DATASETS

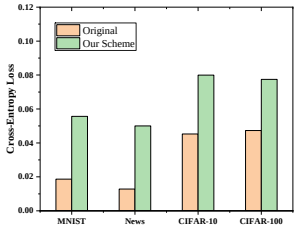
Dataset	Dummy Gradient Size	B.R.R	A.R.R	FR
MNIST	0.96M	85.54%	99%	84.68%
News	0.24M	86.37%	99.67%	86.01%
CIFAR-10	14.09M	87.33%	98%	85.58%
CIFAR-100	9.02M	95.56%	95.5%	91.26%

B.R.R $\rightarrow \frac{BT}{BT+BF}$; A.R.R $\rightarrow \frac{AF}{AT+AF}$.

$FR =$ B.R.R $\times$ A.R.R. $\times 100\%$, i.e., Eq. 2 in Section IV.



(a) Training accuracy change after memorization elimination.



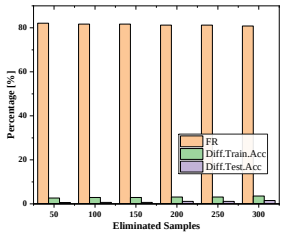
(b) Training loss change after memorization elimination.

Fig. 8. The performance change of the target model after conducting Forsaken, Baseline-1 and Baseline-2.

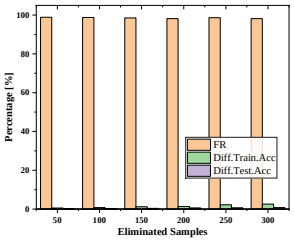
C. Effect Factors for Memorization Elimination

After evaluating the whole performance, we further conduct experiments to analyse some key factors that possibly affect the performance of Forsaken, namely the eliminated data size, the training iterations of the dummy gradient generator and overtraining. The experiments in this section are mainly performed on two datasets, MNIST and News, which severally represent the two classical tasks of machine learning, image classification and text classification.

Eliminated Data Size. For Forsaken, the complexity of the dummy gradient generator is positively related to the eliminated data size. Here, we conduct experiments to test the performance change of Forsaken with variable eliminated data sizes. Usually, there can be only a small part of the users that launch memorization elimination at the same time, Therefore, we set the eliminated size varied from 50 to 300 (about 1% of the total training set) in the experiments. Fig. 9 plots the experiment result. The increased eliminated data size leads to a slight reduction in FR for Forsaken. Besides, from the changes of the target model's accuracy on the training and testing sets, we can observe that the memorization elimination hardly affects the performance of the target model, even the eliminated size reaches 300. The phenomenon shows that Forsaken is robust to the modest change of eliminated data size.



(a) The change of FR with different elimination sizes for MNIST.



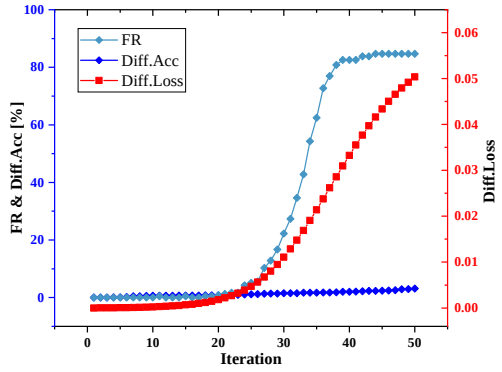
(b) The change of FR with different elimination sizes for News.

Fig. 9. The performance change of Forsaken with different elimination sizes. Diff.Train.Acc and Diff.Test.Acc mean the difference of the training/testing accuracy before and after conducting Forsaken, respectively.

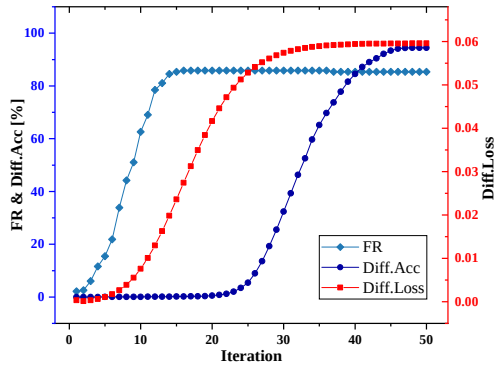
Training Iteration of Generator. Since the dummy gradient generator is implemented by the gradient based optimizer, its performance is strongly influenced by the training iteration. Fig. 10 shows the performance change of memorization elimination with increasing iterations. Commonly, the dummy gradient generator reaches its best performance after tens of iterations and ends its iteration. In the experiments, we force the generator to additionally train for some extra iterations to better illustrate its performance. Once the best point is reached, more training iterations hardly reduce FR but cause more and more accuracy loss of the target model. The reason is that according to our design of the generator, overtraining cannot cancel the eliminated memorization; however, it can cause over elimination of unrelated memorization, which lowers the accuracy of the target model. In practical,

Moreover, the efficiency of memorization elimination is also significant merit to evaluate Forsaken. Concretely, the efficiency of Forsaken is mainly affected by two factors, the training iterations of the generator and the size of the dummy gradient required to be simulated. Table IV reflects the running time to accomplish one user's memorization elimination with different iteration numbers and different sizes of neural networks. The experiments are performed with a laptop, equipped with Intel Core i7-7200 CPU @2.50Ghz and 8GB RAM (no GPU acceleration). The experiment results show that the dummy gradient generation process can be completed in minutes, even for the neural network with 14.09M parameters.

Overtraining. Referring to the former research [8], overtraining is always tightly related to the machine learning memorization. Loosely speaking, overtraining brings deeper and stronger memorization of the trained model towards the training set and leads to worse performance on the unknown data set. Fig. 11 shows a typical example of overtraining,



(a) The FR at each iteration of the dummy gradient generator for MNIST.



(b) The FR at each iteration of the dummy gradient generator for News.

Fig. 10. The FR of Forsaken at each iteration of the dummy gradient generator.

TABLE IV
EFFICIENCY OF MEMORIZATION ELIMINATION

Iteration	Running Time (s)			
	MNIST	News	CIFAR-10	CIFAR-100
10	3.88	14.78	59.89	56.01
15	6.13	22.34	86.97	88.81
20	8.82	29.65	117.51	124.14
25	11.93	36.99	152.71	164.82
30	15.64	44.43	182.48	209.97

which occurs by training the target model with only half of the training set for News. At the first few epochs, the testing loss drops rapidly until reaching the best point. After the point, the trend of testing loss is reversed, i.e., beginning to increase, which means the model is overtrained. To evaluate Forsaken versus overtraining, we record the memorization elimination results at the different stages of model training, including before and after overtraining. Not surprisingly, the memorization is successfully eliminated no matter overtraining occurs or not. Nonetheless, for the overtrained model, Forsaken has a better performance on memorization elimination. The reason is that the overtrained model usually has more redundant memorization on the training set, which leaves more space for Forsaken to remodel the model to operate memorization elimination.

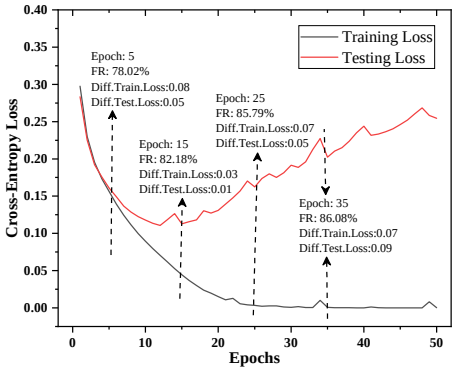


Fig. 11. The performance of Forsaken versus overtraining

D. Memorization Elimination for Language Model

For language model, Carlini *et al.* [8] introduces an efficient way to measure the degree of the memorization for a given sequence $s[r]$, which can be mathematically expressed as the following equation.

$$exposure_\theta(s[r]) = -\log_2 \int_0^{P_\theta(s[r])} \rho(x)dx, \tag{7}$$

where $exposure_\theta(s[r])$ is the memorization degree; $P_\theta(s[r])$ is the log-perplexity of $s[r]$ under the machine learning model f_θ; $\rho(\cdot)$ is a skew-normal function with mean μ, standard derivation σ^2 and skew α. Log-perplexity is a common indicator to evaluate how "surpriese" the language model is to see a given sequence, which can computed according to Eq. 8.

$$P_\theta(s[r]) = \sum_{i=1}^{n}(-\log_2 Pr(x_i|f_\theta(x_1...x_{i-1}))). \tag{8}$$

Notably, $exposure$ is a specially defined indicator that can only be applied to the sequence model, e.g, language model. As stated in [8], the risk of a given sequence to be extracted by an adversary is positively related to $exposure$.

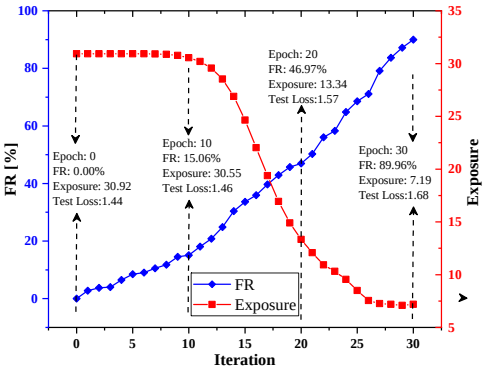


Fig. 12. Comparing FR and test loss to $exposure$ across different training iterations of the dummy gradient generator.

To further evaluate the effectiveness of Forsaken, we use the Penn TreeBank (PTB) dataset[5] to train a language model

[5]https://github.com/tomsercu/lstm/tree/master/dat

with a two-layer LSTM [32] that has 512 hidden units. During the training process, we insert 200 canaries[6] into ten different users' local set. The 200 canaries are treated as the eliminated set. After 100 epochs of training, we conduct memorization elimination. Fig. 12 plots the experiment result, where $exposure$ is the averaged value of the 200 canaries. It can be discovered that $exposure$ deceases to about 1.2 along with the training of our dummy gradient generator. Carlini's experiment [8] points out that the success probability for the adversary to extract a given sequence is negligible when $exposure$ is less than 10. Therefore, the above experiment shows that Forsaken can significantly lower the probability of the adversary to extract the user's private from the unintended memorization. Meanwhile, from the test loss change (increase about 10%), we can conclude that the performance loss caused by the memorization elimination is totally acceptable.

VII. Related Work

A significant amount of related work about machine learning security inspires our work in this paper.

Federated learning. Forsaken is basically designed for the emerging federated learning technique, which is first proposed by Google for privacy-preserving machine learning model training in the mobile crowdsensing scenario [1]. Federated learning significantly improves the security level of traditional distributed learning by raising the attack object from data to gradient. Nevertheless, Bagdasaryan *et al.*pointed out that the gradient was not as secure as Google declared, and vulnerable to the adversarial example attack [33]. Therefore, the followup work of federated learning put a tremendous amount of effort into designing protecting the gradient privacy through different cryptographic tools, e.g., secret sharing [10], differential privacy [9], and homomorphic encryption [11]. The core of all the methods is that they significantly increase the attack hardness of the attacker by using the secure gradient aggregation method to implement the gradient model update [29], which can also be used in Forsaken.

Although massive researches have been done on federated learning, there is still an unresolved practical problem, which is that none of them consider how to let a user securely quit from a learning federation. After all, people are more and more realizing the importance of individual privacy, and the GDPR [7] released by the European Union has ruled that a natural person should have the right to choose to remain or withdraw its private data without special statement.

Membership Inference. The most important evaluation tool used in Forsaken is the membership inference algorithm, i.e., membership oracle. The research of membership inference is first inspired by the membership privacy problem while deploying machine learning as a service [14]. Given a trained model $f(\cdot)$, a training set D and an arbitrary sample x, membership inference answers the question whether x is a member of D based on the model output $f(x)$. Further, it makes it possible to determine whether the memorization of $f(\cdot)$ about the pattern of x is directly trained by x, an individual person's private information in federated learning, or derived from other similar data.

The early membership inference towards machine learning used multiple shadows models to train the membership oracle [14], which had to cost considerable computation resources. Later, the experiments of Salem *et al.* [12] showed that even with only one shadow model, the membership oracle could still work. Considering the efficiency and practicality under the federated learning scenario, we do not add the membership inference performance as a part of the optimization objective for the dummy gradient generator. However, the principle of membership inference inspires us a lot to define our concept of memorization elimination for federated learning.

Memorization of Machine Learning. Few of the present works focus on the security of the memorization in machine learning. Song *et al.* [34] proposed several encoding methods that made the model to secretly "memorize" the training data in the training process. Correspondingly, the adversary could extract the memorized data from the trained model based on the specially designed decoding method, In fact, the memorization mentioned in [34] is intentionally backdoored by the adversary, and to obtain the memorization, the training procedure (i.e., the normal loss function) has to be changed. Carlini *et al.* [8] studied the unintentional memorization phenomenon in the language model and gave an excellent method to measure its exposure level. Besides the applicable range, the critical difference of the above two works and ours is that they focus on how to extract or measure the memorization of machine learning, but ours tries to eliminate the specific memorization.

VIII. Conclusion

To date, most of the researches about federated learning concentrated on how to efficiently and securely memorize the training set. However, there was still a lack of research on how to help the user eliminate unexpected memorization. In this paper, we emphatically discussed this problem and proposed Forsaken, a new framework of federated learning that provided the memorization elimination service for the user. To implement Forsaken, we first presented a quantified indicator, called forgetting rate, to measure the performance of memorization elimination. Then, inspired by the memorization management mechanism of the human body, we proposed a "learn to forget" method to achieve memorization elimination for machine learning. In the method, the user could stimulate the neurons of a machine learning model to eliminate the memorization of the specific data by training a dummy gradient generator. In particular, the dummy gradient could be treated according to the common procedure of federated learning, which indirectly ensured the security of Forsaken.

Although a novel memorization elimination method was proposed, there were still several areas where our work was limited in scope: 1) Forsaken was designed to fit the classification task of machine learning and could not directly handle

[6]Canary is a kind of specially generated sequence defined in [8].

other types of tasks, e.g., regression and clustering. 2) Although only several epochs were required to train the dummy gradient generator, the computation overhead of memorization elimination was still too high for some energy-limited devices when the size of the neural network parameters was large.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[3] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 747–14 756.

[4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[6] E. L. Harding, J. J. Vanto, R. Clark, L. Hannah Ji, and S. C. Ainsworth, "Understanding the scope and impact of the california consumer privacy act of 2018," *Journal of Data Protection & Privacy*, vol. 2, no. 3, pp. 234–253, 2019.

[7] G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016.

[8] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 267–284.

[9] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

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

[11] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, and H. Ludwig, "Hybridalpha: An efficient approach for privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 13–23.

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

[13] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning: Revisited and enhanced," in *International Conference on Applications and Techniques in Information Security*. Springer, 2017, pp. 100–110.

[14] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 3–18.

[15] M. Nasr, R. Shokri, and A. Houmansadr, "Machine learning with membership privacy using adversarial regularization," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 634–646.

[16] S. Awan, F. Li, B. Luo, and M. Liu, "Poster: A reliable and accountable privacy-preserving federated learning framework using the blockchain," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2561–2563.

[17] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[18] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1387–1395.

[19] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530*, 2016.

[20] D. Gao, Y. Liu, A. Huang, C. Ju, H. Yu, and Q. Yang, "Privacy-preserving heterogeneous federated transfer learning," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 2552–2559.

[21] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *European Symposium on Research in Computer Security*. Springer, 2008, pp. 192–206.

[22] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[23] R. L. Davis and Y. Zhong, "The biology of forgettinga perspective," *Neuron*, vol. 95, no. 3, pp. 490–503, 2017.

[24] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[25] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–2.

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

[27] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[28] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2019, pp. 233–239.

[29] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[30] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[33] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," *arXiv preprint arXiv:1807.00459*, 2018.

[34] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 587–601.