
Membership Inference with Privately Augmented Data Endorses the Benign while Suppresses the Adversary

Da Yu *

Sun Yat-sen University

yuda3@mail2.sysu.edu.cn

Huishuai Zhang

Microsoft Research Asia

huishuai.zhang@microsoft.com

Wei Chen

Microsoft Research Asia

wche@microsoft.com

Jian Yin

Sun Yat-sen University

issjyin@mail.sysu.edu.cn

Tie-Yan Liu

Microsoft Research Asia

tie-yan.liu@microsoft.com

Abstract

Membership inference (MI) in machine learning decides whether a given example is in target model’s training set. It can be used in two ways: adversaries use it to steal private membership information while legitimate users can use it to verify whether their data has been forgotten by a trained model. Therefore, MI is a double-edged sword to privacy preserving machine learning. In this paper, we propose using private augmented data to sharpen its good side while passivate its bad side. To sharpen the good side, we exploit the data augmentation used in training to boost the accuracy of membership inference. Specifically, we compose a set of augmented instances for each sample and then the membership inference is formulated as a set classification problem, i.e., classifying a set of augmented data points instead of one point. We design permutation invariant features based on the losses of augmented instances. Our approach significantly improves the MI accuracy over existing algorithms. To passivate the bad side, we apply different data augmentation methods to each legitimate user and keep the augmented data as secret. We show that the malicious adversaries cannot benefit from our algorithms if being ignorant of the augmented data used in training. Extensive experiments demonstrate the superior efficacy of our algorithms. Our source code is available at anonymous GitHub page https://github.com/AnonymousDLMA/MI_with_DA.

1 Introduction

Membership inference (MI) against machine learning models has been extensively studied from the adversary perspective [21, 26, 19, 16, 15, 12, 23]. An adversary applies membership inference algorithms to steal privacy-sensitive membership information from target model’s training data [4, 17]. One example is that participation in a disease-specific dataset indicates the diagnosis of such disease [4]. Hence from the view of legitimate users and machine learning service providers, it is desirable to suppress such attack as much as possible, i.e., diminishing the MI success rate of the adversary.

On the other side, we advocate that membership inference can enable the legitimate users to better control their data, e.g., to infer whether a service provider uses their data to train a public model. In machine learning context, the influence of a user on trained model should be erased at request as the learned model leaks private information about its training data [6, 25, 21, 11, 29]. This aligns with

*The work was done when this author was an intern at Microsoft Research Asia.

the spirit of “*the right to be forgotten*” in the European Union’s General Data Protection Regulation (GDPR) [2] and the California Consumer Privacy Act in the United States [1]. In Section 4, we show that membership inference can be used to verify the compliance of *data deletion in machine learning* (also known as *machine unlearning*), which has attracted lots of attention over the years [7, 5, 9]. We argue that this ability benefits not only the users but also the machine learning service providers because it has been observed that applications with good privacy protection promote user adoption [3]. In this case, it is desirable to enhance the MI success rate for the legitimate users.

It seems that we cannot achieve these two goals simultaneously because of the contradictory objectives. In this paper, however, we propose leveraging private data augmentation as a solution. We first show one can utilize the data augmentation used in training to boost MI success rate. Then we show legitimate users can protect membership privacy by applying data augmentation secretly.

The core technique for boosting MI success rate is to utilize the set of augmented data points instead of one data point to do membership inference. Specifically, with a set of predictions of augmented data points, we formulate the membership inference as a set classification problem. We then design classifiers based on permutation invariant features: one using threshold on some basic statistics of losses, and the other training a neural network with the moments of losses. Our algorithms significantly improve the success rate over existing membership inference algorithms when the same augmented data points are used in training process. Moreover, without the knowledge about the augmented data used in training, our algorithms have no improvement over existing algorithms. Therefore, we argue that the adversary cannot take advantage of our algorithms if the augmented data is kept as secret by the legitimate user.

In this paper, we focus on the *black-box* membership inference [21, 26, 19, 23, 18]. The black-box setting naturally arises in the *machine learning as a service (MLaaS)* system. In MLaaS, a service provider trains a ML model on private crowdsourced data and releases the model to users through prediction API. Under the black-box setting, one has access to the model’s output of a given example. Typical outputs are the loss value [26, 18] and the predicted logits of all classes [21, 19]. In this paper, we use the loss value of a given example as suggested in [18].

Our contributions can be summarized as follows. We design new membership inference algorithms against machine learning models and achieve significantly higher inference accuracy than existing algorithms. We also demonstrate that the adversary can not take advantage of our algorithms if the augmented data used in training is kept as secret. Furthermore, we compose a set of membership inference results over a given dataset to verify the compliance of deleting the given dataset from trained model. To the best of our knowledge, this is the first work to explore such usage of membership inference. Our verification process is easy to implement and achieves high confidence based on our strong MI routine.

1.1 Related work

Existing implementations of membership inference include those based on neural network and those based on simple metrics. Shokri et al. [21] and Salem et al. [19] train a neural network to infer the membership of a given example with multiple *shadow models* generating training data. Song et al. [23] use a threshold on the prediction confidence to determine if an example is a member or not. Yeom et al. [26] and Sablayrolles et al. [18] classify one sample in the training set if the loss of the sample is lower than a predefined threshold. All existing attacks only use the target model’s output on single example even when data augmentation is used.

In machine learning context, “the right to be forgotten” requires to remove the influence of given examples from a trained model [7, 5, 9]. Ginart et al. [7] construct quantized and divide-and-conquer k -means clustering algorithms to avoid retraining the model from scratch. Bourtoule et al. [5] divide the dataset into multiple shards, train sub models on each shard, and only retrain the sub model that is affected by deleting request. Guo et al. [9] focus on linear models, where they use second order update to cancel out target example’s influence and then add noise to achieve certified removal.

Apart from designing provable algorithm that can perform data deletion from a trained model, it is also considered how to verify the compliance of requested deletion. Sommer et al. [22] use backdoor attack [14, 8] to verify whether a server deletes given data faithfully. They force the model to memorize the specific trigger of each user, which requires quite a number of intentionally poisoned examples to succeed. This decreases the model’s test performance. In contrast, we use membership

inference as basic oracle, which neither requires individual user has large dataset nor hurts the target model’s test performance.

1.2 Paper organization

The rest of this paper is organized as follows. In Section 2, we introduce some background knowledge. We present two new membership inference algorithms in Section 3 and show that the malicious adversary cannot take advantage of them. Section 4 introduces how to use membership inference to verify the compliance of machine unlearning. Section 5 presents experiment results on different models and datasets. Finally, we conclude in Section 6.

2 Preliminary

We use $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ to denote a dataset with n pairs of feature and label. The dataset may be constructed from a set of users $U = \{u_1, \dots, u_k\}$. The crowdsourced dataset $D = \cup D_i$ is the union of all users’ datasets. A trained model $f(\mathbf{x}; \theta) : \mathcal{X} \rightarrow \mathcal{Y}$ with parameters θ is a mapping from the feature space to the label space. Given model’s prediction and target label, the loss function l computes the loss value, e.g. the cross-entropy loss for classification problem. We use $f(\mathbf{x})$ to denote $f(\mathbf{x}; \theta)$ if without further explanation.

Data augmentation generates similar variants for each example to enlarge the training set. We use \mathcal{T} to denote the set of all possible transformations. For given (\mathbf{x}, y) , each transformation function $t \in \mathcal{T}$ generates one augmented example $t(\mathbf{x}, y)$. For example, if data is image, each t could be the combination of rotation and random cropping. The set \mathcal{T} then contains the transformations with all possible rotation degrees and cropping locations. The size of \mathcal{T} may be unlimited and we are only able to use a subset in practice. We use $T \subset \mathcal{T}$ to denote a subset with N transformation instances. We adjust N to control the strength of data augmentation.

Let M be a membership inference algorithm against a trained model f . The output of M is a boolean value denotes whether a given example is in the training set of f . For example, the algorithm based on loss threshold [26, 18] can be formulated as

$$M_{loss}(\mathbf{x}, y, f) = \text{True} \quad \text{if} \quad l(f(\mathbf{x}), y) < \tau_{loss},$$

where τ_{loss} is a predefined scalar threshold. Song et al. [23] use prediction confidence to infer the membership. They classify an example as member if the output confidence is higher than a predefined scalar threshold:

$$M_{conf}(\mathbf{x}, y, f) = \text{True} \quad \text{if} \quad f(\mathbf{x})_y > \tau_{prob}.$$

3 Membership Inference with Data Augmentation

In this section, we first show why existing algorithms perform badly when the target model is trained with data augmentation. Then we present our new MI algorithms which can exploit the information of a set of augmented instances rather than a single example. Finally, we show the adversary cannot take advantage of the proposed algorithms if the augmentation used in training is kept secret to the adversary.

3.1 On the limitation of existing MI algorithms

The success of existing MI algorithms largely depends on the generalization gap of target model. Generalization gap measures how differently the target model perform on the training set and the test set. Therefore, MI is easy to do if the generalization gap is large. For example, Shokri et al. [21] achieve inference success rate higher than 70% against a model trained on CIFAR10 [13] dataset while the gap between target model’s training and test accuracy is nearly 40%. They show the inference accuracy drops quickly as the target model’s generalization gap diminishes (Table 2 in [21]).

It has been observed that data augmentation, a powerful weapon to combat overfitting, can significantly reduces the success rate of existing MI algorithms [18]. In order to clearly illustrate the influence of data augmentation, we plot the distribution of losses in Figure 1. We use the ResNet110

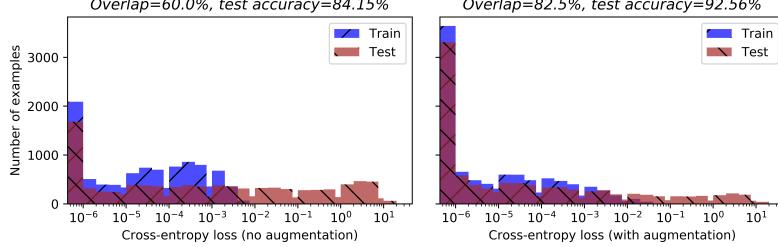


Figure 1: Loss distribution from models trained with/without data augmentation on CIFAR10 dataset. The model is ResNet110. The plot uses 10000 examples from training set and 10000 examples from test set. The overlap area (dark region) between training and test distributions is significantly larger when data augmentation is used.

model [10] to fit CIFAR10 dataset [13]. We use the same transformation pool \mathcal{T} as the one in [10] which contains horizontal flipping and random cropping. The size of T is $N = 10$. As shown in Figure 1, the overlap area between losses of training examples and losses of test examples is much larger when data augmentation is used. For the loss inside the overlap area, it is impossible to classify its membership confidently. The overlap area sets up a ceiling on the membership inference success rates for the MI algorithms which use single loss value as feature.

3.2 New membership inference algorithms

As the existing MI algorithms are inherently limited by the large overlap area in Figure 1, we propose leveraging more information from data augmentation to boost the MI success rate. With data augmentation, the model is trained with a set of augmented data points $T(\mathbf{x}, y) = \{(t(\mathbf{x}), y); t \in T\}$. Consequently, we have a set of outputs based on $T(\mathbf{x}, y)$ and a set of losses $L_{T,f}(\mathbf{x}, y) = \{l_f(\tilde{\mathbf{x}}, y); (\tilde{\mathbf{x}}, y) \in T(\mathbf{x}, y)\}$. We hide T and f for readability when there is no ambiguity. Instead of using single loss as in [18], we use $L(\mathbf{x}, y)$ to do the membership inference. The inference task is then to classify $L(\mathbf{x}, y)$ into either training or test set. For each $(\mathbf{x}, y) \in D$, $L(\mathbf{x}, y)$ is a valid empirical distribution. We plot the distributions of the basic statistics of $L(\mathbf{x}, y)$ in Figure 2. The overlap area in Figure 2 is smaller compared to Figure 1 because $L(\mathbf{x}, y)$ contains more information than single loss value.

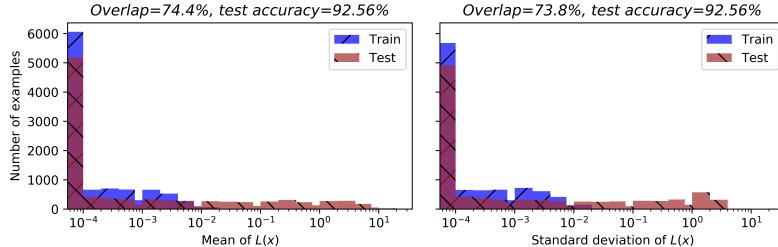


Figure 2: Distributions of mean and standard deviation of $L(\mathbf{x}, y)$. The experiment setting is the same as Figure 1. When using mean or standard deviation of $L(\mathbf{x}, y)$ as metric, the overlap area between training and test distributions is smaller than using single loss.

Smaller overlap area indicates that it is easier to distinguish examples from training and test sets. Therefore, the results in Figure 2 suggest that using the basic statistics of $L(\mathbf{x}, y)$ is better than using single loss. Motivated by this, we first explore using the mean or standard deviation of $L(\mathbf{x}, y)$ as input feature for set classification.

Following previous threshold based algorithms [18, 23], the threshold τ in Algorithm 1 can either be tuned as hyperparameter or set based on the outputs of shadow models. From Table 1, we can see that Algorithm 1 outperforms existing algorithms by a large margin.

As mean and standard deviation are only coarse information of $L(\mathbf{x}, y)$, it is natural to ask if we can design features incorporating the most information of $L(\mathbf{x}, y)$. A straightforward solution is to

Algorithm 1: Membership inference based on basic statistics of $L(\mathbf{x}, y)$ (M_{mean} , M_{std}).

Input :losses of target example $L(\mathbf{x}, y)$; threshold τ ; metric function m computes mean or standard deviation.

Output :boolean value, *true* denotes (\mathbf{x}, y) is a member.

1 Compute $v = m(L(\mathbf{x}, y))$.

2 Return $v < \tau$.

train a neural network which takes entire $L(\mathbf{x}, y)$ as input and outputs final decision. However, such straightforward solution performs badly, whose experiment results are relegated to Appendix. One reason is that using losses as input is not invariant to permutation on losses . As a set classification problem, the order of elements in $L(\mathbf{x}, y)$ should not change the final decision [28]. Using raw losses as input does not possess this property because permuting the input losses will change the outputs of neurons. This makes the inference model hard to distinguish what is useful feature for classification and what is the noise induced by the different positions of losses.

In order to design features which are invariant to permutation on losses, we use the raw moments of $L(\mathbf{x}, y)$. The i_{th} raw moment v_i of a probability density (mass) function $p(z)$ can be computed as

$$v_i = \int_{-\infty}^{+\infty} z^i p(z) dz.$$

The moments of $L(\mathbf{x}, y)$ can be computed easily because it is a valid empirical distribution with uniform probability mass. For probability distributions in bounded intervals, the moments of all orders uniquely determines the distribution (known as *Hausdorff moment problem* [20]). More importantly, shuffling the elements in $L(\mathbf{x}, y)$ would not change the resulting moments. In Algorithm 2, we use the raw moments as features and train a neural network to infer the membership.

Algorithm 2: Membership inference based on moments of $L(\mathbf{x}, y)$ ($M_{moments}$).

Input :losses of target example $L_{target}(\mathbf{x}, y)$; losses from training and test sets

$\mathcal{L}_{train} = \{L(\mathbf{x}, y); (\mathbf{x}, y) \in D_{train}\}$, $\mathcal{L}_{test} = \{L(\mathbf{x}, y); (\mathbf{x}, y) \notin D_{train}\}$; maximum moments order m ; specification of inference network S .

Output :boolean value, *true* denotes (\mathbf{x}, y) is a member.

1 **for** $L(\mathbf{x}, y) \in \mathcal{L}_{train} \cup \mathcal{L}_{test}$ **do**

2 **for** $i \in [m]$ **do**

3 Compute the i_{th} raw moment of $L(\mathbf{x}, y)$: $v_i = \frac{1}{|T|} \sum_{l \in L(\mathbf{x}, y)} l^i$.

4 Normalize $v_i = v_i^{\frac{1}{i}}$

5 **end**

6 Create tuple $(\mathbf{v} = \{v_i; i \in [m]\}, in)$, where $in = \mathbb{1}_{L(\mathbf{x}, y) \in \mathcal{L}_{train}}$.

7 **end**

8 Use created tuples and specification S to train the inference model N .

9 Return $N(L_{target}(\mathbf{x}, y))$

The training data of Algorithm 2 can be collected by using shadow models [21] or by assuming the prior knowledge on part of the target model’s training data [16]. We find that the training of inference network in Algorithm 2 only needs several hundreds of training points, which is much fewer than previous algorithms based on neural net, e.g. Shokri et al. [21] need thousands of training points to train the inference model. This is because the moments of losses are features easier to fit than the output logits in [21].

To demonstrate the effectiveness of Algorithm 1 and 2, we evaluate them on both the small convolution model in the membership inference literature [21, 18] and standard ResNet [10]. We benchmark our algorithms with M_{loss} and M_{conf} . For baselines, we report the best result among using each $l \in L(\mathbf{x}, y)$ and the loss of original image. We do not use the loss of original image for our algorithms. We use the benchmark dataset CIFAR10, which has 10 classes of real-world objects. We use 6 common operations to construct \mathcal{T} , including rotation, translations, random erasing, etc. The details are introduced in Appendix. We sample a subset $T \subset \mathcal{T}$ with $N = 10$. Other implementation

Table 1: Membership inference success rates (in %) on CIFAR10 dataset. The number under membership inference algorithm is the inference accuracy. We use bold font to denote the best inference accuracy. The baseline attacks M_{loss} and M_{conf} are introduced in Section 2. The row with $N = 0$ denotes model trained without data augmentation. Test acc. denotes the target model’s classification accuracy on test set.

Model	$ T $	Test acc.	M_{loss}	M_{conf}	M_{mean}	M_{std}	$M_{moments}$
2-layer ConvNet	$N = 0$	59.7	83.7	83.4	N/A	N/A	N/A
	$N = 3$	64.6	82.2	82.1	90.3	90.9	91.3
	$N = 10$	66.8	63.5	63.5	69.4	71.6	71.4
ResNet110	$N = 0$	84.9	65.4	65.3	N/A	N/A	N/A
	$N = 3$	89.4	63.2	63.2	68.5	68.7	71.4
	$N = 10$	92.7	59.3	58.7	66.3	66.9	67.1

Table 2: Membership inference success rates (in %) with partial or no knowledge on augmented data used in training. We train the ResNet110 model on CIFAR10 dataset with $N = 10$. The numbers (N') in first row denote the number of transformations in T the adversary has access to. When $N' = 0$, we random sample $T' \in \mathcal{T}$ with $|T'| = 10$ to evaluate our algorithms.

N'	10	8	6	4	2	0
$M_{moments}$	67.1	66.5	65.4	63.5	61.6	55.7
M_{std}	66.9	66.0	65.8	64.3	61.4	54.5

details are the same as those used in Section 5. The results are presented Table 1. For a given strength of augmentation, Algorithm 1 and 2 outperform existing methods significantly. Algorithm 2 has the best performance in general because it utilizes the most information of $L(\mathbf{x}, y)$. Surprisingly, our algorithms on models trained with data augmentation sometimes perform even better than the baseline algorithms on models trained without data augmentation. More experiments with varying N are presented in Section 5.

3.3 Using secret transformations to protect membership privacy

Algorithm 1 and 2 give legitimate users new and better ways to verify their influence in trained models. Then the question left is how to prevent the adversary takes advantage of our algorithms to to steal private membership information.

We propose using private data augmentation to suppress the adversary without hurting the inference accuracy of legitimate users. When $T(\mathbf{x}, y)$ is in training set, the elements in $L(\mathbf{x}, y)$ are small because deep neural net can memorize $T(\mathbf{x}, y)$. Otherwise, the elements in $L(\mathbf{x}, y)$ are usually large. This contradistinction makes MI a easier task. Therefore, whether $T(\mathbf{x}, y)$ is used in training matters. Legitimate users can utilize this phenomenon to make our attacks only available to themselves. Specifically, instead of choosing the same T for all users, u_i can construct T_i privately. The malicious attacker can not acquire useful knowledge of T_i by random guessing as long as the used \mathcal{T} is large enough. We record the used transformation set for each user and use it to evaluate our algorithms after training. This process is similar to set a password. Without the “password” (T_i), the malicious attacker can only use the original image or randomly chosen transformations which yields significantly lower inference success rate. In Table 2, we show the inference success rates when the adversary only has partial or no knowledge on the transformation set used in training. The inference success rate drops quickly as the adversary’s knowledge on T_i diminishes. We illustrate the above process in Figure 3.

4 Use Membership Inference to Verify Machine Unlearning

In this section, we show membership inference can be used to verify the compliance of machine learning. We composite the results of membership inference on individual examples in a given set to infer whether the influence of such set is removed form target model. We use D_i and n_i to denote the target dataset and its size, respectively. For every (x, y) in D_i , $M(x, y)$ returns 1 if it predicts

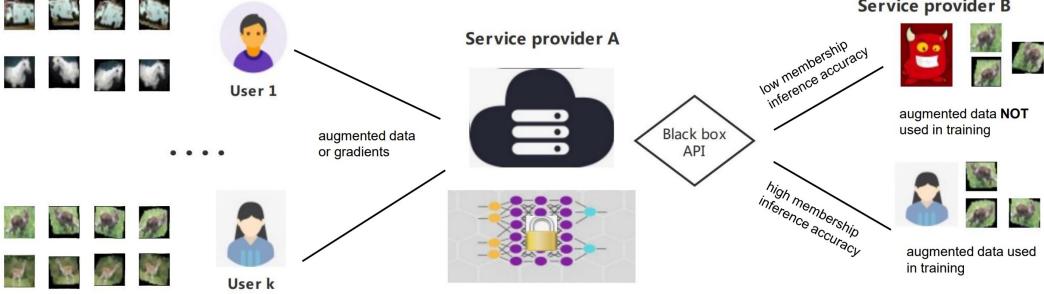


Figure 3: Illustration of using private data augmentation to protect membership privacy. User i first chooses T_i secretly. Then users train a model jointly by sharing the augmented examples or the gradients computed on augmented data to service provider A. Service provider A releases the trained model through black-box API. If service provider B also has u_i as its user, it can use the data to probe the membership of u_i in the published model. However, the malicious attacker has no access to T_i and therefore has low inference success rate.

the example is used in training, otherwise it returns 0. Let $c = \sum_{(x,y) \in D_i} M(x, y)$ be the number of total positive predictions. Large c indicates that the user's data is not deleted. Following Sommer et al. [22], we formulate a hypothesis testing problem. Specifically, we define the *null hypothesis* as $H_0: D_i$ is not in the training set of given model (server removes the influence of D_i from model) and *alternative hypothesis* as $H_1: D_i$ is in the training set. There are two types of errors we might make. The Type I error rate α (false positive) and Type II error rate β (false negative) are

$$\alpha = Pr[\text{Accepts } H_1 | H_0], \quad \beta = Pr[\text{Accepts } H_0 | H_1].$$

We accept H_0 if c is smaller than a given threshold s otherwise we accept H_1 . Let $p = Pr[M(x, y) = 1 | H_0]$ be the probability that the inference oracle correctly predicts (x, y) not a member of training set. Let $q = Pr[M(x, y) = 1 | H_1]$ be the probability that the MI oracle correctly predicts (x, y) a member. We assume the probabilities are the same and independent for every (x, y) in D_i . For a given threshold s , the probability that we make the Type I error is

$$\alpha = Pr \left[\sum_{(x,y) \in D_i} M(x, y) \geq s | H_0 \right] = \sum_{j=s}^{n_i} \binom{n_i}{j} (1-p)^j p^{(n_i-j)},$$

which means that H_0 is true but we rejects it because c is large. Analogously, the probability of making Type II error is

$$\beta = Pr \left[\sum_{(x,y) \in D_i} M(x, y) < s | H_1 \right] = \sum_{j=0}^{s-1} \binom{n_i}{j} q^j (1-q)^{(n_i-j)}.$$

The probability p can be estimated by the inference success rate on testing set and the probability q can be estimated by the inference success rate on training set. For a given tolerance of α , we choose c to minimize the type II error β . The experimental results are shown in Section 5.2.

5 More Experiments

In this section, we sample a random subset $T \in \mathcal{T}$ for every example. We first compare our algorithms with existing inference algorithms. Then we use membership inference to verify the compliance of machine unlearning.

5.1 Comparison with existing algorithms

We conduct more experiments on both CIFAR10 and CIFAR100 datasets. CIFAR100 has 100 classes of real world objectives and therefore it is harder than CIFAR10. We also evaluate our algorithms on the Wide ResNet model [27]. We use the WRN16-8 model, which has more than

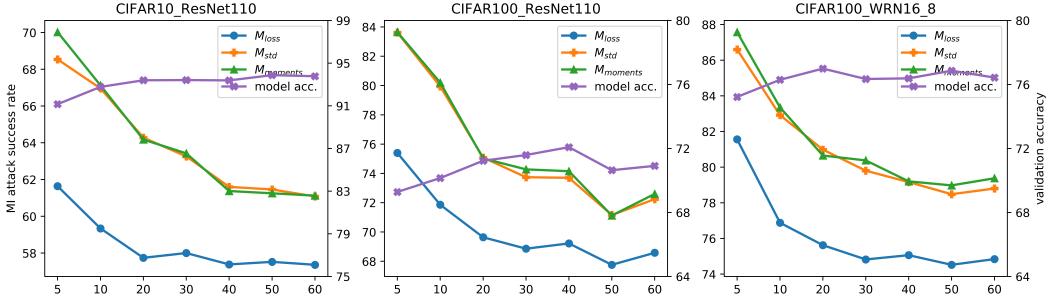


Figure 4: Membership inference success rates with varying N . The left y-axis denotes the membership inference attack success rate. The right y-axis denotes the test accuracy of target model. The advantage of our algorithms is clear on different datasets and models with varying choices of N .

Table 3: Machine unlearning verification confidence on CIFAR10 and CIFAR100 datasets. We use ResNet110 and WRN16-8 with $N = 10$. Notations: p and q denote the MI oracle’s prediction accuracy on test and training set, respectively, and n_i is the size of target dataset. We set Type I error $\alpha < 1 \times 10^{-3}$. Smaller Type II error β indicates higher confidence.

Data (Model)	Attack	MI accuracy	p	q	$\beta (n_i = 15)$	$\beta (n_i = 30)$
CIFAR10 (ResNet110)	M_{loss}	58.8%	0.203	0.973	1.0	1.0
	$M_{moments}$	67.1%	0.382	0.960	0.458	0.031
CIFAR100 (ResNet110)	M_{loss}	71.9%	0.482	0.956	0.139	1.7×10^{-4}
	$M_{moments}$	80.2%	0.653	0.951	5.1×10^{-4}	8.9×10^{-8}
CIFAR10 (WRN16-8)	M_{loss}	61.9%	0.253	0.986	1.0	0.35
	$M_{moments}$	70.1%	0.423	0.978	0.283	4.6×10^{-4}
CIFAR100 (WRN16-8)	M_{loss}	76.8%	0.560	0.977	4.5×10^{-3}	2.9×10^{-7}
	$M_{moments}$	83.4%	0.688	0.979	1.0×10^{-5}	4.4×10^{-13}

10 million parameters. The baseline algorithms are M_{loss} and M_{conf} . For baselines, we report the best result among using every element in $L(\mathbf{x}, y)$ and the loss of original image. For all threshold based algorithms, we tune the thresholds to separate the training and test sets optimally [18, 23]. For Algorithm 2, we use 200 examples from training set and 200 examples from test set to build the training data of inference network. The inference network has one hidden layer with 20 neurons and we compute moments up to 20 orders as input features. We use 2500 examples from the training set and 2500 examples from the test set to evaluate the inference success rates. The examples used to evaluate inference accuracy have no overlap with inference model’s training data. All models are trained on a single Tesla P40 GPU. More details such as training recipes and model configurations are available in Appendix. We plot the results with varying N in Figure 4. Our algorithms achieve high inference success rates against well-generalized models. Both Algorithm 1 and 2 outperform existing algorithms by a large margin on different models and datasets.

5.2 Machine unlearning verification

We use the trained models to verify the proposed verification method. We use the membership inference success rate on training set to estimate q and the success rate on test set to estimate p . We use n_i to denote the size of target dataset. For given n_i and the tolerance of Type I error α , we choose s to minimize β . The experiment results are shown in Table 3.

As shown in Table 3, our verification mechanism needs smaller n_i compared to the verification based on backdoor attack [22] while still achieves very high confidence. Moreover, our mechanism does not need to poison the data with wrong label and therefore causes no damage to target model’s test performance. The target model WRN16-8 in Table 3 achieves 94.6% test accuracy on CIFAR10 dataset. This suggests that membership inference is a powerful and effective way to verify the compliance of requested data deletion.

6 Conclusion

In this paper, we propose membership inference with private data augmentation, which benefits the legitimate data owners while suppresses the adversary. For a given example, we use the losses of its augmented examples to infer its membership. We present two ways: one using the mean and standard deviation of losses, and the other training a neural network with the moments of losses as permutation invariant features. The proposed algorithms achieve high membership inference accuracy against models with good generalization. Moreover, we show that the malicious adversaries can not take advantage of our algorithms if the benign users apply data augmentation privately.

Broader Impact

This work benefits the legitimate data owners with new ways to decide their contribution on a trained ML model without hurting membership privacy. This work also boosts the communication between the participants working on privacy preserving machine learning and the participants working on state-of-the-art ML applications by exploring accurate membership inference algorithms against models with high test accuracy. This work has no explicit ethical concerns and the proposed algorithms do not leverage biases in data.

References

- [1] California consumer privacy act. <https://oag.ca.gov/privacy/ccpa>.
- [2] European union’s general data protection regulation. <https://gdpr-info.eu/>.
- [3] Hannah Alsdurf, Yoshua Bengio, Tristan Deleu, Prateek Gupta, Daphne Ippolito, Richard Janda, Max Jarvie, Tyler Kolody, Sekoul Krastev, Tegan Maharaj, et al. Covi white paper. *arXiv preprint arXiv:2005.08502*, 2020.
- [4] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microRNA-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.
- [5] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. *arXiv preprint arXiv:1912.03817*, 2019.
- [6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, 2015.
- [7] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, 2019.
- [8] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 2019.
- [9] Chuan Guo, Tom Goldstein, Awini Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [11] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-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.
- [12] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.
- [13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- [14] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. *Network and Distributed Systems Security (NDSS) Symposium*, 2018.
- [15] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.
- [16] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.
- [17] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who’s there? membership inference on aggregate location data. *arXiv preprint arXiv:1708.06145*, 2017.

- [18] Alexandre Sablayrolles, Matthijs Douze, Yann Ollivier, Cordelia Schmid, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. *International Conference on Machine Learning*, 2019.
- [19] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. MI-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *Network and Distributed Systems Security (NDSS) Symposium*, 2019.
- [20] James Alexander Shohat and Jacob David Tamarkin. *The problem of moments*. American Mathematical Soc., 1943.
- [21] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (SP)*, 2017.
- [22] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards probabilistic verification of machine unlearning. *arXiv preprint arXiv:2003.04247*, 2020.
- [23] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.
- [24] Shakila Mahjabin Tonni, Farhad Farokhi, Dinusha Vatsalan, and Dali Kaafar. Data and model dependencies of membership inference attack. *Proceedings on Privacy Enhancing Technologies*, 2020.
- [25] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *IEEE Computer Security Foundations Symposium*, 2016.
- [26] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018.
- [27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- [28] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, 2017.
- [29] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, 2019.

Appendix A Supplementary materials for Section 3

A.1 Details of \mathcal{T}

We use 6 standard operations in image processing literature such as rotation, translation, shearing, etc. The details are listed below.

1. Flip the image horizontally with probability $p = 0.5$.
2. Padding 32 image into 40×40 and take a 32×32 crop at random location.
3. Rotate the image by $d \in [-15, 15]$ degrees.
4. Translate the image by $d \in [-6, 6]$ pixels.
5. Shear the image by $d \in [-15, 15]$ degrees.
6. Erase a 4×4 box at random location.

For each $t \in \mathcal{T}$, the operations are applied with random order and the parameters of each operation are also randomly chosen. We record the sequence of operations and parameters of each operation to save a chosen transformation. Then we use the saved transformations to evaluate MI algorithms after training.

A.2 Using raw outputs as features for inference model

We show that using raw outputs of target model to train inference model yields bad inference accuracy. We evaluate two approaches: one using the raw losses (M_{raw_loss}), and the other using the output logits of all augmented examples plus the ground truth label (M_{raw_logits}). For example, we concatenate the outputs of all augmented instances into a one dimension tensor. The pseudocodes of M_{raw_loss} and M_{raw_logits} are shown in Algorithm 3 and 4, respectively. The word ‘specification’ in pseudocode denotes the architecture and hyperparameters of target model. See Appendix B.1 for details.

Algorithm 3: Membership inference based on raw losses (M_{raw_loss}).

Input : losses of target example $L_{target}(\mathbf{x}, y)$; losses from training and test sets $\mathcal{L}_{train} = \{L(\mathbf{x}, y); (\mathbf{x}, y) \in D_{train}\}$, $\mathcal{L}_{test} = \{L(\mathbf{x}, y); (\mathbf{x}, y) \notin D_{train}\}$; specification of inference network S .
Output : boolean value, *true* denotes (\mathbf{x}, y) is a member.
1 **for** $L(\mathbf{x}, y) \in \mathcal{L}_{train} \cup \mathcal{L}_{test}$ **do**
2 Concatenate the elements in $L(\mathbf{x}, y)$ into vector \mathbf{v} .
3 Create tuple (\mathbf{v}, in) , where $in = \mathbb{1}_{L(\mathbf{x}, y) \in \mathcal{L}_{train}}$.
4 **end**
5 Use created tuples and specification S to train the inference model N .
6 Return $N(L_{target}(\mathbf{x}, y))$

We use 200 (2500) examples from training set and 200 (2500) examples from test set to build the training data for M_{raw_loss} (M_{raw_logits}). The configuration of inference model is the same as $M_{moments}$. The results are shown in Table 4. The results suggest that the raw outputs are less informative than moments of losses. This may due to the raw outputs are not invariant to permutation on the augmented instances.

Appendix B Supplementary materials for Section 5

B.1 Implementation details of experiments

The small model used in Section 3 contains two convolution layers with 64 kernels, a global pooling layer and a fully connected layer of size 128. Following [18, 23], we use 15000 examples as training set. The model is trained for 200 epochs with initial learning rate 0.01. We decay the learning rate by 10 at the 100-th epoch. The ResNet110 and WRN16-8 models are adopted from original

Dataset	Model	Test acc.	M_{raw_loss}	M_{raw_logits}	$M_{moments}$
CIFAR10	ResNet110	92.8	61.8	56.9	67.2
	WRN16-8	94.6	63.0	58.3	70.1
CIFAR100	ResNet110	69.9	74.1	51.7	80.2
	WRN16-8	76.1	77.2	52.9	83.4

Table 4: Membership inference success rate (in %) on CIFAR10/100 dataset with $|T| = N = 10$. The numbers in third column denote the target model’s top1 test accuracy. We use bold font to denote the best inference accuracy.

Algorithm 4: Membership inference based on raw logits (M_{raw_logits}).

Input : logits of target example $G_{target}(\mathbf{x}, y) = \{f(\tilde{\mathbf{x}}, y); (\tilde{\mathbf{x}}, y) \in T(\mathbf{x}, y)\}$; logits from training and test sets $\mathcal{G}_{train} = \{G(\mathbf{x}, y); (\mathbf{x}, y) \in D_{train}\}$, $\mathcal{G}_{test} = \{G(\mathbf{x}, y); (\mathbf{x}, y) \notin D_{train}\}$; specification of inference network S .

Output : boolean value, *true* denotes (\mathbf{x}, y) is a member.

- 1 **for** $G(\mathbf{x}, y) \in \mathcal{G}_{train} \cup \mathcal{G}_{test}$ **do**
- 2 Concatenate all elements in $G(\mathbf{x}, y)$ and label y into vector \mathbf{v} .
- 3 Create tuple (\mathbf{v}, in) , where $in = \mathbb{1}_{L(\mathbf{x}, y) \in \mathcal{L}_{train}}$.
- 4 **end**
- 5 Use created tuples and specification S to train the inference model N .
- 6 Return $N(G_{target}(\mathbf{x}, y))$

papers and we train the models with the same hyperparameters. For ResNet110 and WRN16-8, we use 50000 examples as training set and 10000 examples as test set.

We use 200 examples from training set and 200 examples from test set to build the training set for $M_{moments}$. We use moments up to 20 orders, which yields a 20-dimensional input. The inference model has one hidden layer with 20 neurons and we use Tanh non-linearity as activation function. We train the attack network for 1000 steps with learning rate 0.5. To compute the inference success rate, we use 2500 examples from the training set and 2500 examples from the test set.

B.2 More experiments

In Table 5, we compare our algorithms with two baseline algorithms M_{loss} and M_{conf} with $N = 10$. We apply different data augmentation to every example. The experiment settings are the same as those in Section 5. The results in Table 5 further justify the effectiveness of our algorithms. Interestingly, although the generalization gap of WRN16-8 is smaller than the gap of ResNet110, the algorithms achieve higher inference accuracy on WRN16-8. This phenomenon aligns with the finding in Tonni et al. [24] that the inference accuracy is affected not only by the generalization gap but also by the model and dataset in use.

Dataset	Model	Test acc.	M_{loss}	M_{conf}	M_{mean}	M_{std}	$M_{moments}$
CIFAR10	ResNet110	92.8	59.3	59.2	66.3	67.0	67.2
	WRN16-8	94.6	61.9	61.9	67.2	68.8	70.1
CIFAR100	ResNet110	69.9	71.9	71.8	79.2	79.9	80.2
	WRN16-8	76.1	76.9	76.8	81.9	82.9	83.4

Table 5: Membership inference success rate (in %) on CIFAR10/100 dataset with $N = 10$. The numbers in third column denote the target model’s top1 validation accuracy. The baseline algorithms M_{loss} and M_{conf} are introduced in Section 2. We use bold font to denote the best inference accuracy.