When Machine Unlearning Jeopardizes Privacy

Min Chen[1*] Zhikun Zhang[1*] Tianhao Wang[2,3†] Michael Backes[1]
Mathias Humbert[4] Yang Zhang[1]
[1]CISPA Helmholtz Center for Information Security [2]Carnegie Mellon University
[3]University of Virginia [4]University of Lausanne

ABSTRACT

The right to be forgotten states that a data owner has the right to erase their data from an entity storing it. In the context of machine learning (ML), the right to be forgotten requires an ML model owner to remove the data owner's data from the training set used to build the ML model, a process known as *machine unlearning*. While originally designed to protect the privacy of the data owner, we argue that machine unlearning may leave some imprint of the data in the ML model and thus create unintended privacy risks. In this paper, we perform the first study on investigating the unintended information leakage caused by machine unlearning. We propose a novel membership inference attack that leverages the different outputs of an ML model's two versions to infer whether a target sample is part of the training set of the original model but out of the training set of the corresponding unlearned model. Our experiments demonstrate that the proposed membership inference attack achieves strong performance. More importantly, we show that our attack in multiple cases outperforms the classical membership inference attack on the original ML model, which indicates that machine unlearning can have counterproductive effects on privacy. We notice that the privacy degradation is especially significant for well-generalized ML models where classical membership inference does not perform well. We further investigate four mechanisms to mitigate the newly discovered privacy risks and show that releasing the predicted label only, temperature scaling, and differential privacy are effective. We believe that our results can help improve privacy protection in practical implementations of machine unlearning. [1]

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Machine Unlearning; Membership Inference; Machine Learning Security and Privacy

ACM Reference Format:
Min Chen[1*] Zhikun Zhang[1*] Tianhao Wang[2,3†] Michael Backes[1], Mathias Humbert[4] Yang Zhang[1]. 2021. When Machine Unlearning Jeopardizes Privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484756

1 INTRODUCTION

The *right to be forgotten* entitles data owners the right to delete their data from an entity storing it. Recently enacted legislation, such as the General Data Protection Regulation (GDPR) [1] in the European Union, the California Consumer Privacy Act (CCPA) [2] in California, and the Personal Information Protection and Electronic Documents Act (PIPEDA) [3] in Canada, have legally solidified this right. Google Search has received nearly 3.2 million requests to delist certain URLs in search results over five years [8].

In the machine learning context, the right to be forgotten requires that, in addition to the data itself, any influence of the data on the model disappears [11, 71]. This process, also called *machine unlearning*, has gained momentum both in academia and industry [6, 10–12, 15, 21, 22, 24, 32, 42, 50, 65, 74]. The most legitimate way to implement machine unlearning is to remove the data sample requested to be deleted (referred to as target sample), and retrain the ML model from scratch, but this incurs high computational overhead. To mitigate this, several approximate approaches have been proposed [6, 10, 11, 32].

Machine unlearning naturally generates two versions of ML models, namely the *original model* and the *unlearned model*, and creates a discrepancy between them due to the target sample's deletion. While originally designed to protect the target sample's privacy, we argue that machine unlearning may leave some imprint of it, and thus create unintended privacy risks. Specifically, while the original model may not reveal much private information about the target sample, additional information might be leaked through the unlearned model.

Our Contributions. In this paper, we study to what extent data is indelibly imprinted in an ML model by quantifying the additional information leakage caused by machine unlearning. We concentrate on machine learning classification, the most common machine learning task, and assume both original and unlearned models to be black-box, the most challenging setting for an adversary.

We first propose a novel membership inference attack in the machine unlearning setting that aims at determining whether the target sample is part of the training set of the original model. Different from classical membership inference attacks [60, 64] which leverage the output (posteriors) of a single target model, our attack leverages outputs of both original and unlearned models. More

[*] Min and Zhikun contributed equally to the paper.
[†] Tianhao did most of the work while at Purdue University.
[1] Our code is available at https://github.com/MinChen00/UnlearningLeaks.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484756

concretely, we propose several aggregation methods to jointly use the two posteriors from the two models as our attack model's input, either by concatenating them or by computing their differences. Our empirical results show that the concatenation-based methods perform better in overfitted models, while the difference-based methods perform better in well-generalized models.

Second, in order to quantify the unintended privacy risks incurred by machine unlearning, we propose two novel privacy metrics, namely *Degradation Count* and *Degradation Rate*. Both of them quantify how much relative privacy the target has lost due to machine unlearning. Concretely, Degradation Count calculates the proportion of cases for which the adversary's confidence about the membership status of the target sample is larger with our attack than with classical membership inference attack. Degradation Rate calculates the average confidence increase between our attack and classical membership inference.

We conduct extensive experiments to evaluate the performance of our attack over a series of ML models, ranging from logistic regression to convolutional neural networks, with multiple categorical datasets and image datasets. The experimental results show that our attack consistently degrades the membership privacy of the target sample, which indicates machine unlearning can have counterproductive effects on privacy. In particular, we observe that privacy is especially degraded because of machine unlearning in the case of well-generalized models. For example, we observe that the classical membership inference attack has an accuracy (measured by AUC) close to 0.5, or random guessing, on the well-generalized decision tree classifier. On the contrary, the AUC of our attack is 0.89, and the Degradation Count and Degradation Rate are 0.85 and 0.28, respectively, which demonstrates that machine unlearning can have a detrimental effect on membership privacy even with well-generalized models. We further show that we can effectively infer membership information in more practical scenarios, including the scenario where there are multiple intermediate unlearned models, the scenario where a group of samples (instead of a single one) are deleted together from the original target model, and the online learning scenario where there are samples to be deleted and added simultaneously.

Finally, in order to mitigate the privacy risks stemming from machine unlearning, we propose four possible defense mechanisms: (1) publishing only the top-k confidence values of the posterior vector, (2) publishing only the predicted label, (3) temperature scaling, and (4) differential privacy. The experimental results show that our attack is robust to the top-k defense, even when the model owner only releases the top-1 confidence value. On the other hand, publishing only the predicted label, temperature scaling, and differential privacy can effectively prevent our attack.

To summarize, we show that machine unlearning degrades the privacy of the target sample in general. This discovery sheds light on the risks of implementing the right to be forgotten in the ML context. We believe that our attack and metrics can help develop more privacy-preserving machine unlearning approaches in the future. The main contributions of this paper are four-fold:

- We take the first step to quantify the unintended privacy risks in machine unlearning through the lens of membership inference attacks.
- We propose several practical approaches for aggregating the information returned by the two versions of the ML models.
- We propose two novel metrics to measure the privacy degradation stemming from machine unlearning and conduct extensive experiments to show the effectiveness of our attack.
- We propose four defense mechanisms to mitigate the privacy risks stemming from our attack and empirically evaluate their effectiveness.

Roadmap. In section 2, we introduce some background knowledge about machine learning and machine unlearning, and the threat model. section 3 presents the details of our proposed attack. We propose two privacy degradation metrics in section 4. We conduct extensive experiments to illustrate the effectiveness of the proposed attack in section 5 and section 6. In section 7, we introduce several possible defense mechanisms and empirically evaluate their effectiveness. We discuss the related work in section 8 and conclude the paper in section 9.

2 PRELIMINARIES

2.1 Machine Learning

In this paper, we focus on machine learning classification, the most common ML task. An ML classifier $\mathcal{M}$ maps a data sample x to posterior probabilities $\mathbb{P}$, where $\mathbb{P}$ is a vector of entries indicating the probability of x belonging to a specific class y according to the model $\mathcal{M}$. The sum of all values in $\mathbb{P}$ is 1 by definition. To construct an ML model, one needs to collect a set of data samples, referred to as the training set $\mathcal{D}$. The model is then built through a training process that aims at minimizing a predefined loss function with some optimization algorithms, such as stochastic gradient descent.

2.2 Machine Unlearning

Recent legislation such as GDPR and CCPA enact the "right to be forgotten", which allows individuals to request the deletion of their data by the model owner to preserve their privacy. In the context of machine learning, e.g., MLaaS, this implies that the model owner should remove the *target sample* x from its training set $\mathcal{D}$. Moreover, any influence of x on the model $\mathcal{M}$ should also be removed. This process is referred to as machine unlearning.

Retraining from Scratch. The most legitimate way to implement machine unlearning is to retrain the whole ML model from scratch. Formally, denoting the *original model* as $\mathcal{M}_o$ and its training dataset as $\mathcal{D}_o$, this approach consists of training a new model $\mathcal{M}_u$ on dataset $\mathcal{D}_u = \mathcal{D}_o \setminus x$.[2] We call this $\mathcal{M}_u$ the *unlearned model*.

Retraining from scratch is easy to implement. However, when the size of the original dataset $\mathcal{D}_o$ is large and the model is complex, the computational overhead of retraining is too large. To reduce the computational overhead, several approximate approaches have been proposed [6, 10, 11, 32].

SISA. SISA [10] works in an ensemble style, which is an efficient and general method to implement machine unlearning. The training dataset $\mathcal{D}_o$ in SISA is partitioned into k disjoint parts $\mathcal{D}_o^1, \mathcal{D}_o^2, \cdots, \mathcal{D}_o^k$. The model owner trains a set of original ML models $\mathcal{M}_o^1, \mathcal{M}_o^2, \cdots, \mathcal{M}_o^k$ on each corresponding dataset $\mathcal{D}_o^i$. When the model owner receives

[2] Note that we also study the removal of more than one sample in our experimental evaluation, but for simplicity we formalize our problem with one sample only.

a request to delete a data sample x, it just needs to retrain the sub-model $\mathcal{M}_o^i$ that contains x, resulting in unlearned model $\mathcal{M}_u^i$. Sub-models that do not contain x remain unchanged. Notice that the size of dataset $\mathcal{D}_o^i$ is much smaller than $\mathcal{D}_o$; thus, the computational overhead of SISA is much smaller than the "retraining from scratch" method.

At inference time, the model owner aggregates predictions from the different sub-models to provide an overall prediction. The most commonly used aggregation strategy is majority vote and posterior average. In our experiments, we use posterior average as aggregation strategy.

2.3 Threat Model

Adversary's Goal. Given a target sample x, an original model, and its unlearned model, the adversary aims to infer whether x is unlearned from the original model. In other words, the adversary aims to know that the target sample is in the training dataset of the original model but it is not in the training dataset of the unlearned model. While the goal of unlearning x is to protect x's privacy, a successful attack considered here can show unlearning instead jeopardizes x's privacy (especially when x's membership leakage risk is not severe on the original model before machine unlearning).

We focus on membership privacy as it is one of the most established method on quantifying privacy risks of ML models [64]. Knowing that a specific data sample x was used to train a particular model may lead to potential privacy breaches. For example, knowing that a certain patient's clinical records were used to train a model associated with a disease (e.g., to determine the appropriate drug dosage or to discover the genetic basis of the disease) can reveal that the patient carries the associated disease. Unlike classical membership inference, which only leverages the output of a target ML model, our adversary can exploit information of both original and unlearned models to perform their attack.

Other Cases. Besides the samples that are in the original model's training dataset but not in the unlearned model's training dataset, referred to as $\langle in, out\rangle$, there are other three cases, including $\langle out, out\rangle$, $\langle in, in\rangle$, $\langle out, in\rangle$. Samples in the $\langle out, out\rangle$ category are considered as the negative cases for the adversary to train their attack model (see subsection 3.2). For the $\langle in, in\rangle$ samples, their membership privacy can be quantified by a classical membership inference attack. The difference is that the attack model can leverage both original and unlearned models' information (see subsection 6.4). It is also possible that during the unlearning process, the unlearned model unlearns samples from the target model and updates itself with other new samples (referred to as incremental learning). In such cases, we have $\langle out, in\rangle$ samples. To infer the membership status of such samples, the adversary can similarly perform a classical membership inference attack on the unlearned model. Note that the privacy of $\langle out, in\rangle$ samples are not directly related to the privacy risks caused by machine unlearning, and the current literature on machine unlearning [10] also does not consider such cases. In subsection 6.3, we evaluate our attack when unlearning and updating are both performed on the target model simultaneously.

Adversary's Knowledge. We assume that the adversary has black-box access to an original ML model and its unlearned model. This is realistic as the target black-box model can be queried at any time, such as in the setting of MLaaS, and all of the query results can be stored locally by the adversary; this also follows the assumption of previous works [7, 59, 60, 64, 68]. As such, when there are no changes on the target sample's outputs from two consecutive queries, then the unlearning did not happen and the adversary does not need to launch the attack. On the other hand, when the adversary observes changes on the target sample's outputs, they know that the target model has been updated. In most parts of the paper, we consider the scenario where the original model and the unlearned model differ only by one sample. However, we further show that when the two models differ by multiple samples, the adversary can still mount their attack effectively (see section 6).

We also assume that the adversary has a local shadow dataset which can be used to train a set of shadow models to mimic the behavior of the target model. The shadow models are then used to generate training data for the attack model (see section 3 for more details). The shadow dataset can either come from the same distribution as the target dataset or from a different one. We evaluate both settings in section 5.

Difference with Updates-Leak [59]. There are recent studies aiming to quantify the information leakage in the model updating process. Salem et al. [59] show that in the online learning applications, where an ML image classifier is updated by new data samples, the adversary can reconstruct the updated samples by exploiting information from two versions of the target ML model (before and after the updating). Brockschmidt et al. [7] show similar results in the natural language models as well as the data deletion scenario. This line of work is related to our attack in the sense that we all study the unintended information leakage in model updating processes. However, the attack goals are different.

Our attack focuses on membership inference while Salem et al. [59] propose two attacks (in the single-sample setting), including single-sample reconstruction and single-sample label inference. The former (single-sample reconstruction) can only reconstruct a data sample that is "similar" to the updated sample; but it cannot further determine the membership status of the target sample by simply comparing the difference between the reconstructed sample and the target sample, and concluding the membership status with a predefined threshold. One reason is that it is unclear which metric is the best to measure two samples' similarity. Updates-Leak uses MSE, which is not ideal. A possible way to address this is to involve humans in the loop to judge samples' similarities visually. Nevertheless, besides the scalability issues, the qualitative results presented in [59] show that the reconstructed samples are not very visually close to the original ones (see Figure 8 of [59]). Another reason is that Updates-Leak relies on a pretrained autoencoder which may introduce biases to the reconstructed sample. The latter (single-sample label inference) is a coarser-grained attack aiming at "class-level" inference, while the membership inference attack in this paper is a finer-grained attack aiming at "sample-level" inference. We emphasize that membership inference is the most well-established privacy attack and arguably constitutes a bigger privacy threat [13, 28, 30, 37, 46, 48, 60, 66, 67, 75] than class-level attacks, such as the attacks in [7, 59] and model inversion [18, 19].

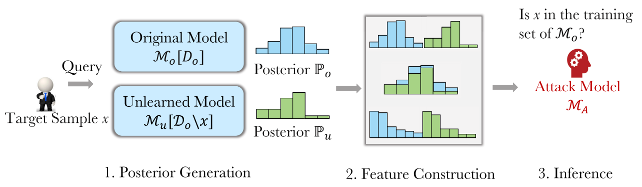


Figure 1: A schematic view of the general attack pipeline. The membership status of the target sample x is leaked due to the two versions of model.

3 MEMBERSHIP INFERENCE IN MACHINE UNLEARNING

3.1 Attack Pipeline

The general attack pipeline of our attack is illustrated in Figure 1. It consists of three phases: posteriors generation, feature construction and (membership) inference.

Posteriors Generation. The adversary has access to two versions of the target ML model, the original model $\mathcal{M}_o$ and the unlearned model $\mathcal{M}_u$. Given a target sample x, the adversary queries $\mathcal{M}_o$ and $\mathcal{M}_u$, and obtains the corresponding posteriors, i.e., $\mathbb{P}_o$ and $\mathbb{P}_u$, also referred to as confidence values [64].

Feature Construction. Given the two posteriors $\mathbb{P}_o$ and $\mathbb{P}_u$, the adversary aggregates them to construct the feature vector $\mathcal{F}$. There are several alternatives to construct the feature vector. We discuss five representative methods in subsection 3.3.

Inference. Finally, the adversary sends the obtained $\mathcal{F}$ to the attack model, which is a binary classifier, to determine whether the target sample x is in the training set of the original model. We describe how to build the attack model in subsection 3.2.

3.2 Attack Model Training

We assume the adversary has a local dataset, which we call the *shadow dataset* $\mathcal{D}^s$. The shadow dataset can come from a different distribution than the one used to train the target model. To infer whether the target sample x is in the original model or not, our core idea is to train an attack model $\mathcal{M}_A$ that captures the difference between the two posteriors. The intuition is that, if the target sample x is deleted, the two models $\mathcal{M}_o$ and $\mathcal{M}_u$ will behave differently. Figure 2 illustrates the training process of the attack model, and the detailed training procedure is presented as follows.

Training Shadow Models. To mimic the behavior of the target model, the adversary needs to train a shadow original model and a set of shadow unlearned models. To do this, the adversary first partitions $\mathcal{D}^s$ into two disjoint parts, the shadow negative set $\mathcal{D}^s_n$ and the shadow positive set $\mathcal{D}^s_p$. The shadow positive set $\mathcal{D}^s_p$ is used to train the shadow original model $\mathcal{M}^s_o$. The shadow unlearned model $\mathcal{M}^s_u$ is trained by deleting samples from $\mathcal{D}^s_p$. For ease of presentation, we assume the shadow unlearned model $\mathcal{M}^s_u$ is obtained by deleting exactly one sample. We will show that our attack is still effective for group deletion in subsection 6.2. The adversary randomly generates a set of deletion requests (target samples) $\mathcal{R}_p = \{x^1_p, x^2_p, \cdots, x^m_p\}$ and train a set of shadow unlearned models $\mathcal{M}^{s,1}_u, \mathcal{M}^{s,2}_u, \cdots, \mathcal{M}^{s,m}_u$, where the shadow unlearned model $\mathcal{M}^{s,i}_u$ is trained on dataset $\mathcal{D}^s_p \setminus x^i_p$.

Obtaining Posteriors. At the posteriors generation phase, the adversary feeds each target sample $x^i_p \in \mathcal{R}_p$ to the shadow original model $\mathcal{M}^s_o$ and its corresponding shadow unlearned model $\mathcal{M}^{s,i}_u$, and gets two posteriors $\mathbb{P}^i_o$ and $\mathbb{P}^i_u$.

Constructing Features. The adversary then uses the feature construction methods discussed in subsection 3.3 to construct training cases for the attack model. In classical membership inference, posteriors of $x^i_p \in \mathcal{R}_p$ serve as member cases of the attack model. But in the machine unlearning setting, $x^i_p \in \mathcal{R}_p$ is member of the shadow original model $\mathcal{M}^s_o$ and non-member of the shadow unlearned model $\mathcal{M}^s_u$. To avoid confusion, we call the samples related to $x^i_p \in \mathcal{R}_p$ *positive* cases instead of member cases for the attack model.

To train the attack model, the adversary also needs a set of negative cases. This can be done by sampling a set of negative query samples $\mathcal{R}_n$ from the shadow negative dataset $\mathcal{D}^s_n$ and query the shadow original model and unlearned model. To get a good attack model generalization performance, the adversary needs to ensure that the number of positive cases and the number of negative cases of the attack model are balanced, i.e., $|\mathcal{R}_p| = |\mathcal{R}_n|$, where $|\cdot|$ is the cardinality of the sample set.

Improving Diversity. To improve the diversity of the attack model, the adversary obtains multiple shadow original models by randomly sampling multiple subsets of samples from the shadow positive dataset $\mathcal{D}^s_p$. For each shadow original model, the adversary randomly generates a set of deletion requests and trains a set of shadow unlearned models. In subsection 5.5, we conduct empirical experiments to show the impact of the number of shadow original models on the attack performance.

Training the Attack Model. Given sets of positive cases with features and negative cases with features, we adopt four standard and widely used classifiers as our attack model: Logistic regression, decision tree, random forest, and multi-layer perceptron.

3.3 Feature Construction

Given the two posteriors, a straightforward approach to aggregate the information is to concatenate them, i.e., $\mathbb{P}_o||\mathbb{P}_u$, where $||$ is the concatenation operation. This preserves the full information. However, it is possible that the concatenation contains redundancy. In order to reduce redundancy, we can instead rely on the difference between $\mathbb{P}_o$ and $\mathbb{P}_u$ to capture the discrepancy left by the deletion of the target sample. In particular, we make use of the element-wise difference $\mathbb{P}_o - \mathbb{P}_u$ and the Euclidean distance $\left\|\mathbb{P}_o - \mathbb{P}_u\right\|_2$.

In order to better capture the level of confidence of the model, one may also sort the posteriors before the difference or concatenation operations [20]. Specifically, we sort the original posteriors $\mathbb{P}_o$ in descending order and get the sorted original posteriors $\mathbb{P}^s_o$. We then rearrange the order of the unlearned posteriors $\mathbb{P}_u$ to align its elements with $\mathbb{P}_o$, and get the sorted unlearned posteriors $\mathbb{P}^s_u$.

To summarize, we adopt the following five methods to construct the features for the attack model:

- Direct concatenate (DirectConcat), i.e., $\mathbb{P}_o||\mathbb{P}_u$
- Sorted concatenate (SortedConcat), i.e., $\mathbb{P}^s_o||\mathbb{P}^s_u$

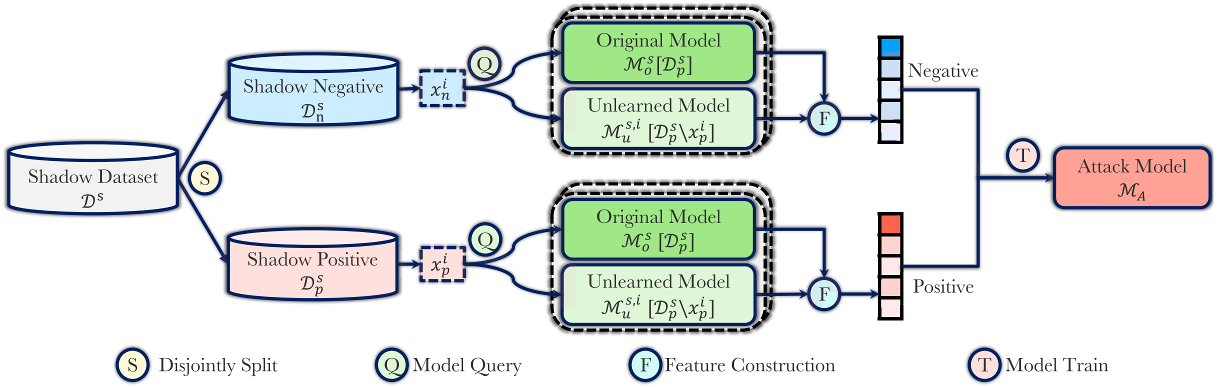

Shadow Dataset $\mathcal{D}^s$
S
Shadow Negative $\mathcal{D}^s_n$
Shadow Positive $\mathcal{D}^s_p$
x^i_n
x^i_p
Q
Q
Original Model $\mathcal{M}^s_o[\mathcal{D}^s_p]$
Unlearned Model $\mathcal{M}^{s,i}_u\ [\mathcal{D}^s_p \backslash x^i_p]$
Original Model $\mathcal{M}^s_o\ [\mathcal{D}^s_p]$
Unlearned Model $\mathcal{M}^{s,i}_u\ [\mathcal{D}^s_p \backslash x^i_p]$
F
F
Negative
Positive
T
Attack Model $\mathcal{M}_A$
S Disjointly Split
Q Model Query
F Feature Construction
T Model Train

Figure 2: Training process of the attack model. The shadow dataset $\mathcal{D}^s$ is split into disjoint shadow positive dataset $\mathcal{D}^s_p$ and shadow negative dataset $\mathcal{D}^s_n$. The shadow positive dataset $\mathcal{D}^s_p$ is used to train the shadow original model $\mathcal{M}^s_o$. The shadow unlearned model $\mathcal{M}^{s,i}_u$ is trained on $\mathcal{D}^s_p \setminus x^i_p$, where $x^i_p \in \mathcal{D}^s_p$. In the inference phase, the adversary first uses target sample x^i_p to query the original and unlearned models simultaneously to generate the positive features. Then they use a random sample $x^i_n \in \mathcal{D}^s_n$ to query the corresponding models to generate the negative features. Finally, they use the positive and negative features to train the attack model $\mathcal{M}_A$.

- Direct difference (DirectDiff), i.e., $\mathbb{P}_o - \mathbb{P}_u$.
- Sorted difference (SortedDiff), i.e., $\mathbb{P}^s_o - \mathbb{P}^s_u$.
- Euclidean distance (EucDist), i.e., $\left\|\mathbb{P}_o - \mathbb{P}_u\right\|_2$

In subsection 5.3, we conduct empirical experiments to evaluate the performance of the above methods and provide a high-level summary of the best features to use depending on the behavior of the underlying ML model.

4 PRIVACY DEGRADATION MEASUREMENT

In this paper, we aim to evaluate to what extent machine unlearning may degrade the membership privacy of an individual whose data sample has been deleted from the training set (we also call this the target sample). Specifically, we want to quantify the additional privacy degradation our attack brings over classical membership inference (or the improvement of membership inference) in order to measure the unintended information leakage due to data deletion in machine learning. To this end, we propose two privacy degradation metrics that measure the difference in the confidence levels of our attack and classical membership inference when predicting the correct membership status of the target sample.

Given n target samples x^1 to x^n, define p^i_u as the confidence of our attack in classifying x^i as a member, and p^i_m as the confidence of classical membership inference. Let b^i be the true status of x^i, i.e., $b^i = 1$ if x^i is a member, and $b^i = 0$ otherwise. With that, we define the following two metrics:

- **DegCount.** DegCount stands for Degradation Count. It calculates the proportion of target samples whose true membership status is predicted with higher confidence by our attack than by classical membership inference. Formally, DegCount is defined as

$$DegCount = \frac{1}{n}\sum_{i}^{n}\left[b^i \mathbb{1}_{p^i_u > p^i_m} + (1-b^i)\mathbb{1}_{p^i_u < p^i_m}\right]$$

where $\mathbb{1}_P$ is the indicator function which equals 1 if P is true, and 0 otherwise. Higher DegCount means higher privacy degradation.

- **DegRate.** DegRate stands for Degradation Rate. It calculates the average confidence improvement rate of our attack predicting the true membership status compared to classical membership inference. DegRate can be formally defined as

$$DegRate = \frac{1}{n}\sum_{i}^{n}\left[b^i(p^i_u - p^i_m) + (1-b^i)(p^i_m - p^i_u)\right]$$

Higher DegRate means higher privacy degradation.

5 EVALUATION

In this section, we conduct extensive experiments to evaluate the unintended privacy risks of machine unlearning. We first conduct an end-to-end experiment to validate the effectiveness of our attack on multiple datasets using the most straightforward unlearning method, i.e., retraining from scratch. Second, we compare different feature construction methods proposed in subsection 3.3 and provide a summary of the most appropriate to choose depending on the context. Third, we evaluate the impact of overfitting and of different hyperparameters. Fourth, we conduct experiments to evaluate dataset and model transferability between shadow model and target model. Finally, we show the effectiveness of our attack against the SISA unlearning method. We leave the evaluation of our attack in other scenarios to section 6.

5.1 Experimental Setup

Target Models. In our experiments, we evaluate the vulnerability of both simple machine learning models, including logistic regression (LR), decision tree (DT), random forest (RF), and 5-layer multi-layer perceptron (MLP), and the state-of-the-art convolutional neural networks, including SimpleCNN (implemented by us), DenseNet [31], and ResNet50 [29]. For reproducibility purpose, we provide the hyperparameter settings for the simple machine

Table 1: Dataset statistics.

Dataset	Type	Feature Dimension	#. Classes	#. Samples
Adult	Categorical	14	2	50,000
Accident	Categorical	30	3	3,000,000
Insta-NY	Categorical	169	9	19,215
Insta-LA	Categorical	169	9	16,472
MNIST	Image	28*28*1	10	42,000
CIFAR10	Image	32*32*3	10	60,000
STL10	Image	32*32*3	10	13,000

learning models in Appendix B, and the implementation details of SimpleCNN in Appendix C. All the convolutional networks are trained for 100 epochs.

Datasets. We run experiments on two different types of datasets: categorical datasets and image datasets. The categorical datasets are used to evaluate the vulnerability of simple machine learning models, while the image datasets are used to evaluate the vulnerability of the convolutional neural networks. Due to space limitation, we defer the detailed description of these datasets to Appendix A. The statistics of all datasets used in our experiments are listed in Table 1. Note that the Insta-NY and STL10 datasets are only used for data transferring attack in subsection 5.6.

Metrics. In addition to the two privacy degradation metrics proposed in section 4, we also rely on the traditional AUC metric to measure the absolute performance of our attack and classical membership inference. To summarize, we have the following three metrics:

- **AUC.** It is a widely used metric to measure the performance of binary classification in a range of thresholds [5, 19, 27, 36, 56, 60]. An AUC value equals to 1 shows a maximum performance while an AUC value of 0.5 shows a performance equivalent to random guessing.
- **DegCount.** It stands for Degradation Count, which is defined in section 4.
- **DegRate.** It stands for Degradation Rate, which is defined in section 4.

Experimental Settings. We evenly split each dataset $\mathcal{D}$ into disjoint target dataset $\mathcal{D}^t$ and shadow dataset $\mathcal{D}^s$. In subsection 5.6, we will show that the shadow dataset can come from a different distribution than the target dataset. The shadow dataset $\mathcal{D}^s$ is further split into shadow positive dataset $\mathcal{D}^s_p$ and shadow negative dataset $\mathcal{D}^s_n$ (80% for $\mathcal{D}^s_p$ and 20% for $\mathcal{D}^s_n$). We randomly sample S_o subsets of samples from $\mathcal{D}^s_p$, each containing S_r samples, to train S_o shadow original models. For each shadow original model $\mathcal{M}^{s,i}_o$, we train S_u shadow unlearned models on $\mathcal{D}^{s,i}_o \setminus x$. We split the target dataset $\mathcal{D}^t$ in a similar way as the shadow dataset $\mathcal{D}^s$.

By default, we set the hyperparameters of the shadow models to $S_o = 20, S_r = 5000, S_u = 100$, and the corresponding hyperparameters of the target models to $T_o = 20, T_r = 5000, T_u = 100$. These hyperparameters have shown to achieve good balance between computational overhead and attack performance in subsection 5.5.

Implementation. All algorithms are implemented in Python 3.7 and the experiments are conducted on a Ubuntu 19.10 LTS server with Intel Xeon E7-8867 v3 @ 2.50GHz and 1.5TB memory.

5.2 Evaluation of the Scratch Method

In this subsection, we conduct end-to-end experiments to evaluate our attack against the most straightforward approach of retraining the ML model from scratch.

Setup. We start by considering the scenario where only one sample is deleted for each unlearned model. The scenario where multiple samples are deleted before the ML model is retrained will be evaluated in subsection 6.2. We conduct the experiment on both categorical datasets and image datasets with three evaluation metrics, namely AUC, DegCount, DegRate, and report the results with the optimal features as explained in subsection 5.3.

Results for Categorical Datasets. Figure 3 depicts the attack performance of categorical datasets. In general, we observe that our attack performs consistently better than classical membership inference on all datasets, target models, attack models, and metrics. Compared to classical membership inference, our attack achieves up to 0.48 improvement of the AUC. The best DegCount and DegRate values are of 0.94 and 0.40, respectively. This indicates that our attack indeed degrades membership privacy of the target sample in the machine unlearning setting. Comparing the performance of different target models, we observe that decision tree is the most vulnerable ML model. We posit this is due to the fact that the decision tree forms a tree structure and deleting one sample could explicitly change its structure; thus the posterior difference of decision tree's original model and unlearned model is more significant, leading to a better attack performance.

Results for Image Datasets. Figure 4 illustrates the performance for the image datasets and complex convolutional neural networks. We keep the same attack models as categorical datasets, and use the SimpleCNN model for MNIST, use the ResNet50 and DenseNet models for CIFAR10. In general, we also observe that our attack outperforms classic membership inference attack in all settings. Besides, CIFAR10 trained with DenseNet shows the highest privacy degradation, while MNIST dataset trained with SimpleCNN shows the lowest. The reason behind is that the overfitting level of CIFAR10 trained with DenseNet is the largest. To further confirm this, we list the overfitting level of different models in Table 2. We observe that the overfitting level of CIFAR10 trained with DenseNet is 0.439, while the MNIST dataset trained with SimpleCNN has an overfitting level smaller than 0.05.

5.3 Finding Optimal Features

Figure 5 illustrates the attack AUC of different feature construction methods. We compare two different types of target models: (a) the well-generalized model logistic regression (trained on Insta-NY dataset), and (b) the overfitted model ResNet50 (trained on CIFAR10 dataset). We then apply the 5 different feature construction methods proposed in subsection 3.3 to 4 different attack models, resulting in 20 combinations. For comparison, we also include the classical membership inference as a baseline.

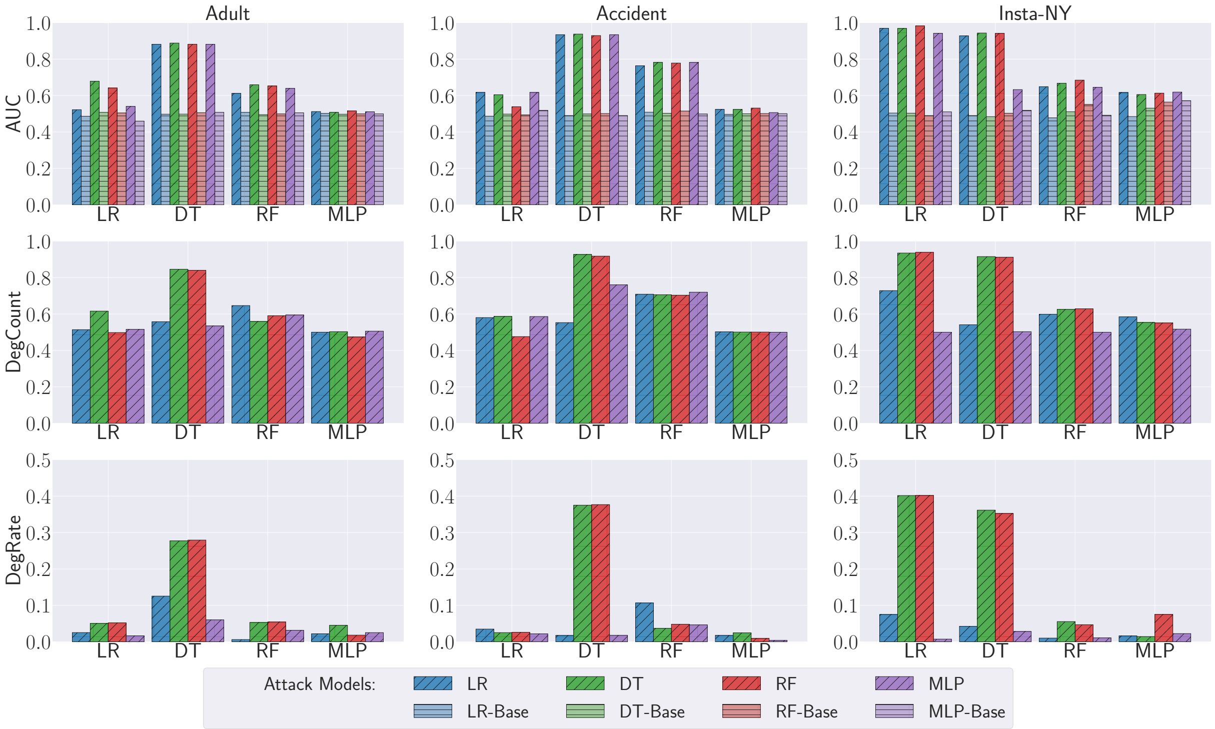


Figure 3: Privacy degradation level on the Scratch **method for three categorical datasets. Rows stand for different metrics, columns stand for different datasets. In each subfig, the groups in x-axis represent different target models, and the legends (in different colors) represent different attack models. For the AUC metric, the right bars (transparent ones) stand for the AUC value of classical membership inference.**

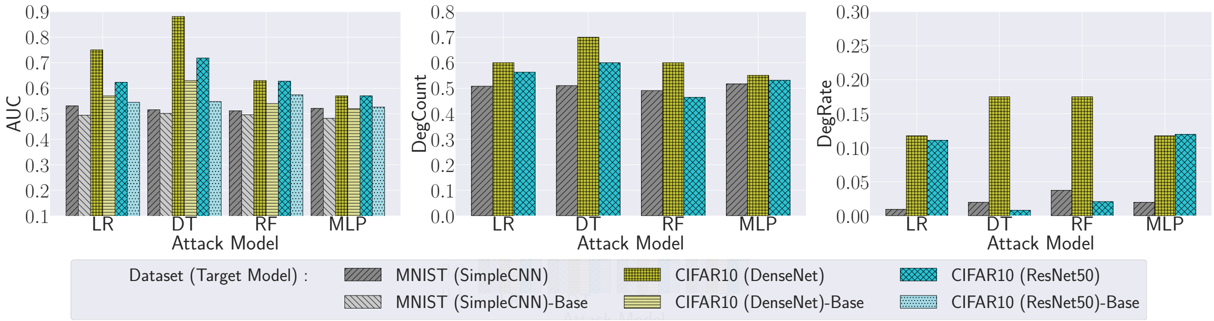
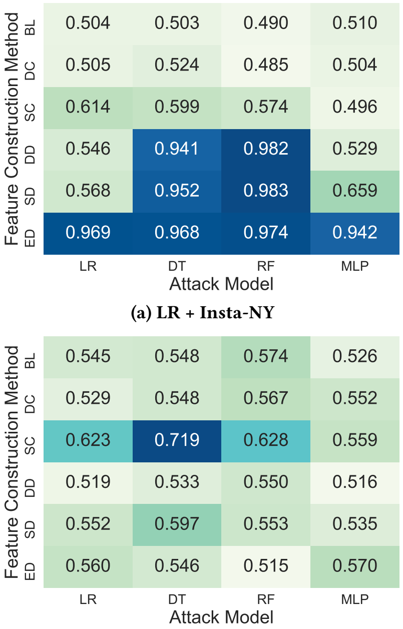


Figure 4: Privacy degradation level on the Scratch **method for image datasets. In each subfig, the groups in x-axis stand for different attack models, the legends (in different colors) stand for different datasets and the corresponding target models. For the AUC metric, the right bars (transparent ones) stand for the AUC value of classical membership inference.**

Concatenation vs. Difference. Concatenation-based methods (DirectConcat, SortedConcat) directly concatenate the two posteriors to preserve the full information, while difference-based methods capture the discrepancy between two versions of posteriors. We use two approaches to capture this discrepancy: element-wise difference (DirectDiff, SortedDiff) and Euclidean distance (EucDist).

Overall, Figure 5 shows that, on one hand, concatenation-based methods perform better on the overfitted model, i.e., ResNet50. On the other hand, the difference-based methods perform better on the well-generalized model, i.e., logistic regression. We suspect this is due to the fact that the concatenation-based methods rely on the plain posterior information, which can provide a strong signal for membership inference on the overfitted target model. This is consistent with the conclusion of previous studies [60, 64] that classical membership inference (which uses plain posterior information) performs well on overfitted target models. While we can also exploit the difference-based methods to mount the attack on the overfitted target models, the attack signal is not as strong as that of the concatenation-based methods as shown in Figure 5b. For the well-generalized target models, exploiting the plain posterior information has shown to perform poorly in terms of membership inference [60, 64]. In this case, the discrepancy information between



(a) LR + Insta-NY

(b) ResNet50 + CIFAR10

Figure 5: Attack AUC for different feature construction methods for target models (a) logistic regression (trained on Insta-NY) and (b) ResNet50 (trained on CIFAR10). DC, SC, DD, SD, ED **stand for** DirectConcat, SortedConcat, DirectDiff, SortedDiff, EucDist, **respectively.** BL **stands for the baseline, i.e., classical membership inference.**

two versions of the posteriors captured by the difference-based methods is more informative than the concatenation-based feature construction methods.

Sorted vs. Unsorted. Comparing DirectConcat to SortedConcat and DirectDiff to SortedDiff in Figure 5, we observe that the attack AUC of both concatenation-based method and difference-based method are clearly better after sorting. These results confirm our conjecture that sorting could improve the confidence level of the adversary.

Feature Selection Summary. Our empirical comparison provides us with the following rules for the feature construction methods: (1) use concatenation-based methods on overfitted models; (2) use difference-based methods on well-generalized models; (3) sort the posteriors before the concatenation and difference operations.

5.4 Impact of Overfitting

Overfitting measures the difference of accuracy between training and testing data. Previous studies [45, 64, 75] have shown that overfitted models are more susceptible to classical membership inference attacks, while well-generalized models are almost immune to them. In this subsection, we want to revisit the impact of overfitting on our attack.

Table 2: Attack AUC in different overfitting levels.

Dataset	$\mathcal{M}_o$	Train / Test Acc.	Overfitting	AUC / Base-AUC
Adult	LR	0.795 / 0.782	0.013	0.600 / 0.505
	DT	0.853 / 0.834	0.019	0.882 / 0.497
	RF	0.852 / 0.843	0.009	0.659 / 0.459
	MLP	0.767 / 0.763	0.004	0.506 / 0.503
Accident	LR	0.702 / 0.698	0.002	0.538 / 0.494
	DT	0.722 / 0.701	0.021	0.929 / 0.501
	RF	0.730 / 0.709	0.021	0.78 / 0.499
	MLP	0.670 / 0.644	0.026	0.513 / 0.493
Insta-NY	LR	0.508 / 0.439	0.069	0.983 / 0.490
	DT	0.404 / 0.373	0.031	0.941 / 0.503
	RF	0.523 / 0.442	0.081	0.685 / 0.551
	MLP	0.738 / 0.483	0.255	0.619 / 0.553
MNIST	SimCNN	0.954 / 0.951	0.003	0.511 / 0.496
CIFAR10	DenseNet	0.942 / 0.477	0.465	0.881 / 0.630
	ResNet50	0.975 / 0.592	0.383	0.719 / 0.548

Table 2 depicts the attack AUC for different overfitting levels. We use random forest as attack model, and use SortedDiff and SortedConcat as feature construction method for well-generalized and overfitted target model, respectively. In general, our attack consistently outperforms the classical membership inference on both well-generalized and overfitted models. On the overfitted models, i.e., CIFAR10 datasets with ResNet50 and DenseNet as target model, we can observe that the classical membership inference also works. However, our attack can achieve much better performance. On the other hand, the experimental results show that **our attack can still correctly infer the membership status of the target sample in well-generalized models**. For example, when the target model is a decision tree, the overfitting level in Adult (Income) dataset is 0.019, thus decision tree can be regarded as a well-generalized model. While the performance of classical membership inference on this model is equivalent to random guessing (AUC = 0.497), our attack performs very well, with an AUC of 0.882.

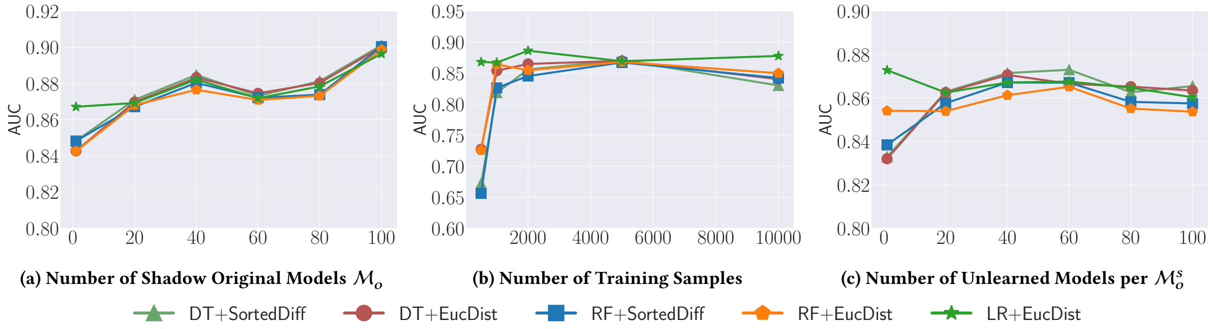
In summary, our attack performance is relatively independent of the overfitting level.

5.5 Ablation Study

We now evaluate the impact of the hyperparameters on the performance of our attack. Specifically, we focus on the number of shadow original models S_o, the number of samples S_r per shadow original model, and the number of unlearned models S_u per shadow original model. The corresponding hyperparameters of the target models are fixed (as defined at the end of subsection 5.1), since only the hyperparameters of the shadow models can be tuned to launch the attack.

We conduct the experiments on Adult (Income) dataset with decision tree as target model. Following our findings in subsection 5.3, we evaluate the attack AUC of different combinations of attack models, i.e., decision tree, random forest and logistic regression, and difference-based feature construction methods, i.e., DirectDiff, SortedDiff, EucDist.

Number of Shadow Original Models S_o. Figure 6a depicts the impact of S_o, which varies from 1 to 100. The figure shows that the



(a) Number of Shadow Original Models $\mathcal{M}_o$

(b) Number of Training Samples

(c) Number of Unlearned Models per $\mathcal{M}_o^s$

Figure 6: Attack AUC sensitivity to different hyperparameters on Adult (income) dataset with decision tree as target model. The legends stand for 5 combinations of attack models and feature construction methods guided by subsection 5.3.

Table 3: Attack AUC for dataset and model transfer. Names in the left of arrows stand for configurations of shadow model. Values in the parentheses stand for the attack AUC of the classical membership inference. Columns stand for dataset transfer, rows stand for model transfer.

Shadow→Target	**Insta-NY→Insta-NY**	Insta-NY→Insta-LA
DT→DT	**0.944 (0.491)**	0.931 (0.503)
DT→LR	0.964 (0.494)	0.974 (0.513)
LR→LR	**0.986 (0.505)**	0.982 (0.511)
LR→DT	0.927 (0.502)	0.926 (0.508)

Shadow→Target	**CIFAR10→CIFAR10**	CIFAR10→STL10
DenseNet →DenseNet	**0.881 (0.630)**	0.813 (0.621)
DenseNet→ResNet50	0.847 (0.624)	0.805 (0.632)
ResNet50→ResNet50	**0.719 (0.548)**	0.687 (0.550)
ResNet50→DenseNet	0.721 (0.523)	0.675 (0.542)

attack AUC sharply increases when S_o increases from 1 to 5, but remains quite stable for greater values of S_o. This indicates that setting $S_o = 5$ is enough for the diversity of the shadow original models.

Number of Samples S_r per Model. Figure 6b illustrates the impact of $S_r \in \{500, 1000, 2000, 5000, 10000\}$. When S_r increases from 500 to 1000, the attack AUC with SortedDiff increases from 0.67 to 0.83, while the attack AUC with EucDist increases from 0.73 to 0.86, except for logistic regression. However, adding more than 1000 samples does not help improve the attack performance further.

Number of Unlearned Models S_u per Shadow Original Model. Figure 6c illustrates the impact of S_u, which varies from 1 to 100. We observe that S_u has negligible impact on the attack AUC. This indicates that using a few unlearned models is sufficient to achieve a high attack performance.

5.6 Attack Transferability

In practice, the adversary might not able to get the same distribution dataset or same model structure to train the shadow models. We next validate the dataset and model transferability between shadow model and target model. That is, we evaluate whether the adversary can use a different dataset and model architecture than the target model to train the shadow models. We evaluate on both categorical datasets with simple model structure and image dataset with complex model structure.

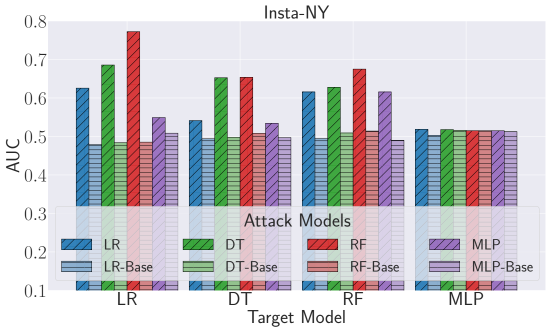


Figure 7: Attack AUC for the SISA method on the Insta-NY dataset. The transparent bars stand for the classical membership inference.

Dataset Transferability. Comparing the AUC values of the transfer setting with that of the non-transfer setting, i.e., bold rows in column Insta-NY→Insta-LA and CIFAR10→STL10, we only observe a small performance drop for all target models. For instance, when the target model is decision tree, the attack AUC of transfer setting and non-transfer setting are 0.944 and 0.931, respectively. The attack AUC only drops by 1%.

Model Transferability. For model transferring attack, we evaluate the pairwise transferability among decision tree and logistic regression. In Table 3, unbold rows in column Insta-NY→Insta-NY and CIFAR10→CIFAR10 illustrate the performance of model transfer. The experimental results show that model transfer only slightly degrades the attack performance of our attack. For example, when the shadow model and target model are both LR, the attack AUC equals to 0.986. When we change the target model to decision tree, the attack AUC is still of 0.927.

Dataset and Model Transferability. Unbold rows of unbold columns show the attack AUC when we transfer both the dataset and the model simultaneously. Even in this setting, our attack can achieve pretty good performance. This result shows robust transferability of our attack, when the adversary do not have access to same distribution data and same model architectures.

5.7 Evaluation of the SISA Method

The unlearning algorithm we focused on so far is retraining from scratch, which can become computationally prohibitive for large datasets and complex models. Several *approximate* unlearning algorithms have been proposed to accelerate the training process. In this subsection, we evaluate the performance of our attack against the most general approximate unlearning algorithm, SISA [10].

Setup. We remind the readers that the main idea of SISA is to split the original dataset into k disjoint shards and train k sub-models. In the inference phase, the model owner aggregates the prediction of each sub-model to produce the global prediction using some aggregation algorithm. In this experiment, we set $k = 5$ and use posterior average as aggregation algorithm. Figure 7 illustrates the attack AUC on the Insta-NY dataset. We report the experimental results of four different target models and four different attack models. For each attack model, we select the best features following the principles described in subsection 5.3.

Results. The experimental results show that our attack performance drops compared to the Scratch algorithm. We posit this is because the aggregation algorithm of SISA reduces the influence of a specific sample on its global model. This observation further motivates the deployment of unlearning methods such as SISA in real-world applications.

6 ATTACK UNDER DIFFERENT SCENARIOS

Next, we evaluate the effectiveness of our attack in different scenarios that might exist in practice. We first focus on the case when there exists multiple intermediate versions of unlearned models. Second, we consider when a group of samples are deleted. Third, we investigate the online learning setting when multiple samples are deleted and added simultaneously. Finally, we evaluate the impact of unlearning on remaining samples' membership privacy. The experimental setup is the same as in section 5.

6.1 Multiple Intermediate Unlearned Models

Intermediate Models. As discussed in the threat model (subsection 2.3), the adversary gains access to the original model and unlearned model by continuously querying the black-box target model. In practice, the adversary obtains access to two consecutive versions of models by two consecutive queries. However, the model owner would produce an unlearned model every time when it receives deletion requests; thus, there might be multiple unlearned models between these two consecutive queries that is unknown to the adversary. We call these models *intermediate models.* Here, we evaluate the effectiveness of our attack when multiple intermediate unlearned models exist.

Setup. Due to space limitation, we concentrate on the Insta-NY dataset with three different target models, while the conclusions are consistent for other datasets. We use LR as the attack model and select the best features following the principles described in subsection 5.3. Figure 8a depicts the results. The x-axis represents the number of intermediate unlearned models we studied, i.e., $\{1, 10, 50, 100, 250\}$.

Results. The experimental results show that our attack consistently degrades privacy of the target sample comparing with the classical membership inference. In addition, the attack AUC drops when the number of intermediate models increases. This is expected since the impact of the target sample is masked by previously deleted samples. That is, if there exist multiple intermediate models, the discrepancy information between the original model and the unlearned model is contributed by both the target sample and other deleted samples corresponding to the intermediate models. In other words, the impact of the target sample and other deleted samples is entangled with each other, making the inference of the membership status of the target sample more difficult.

Note that the data samples are unlikely to be revoked very frequently in practice, and the number of intermediate models are unlikely to be very large, which means our attack is still effective in real-world settings. For instance, our attack AUC can achieve at least 0.84 when the number of intermediate models are less than 10 when the target model is LR.

6.2 Group Deletion

In practice, there could exist cases where a group of samples are deleted at once before generating the unlearned model. This can happen when multiple data owners request the deletion at the same time, or when the model owner caches the deletion requests and updates the model only when he has received numerous requests to save computational resources.

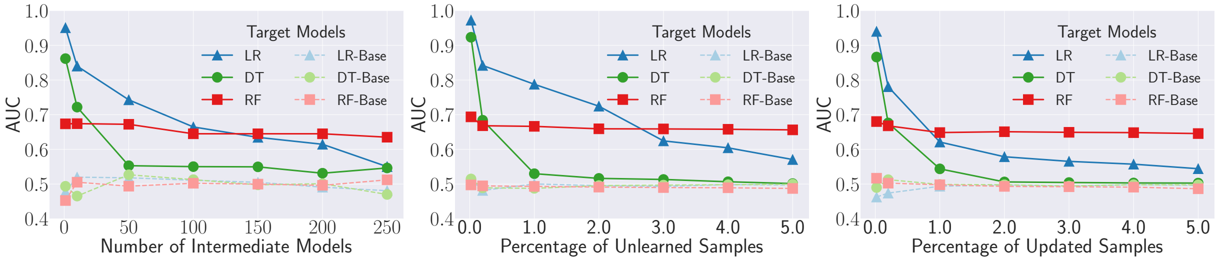
Setup. We conduct experiments on our attack in the group deletion scenario. We randomly delete a group of data samples from each original model to generate the unlearned model. The ratio of samples in each group takes value from $\{0.02\%, 0.2\%, 1\%, 2\%, 5\%\}$. We delete at most 5% of the data samples since in practice it is unlikely that more than 5% of users revoke their data. We evaluate our attack on the Insta-NY dataset with three target models. Notice that the unlearned model of the group deletion is the same as in subsection 5.7 when the group size equals the number of intermediate unlearned models. The difference is that in group deletion, we consider all samples in the group as target samples.

Results. Figure 8b shows that our consistently outperforms classical membership inference attack, demonstrating extra information leakage in group deletion. However, the attack performance of group deletion is slightly worse than single sample deletion, even though our attack is still effective when the group size is smaller than 0.2%. For example, when the target model is LR, the attack AUC of single deletion and group deletion (0.2% target samples) are 0.972 and 0.842, respectively. The reason is that a single sample could be hidden among the group of deleted samples, thereby preserving its membership information. This result reveals that conducting group deletion could mitigate, to some extent, the impact of our attack.

In practice, we believe 0.2% might already be too large for unlearning. The results of [8] show that 3.2 million requests for removing URLs have been issued to Google for 5 years which certainly constitutes less than 0.2% of the total URLs Google indexes.

6.3 Online Learning

In real-world deployments, ML models are often updated with new samples, which is known as online learning or incremental learning. Next, we evaluate the performance of our attack in online



(a) Multiple Intermediate Unlearned Models **(b) Group Deletion** **(c) Online Learning**

Figure 8: Attack AUC under different scenarios on Insta-NY. The dashed lines stand for the attack AUC of classical membership inference. Due to space limitation, we omit the results for other models and datasets that hold similar conclusions.

learning settings where multiple samples are deleted and added simultaneously.

Setup. To set up the experiment, we delete a group of target samples from the original dataset and add the same number of new samples; then we retrain the model from scratch to obtain the unlearned model. We conduct experiments on Insta-NY with different target models and use LR as the attack model.

Results. Figure 8c show that adding samples to the target model in the unlearning process has slight impact on our attack. For example, compared with purely deletion, the attack AUC only slightly drops from 0.972 to 0.940 when the target model is LR and the number of unlearned samples equals to 10 (0.2%).

6.4 Impact on Remaining Samples

In the end, we evaluate whether deleting the target sample can influence the privacy of other remaining samples.

Setup. We use the same attack pipeline described in subsection 3.1 to mount the attack. Concretely, we use data samples that reside in both the original model and unlearned model as positive cases, and use shadow/target negative dataset as negative cases. We concentrate on the Insta-NY dataset with four target models and four attack models where one data sample is deleted.

Results. Figure 9 shows that the attack AUC of our attack is higher than that of the classical membership inference, which only exploit information of the original model, indicating deleting the target sample also degrades privacy, to some extent, of the remaining samples. However, the attack AUC of all target models are less than 0.6, meaning the remaining samples are less sensitive to our attack. This is expected due to the fact that the remaining samples are members of both the original model and unlearned model. Deleting other data samples has some but limited impact on their posteriors in the unlearned model.

6.5 Takeaways

Through our extensive experiments in section 5 and section 6, we have made the following important observations:

- Our attack consistently degrades the membership privacy of the Scratch unlearning method compared to classical membership inference. The attack performance drops for the SISA unlearning method, which motivates the deployment of unlearning methods such as SISA in real-world applications.

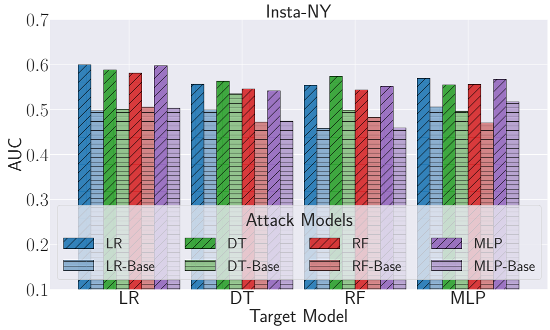


Figure 9: Attack AUC of the remaining samples on the Insta-NY dataset. The transparent bars stand for the classical membership inference.

- We obtain the following rules for selecting the feature construction methods: (1) use concatenation-based methods on overfitted models; (2) use difference-based methods on well-generalized models; (3) sort the posteriors before the concatenation and difference operations.
- Our transferring attacks show that our attack is still effective when the shadow model is trained on different-distributed datasets and different architecture from the target model.
- When the number of unlearned/updating samples represents less than 0.2% of the training dataset, our attack is still effective in the scenarios of multiple intermediate unlearned models, group deletion, and online learning.
- Deleting the target sample also degrades privacy of the remaining samples to some extent; however, the remaining samples are less sensitive to our attack.

7 POSSIBLE DEFENSES

In this section, we explore four possible defense mechanisms and empirically evaluate their effectiveness. The former two mechanisms reduce the information accessible to the adversary [64], and the latter two eliminate the impact of a single sample on the output of the ML models.

Publishing the Top-k Confidence Values. This defense reduces the attacker's knowledge by only publishing top k confidence values of the posteriors returned by both original and unlearned models. Formally, we denote the posterior vector as $\mathbb{P} = [p_1, p_2, \cdots, p_\ell]$, where ℓ is the number of classes of the target model and p_i is the confidence value of class i. When the target model receives

Table 4: Attack AUC of the defense mechanisms. We list the attack performance of no defense mechanisms (ND), publishing the Top-k confidence values (Top-1, Top-2, Top-3), label-only defense (Label), temperature scaling (TS) based defense, and differential privacy (DP) based defense. $\mathcal{D}_o$, $\mathcal{M}_T$, and $\mathcal{M}_A$ stand for original dataset, target model and attack model, respectively. For DP, we set $\delta = 10^{-5}, \epsilon_1 = 4.64, \epsilon_2 = 0.7$.

$\mathcal{D}_o$ ($\mathcal{M}_T$)	$\mathcal{M}_A$	ND	Top-1	Top-2	Top-3	Label	TS	DP[ϵ_1]	DP[ϵ_2]
Adult (DT)	RF	0.916	0.899	0.906	0.911	0.501	-	-	-
	DT	0.918	0.903	0.906	0.910	0.506	-	-	-
	LR	0.918	0.904	0.907	0.911	0.506	-	-	-
	MLP	0.918	0.904	0.909	0.907	0.493	-	-	-
Insta-NY (DT)	RF	0.937	0.930	0.931	0.942	0.506	-	-	-
	DT	0.938	0.932	0.932	0.943	0.502	-	-	-
	LR	0.928	0.923	0.927	0.926	0.502	-	-	-
	MLP	0.928	0.923	0.927	0.929	0.505	-	-	-
Insta-NY (LR)	RF	0.976	0.947	0.965	0.965	0.546	0.635	0.519	0.477
	DT	0.972	0.946	0.961	0.961	0.546	0.654	0.524	0.500
	LR	0.969	0.948	0.960	0.962	0.546	0.610	0.519	0.500
	MLP	0.970	0.948	0.960	0.966	0.453	0.653	0.506	0.504

a query, the model owner calculates posteriors $\mathbb{P}$ and sorts them in descending order, resulting in $\mathbb{P}^s = [p^s_1, p^s_2, \cdots, p^s_\ell]$. The model owner then publishes the first k values in $\mathbb{P}^s$, i.e., $[p^s_1, p^s_2, \cdots, p^s_k]$.

In the machine unlearning setting, the top k confidence values of the original model and the unlearned model may not correspond to the same set of classes. To launch our attack, the adversary constructs a *pseudo-complete posterior vector* for both original model and unlearned model. The pseudo-complete posteriors take the published confidence values for their corresponding classes, and evenly distributes the remaining confidence value to other classes, i.e., for $j \in \{k+1, \ldots, \ell\}$, $p^s_j = \frac{1-(p^s_1+p^s_2+\cdots+p^s_k)}{\ell-k}$. The adversary can then launch our attack using the pseudo-complete posteriors.

Table 4 shows the experimental results of Top-1, Top-2 and Top-3 defenses on Insta-NY and Adult. For the Adult dataset, we report the results of decision tree as the target model; for the Insta-NY dataset, we report the results of decision tree and logistic regression as the target model. We report the performance of 4 different attack models, each selecting the best feature following the principle described in subsection 5.3. The results show that publishing top k confidence value *cannot* effectively mitigate our attack.

Publishing the Label Only. This defense further reduces the information accessible to the adversary by only publishing the predicted label instead of confidence values (posteriors). To launch our attack, the adversary also needs to construct the pseudo-complete posteriors for both the original model and unlearned model. The main idea is to set the confidence value of the predicted class as 1, and set the confidence value of other classes as 0. Table 4 illustrates the performance of the “label only” defense. The experimental setting is similar to Top-k defense. The experimental results show that the “label only” defense can effectively mitigate our attack in all cases. The reason is that deleting one sample is unlikely to change the output label of a specific target sample.

It is worth noting that recent studies have shown that an adversary can recover to a large extent the posteriors from the label with the so-called sampling attack [40, 58]. In this case, our membership inference attack is still effective. We leave the investigation of the improved attack in the presence of label-only publishing defense as future work.

Temperature Scaling. Temperature scaling divides the logits vector by a learned scaling parameter, which is a simple yet effective approach to eliminate the over-confident problem of the output posteriors of neural networks [25]. This defense reduces the impact of a single sample on the output posteriors.

Table 4 illustrates the performance of the “temperature scaling” defense. We report the performance of 4 different attack models, each selecting the best feature following the principle described in subsection 5.3. The experimental results show that temperature scaling is an effective defense mechanism. However, this method is only applicable to neural networks whose last layer is softmax. Logistic regression in our experiment can be regarded as a neural network with one input layer and one softmax layer.

Differential Privacy (DP). DP [9, 16, 39, 49, 55] guarantees that any single data sample in a dataset has limited impact on the output. Previous studies have shown DP can effectively prevent classical membership inference attacks [34, 44]. To validate whether DP can prevent our membership inference attack in the machine unlearning setting, we train both the original model and unlearned model in a differentially private manner.

We experiment with Differentially-Private Stochastic Gradient Descent (DP-SGD) [4], the most representative DP mechanism for protecting machine learning models. The core idea of DP-SGD is to add Gaussian noise to the gradient g during the model training process, i.e., $\tilde{g} = g + \mathcal{N}\left(0, \Delta^2_f \sigma^2 \mathbf{I}\right)$. We use the Opacus library[3] developed by Facebook to conduct our experiments. Note that, since DP-SGD can only be applied to the ML models that encounter gradient updating in the training process, we only report the results for logistic regression. The last two columns of Table 4 illustrate the effectiveness of the DP defense. The experimental results show that DP can effectively prevent our membership inference attack. It worth noting that DP can inevitably degrade the target model’s accuracy. We need carefully tune the privacy budget parameters to strike a trade-off between privacy and model utility in practice.

We leave the in-depth exploration of more effective defense mechanisms against our attack as a future work.

8 RELATED WORK

Machine Unlearning. The notion of machine unlearning is first proposed in [11], which is the application of the right to be forgotten in the machine learning context. The most legitimate approach to implement machine unlearning is to remove the revoked samples from the original training dataset and retrain the ML model from scratch. However, retraining from scratch incurs very high computational overhead when the dataset is large and when the revoke requests happen frequently. Thus, most of the previous studies in

[3] https://github.com/pytorch/opacus

machine unlearning focus on reducing the computational overhead of the unlearning process [6, 10, 11, 32].

For instance, Cao et al. proposed to transform the learning algorithms into summation form that follows statistical query learning, breaking down the dependencies of training data [11]. To remove a data sample, the model owner only needs to remove the transformations of this data sample from the summations that depend on this sample. However, this algorithm is not applicable to learning algorithms that cannot be transformed into summation form, such as neural networks. Bourtoule et al. [10] proposed a more general algorithm named SISA. The main idea of SISA is to split the training data into disjoint shards, with each shard training one sub-model. To remove a specific sample, the model owner only needs to retrain the sub-model that contains this sample. To further speed up the unlearning process, the authors proposed to split each shard into several slices and store the intermediate model parameters when the model is updated by each slice.

Another line of machine unlearning study aims to verify whether the model owner complies with the data deletion request. Sommer et al. [65] proposed a backdoor-based method. The main idea is to allow the data owners to implant a backdoor in their data before training the ML model in the MLaaS setting. When the data owners later request to delete their data, they can verify whether their data have been deleted by checking the backdoor success rate.

The research problem in this paper is orthogonal to previous studies. Our goal is to quantify the unintended privacy risks for the deleted samples in machine learning systems when the adversary has access to both original model and unlearned model. To the best of our knowledge, this paper is the first to investigate this problem. Although quantifying privacy risks of machine unlearning has not been investigated yet, there are multiple studies on quantifying the privacy risks in the general right to be forgotten setting. For example, Xue et al. [74] demonstrate that in search engine applications, the right to be forgotten can enable an adversary to discover deleted URLs when there are inconsistent regulation standards in different regions. Ellers et al. [17] demonstrate that, in network embeddings, the right to be forgotten enables an adversary to recover the deleted nodes by leveraging the difference between the two versions of the network embeddings.

Membership Inference. Shokri et al. [64] presented the first membership inference attack against ML models. The main idea is to use shadow models to mimic the target model's behavior to generate training data for the attack model. Salem et al. [60] gradually removed the assumptions of [64] by proposing three different attack methods. Since then, membership inference has been extensively investigated in various ML models and tasks, such as federated learning [46], white-box classification [48], generative adversarial networks [13, 28], natural language processing [67], and computer vision segmentation [30]. Another line of study focused on investigating the impact of overfitting [37, 75] and of the number of classes of the target model [63] on the attack performance.

To mitigate the threat of membership inference, a plethora of defense mechanisms have been proposed. These defenses can be classified into three classes: reducing overfitting, perturbing posteriors, and adversarial training. There are several ways to reduce overfitting in the ML field, such as ℓ_2-regularization [64], dropout [60], and model stacking [60]. In [38], the authors proposed to explicitly reduce the overfitting by adding to the training loss function a regularization term, which is defined as the difference between the output distributions of the training set and the validation set. Jia et al. [36] proposed a posterior perturbation method inspired by adversarial example. Nasr et al. [47] proposed an adversarial training defense to train a secure target classifier. During the training of the target model, a defender's attack model is trained simultaneously to launch the membership inference attack. The optimization objective of the target model is to reduce the prediction loss while minimizing the membership inference attack accuracy.

Attacks against Machine Learning. Besides membership inference attacks, there exist numerous other types of attacks against ML models [20, 26, 33, 35, 37, 41, 43, 51–54, 57, 59, 61, 62, 69, 70, 72, 73]. Ganju et al. [20] proposed a property inference attack aiming at inferring general properties of the training data (such as the proportion of each class in the training data). Model inversion attack [18, 19] focuses on inferring the missing attributes of the target ML model. A major attack type in this space is adversarial examples [52–54, 69]. In this setting, an adversary adds carefully crafted noise to samples aiming at misleading the target classifier. A similar type of attack is the backdoor attack, where the adversary as a model trainer embeds a trigger into the model for them to exploit when the model is deployed [23, 43, 73]. Another line of work is model stealing, Tramèr et al. [70] proposed the first attack on inferring a model's parameters. Other works focus on inferring a model's hyperparameters [51, 72]. An interesting future work will be evaluating these attacks under machine unlearning.

9 CONCLUSION

This paper takes the first step to investigate the unintended information leakage in machine unlearning through the lens of membership inference. We propose several feature construction methods to summarize the discrepancy between the posteriors returned by original model and unlearned model. Extensive experiments on five real-world datasets show that our attack in multiple cases outperform the classical membership inference attack on the target sample, especially on well-generalized models. We further show that we can effectively infer membership information in other scenarios might exist in practice, including the scenario where there are multiple intermediate unlearned models, the scenario where a group of samples (instead of a single one) are deleted together from the original target model, and the online learning scenario where there are samples to be deleted and added simultaneously. Finally, we present four defense mechanisms to mitigate the newly discovered privacy risks. We hope that these results will help improve privacy in practical implementation of machine unlearning.

ACKNOWLEDGMENTS

We thank our shepherd Gautam Kamath and the anonymous reviewers for their constructive comments. This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4). Tianhao Wang is funded by National Science Foundation (NSF) under Grand No.1931443, a Bilsland Dissertation Fellowship, and a Packard Fellowship.

REFERENCES

[1] https://gdpr-info.eu/.

[2] https://oag.ca.gov/privacy/ccpa.

[3] https://laws-lois.justice.gc.ca/ENG/ACTS/P-8.6/index.html.

[4] Martin Abadi, Andy Chu, Ian Goodfellow, Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318. ACM, 2016.

[5] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring Social Links from Mobility Profiles. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1943–1957. ACM, 2017.

[6] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine Unlearning: Linear Filtration for Logit-based Classifier. *CoRR abs/2002.02730*, 2020.

[7] Santiago Zanella Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing Information Leakage of Updates to Natural Language Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 363–375. ACM, 2020.

[8] Theo Bertram, Elie Bursztein, Stephanie Caro, Hubert Chao, Rutledge Chin, FemanPeter Fleischer, Albin Gustafsson, Jess Hemerly, Chris Hibbert, Luca InvernizziLanah, Kammourieh Donnelly, Jason Ketover, Jay Laefer, Paul Nicholas, Yuan Niu, Harjinder Obhi, David Price, Andrew Strait, Kurt Thomas, and Al Verney. Five Years of the Right to be Forgotten. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 959–972. ACM, 2019.

[9] Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan R. Ullman. CoinPress: Practical Private Mean and Covariance Estimation. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[10] Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[11] Yinzhi Cao and Junfeng Yang. Towards Making Systems Forget with Machine Unlearning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 463–480. IEEE, 2015.

[12] Yinzhi Cao, Alexander Fangxiao Yu, Andrew Aday, Eric Stahl, Jon Merwine, and Junfeng Yang. Efficient Repair of Polluted Machine Learning Systems via Causal Unlearning. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 735–747. ACM, 2018.

[13] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 343–362. ACM, 2020.

[14] Adam Coates, Andrew Y. Ng, and Honglak Lee. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223. JMLR, 2011.

[15] Min Du, Zhi Chen, Chang Liu, Rajvardhan Oak, and Dawn Song. Lifelong Anomaly Detection Through Unlearning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1283–1297. ACM, 2019.

[16] Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy*. Now Publishers Inc., 2014.

[17] Michael Ellers, Michael Cochez, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. Privacy Attacks on Network Embeddings. *CoRR abs/1912.10979*, 2019.

[18] 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 (CCS)*, pages 1322–1333. ACM, 2015.

[19] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *USENIX Security Symposium (USENIX Security)*, pages 17–32. USENIX, 2014.

[20] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 619–633. ACM, 2018.

[21] Antonio A. Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making AI Forget You: Data Deletion in Machine Learning. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3513–3526. NeurIPS, 2019.

[22] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9301–9309. IEEE, 2020.

[23] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Grag. Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR abs/1708.06733*, 2017.

[24] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. Certified Data Removal from Machine Learning Models. In *International Conference on Machine Learning (ICML)*, pages 3832–3842. PMLR, 2020.

[25] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2017.

[26] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, and Gang Wang abd Xinyu Xing. LEMNA: Explaining Deep Learning based Security Applications. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 364–379. ACM, 2018.

[27] Inken Hagestedt, Yang Zhang, Mathias Humbert, Pascal Berrang, Haixu Tang, XiaoFeng Wang, and Michael Backes. MBeacon: Privacy-Preserving Beacons for DNA Methylation Data. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[28] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks. *Symposium on Privacy Enhancing Technologies Symposium*, 2019.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016.

[30] Yang He, Shadi Rahimian, Bernt Schiele1, and Mario Fritz. Segmentations-Leak: Membership Inference Attacks and Defenses in Semantic Image Segmentation. In *European Conference on Computer Vision (ECCV)*, pages 519–535. Springer, 2020.

[31] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE, 2017.

[32] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate Data Deletion from Machine Learning Models: Algorithms and Evaluations. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2008–2016. PMLR, 2021.

[33] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 1345–1362. USENIX, 2020.

[34] Bargav Jayaraman and David Evans. Evaluating Differentially Private Machine Learning in Practice. In *USENIX Security Symposium (USENIX Security)*, pages 1895–1912. USENIX, 2019.

[35] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-Reuse Attacks on Deep Learning Systems. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 349–363. ACM, 2018.

[36] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 259–274. ACM, 2019.

[37] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium (USENIX Security)*, pages 1605–1622. USENIX, 2020.

[38] Jiacheng Li, Ninghui Li, and Bruno Ribeiro. Membership Inference Attacks and Defenses in Supervised Learning via Generalization Gap. In *ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 5–16. ACM, 2021.

[39] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. *Differential Privacy: From Theory to Practice*. Morgan & Claypool Publishers, 2016.

[40] Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021.

[41] Xiang Ling, Shouling Ji, Jiaxu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. DEEPSEC: A Uniform Platform for Security Analysis of Deep Learning Model. In *IEEE Symposium on Security and Privacy (S&P)*, pages 673–690. IEEE, 2019.

[42] Yang Liu, Zhuo Ma, Ximeng Liu, Jian Liu, Zhongyuan Jiang, JianFeng Ma, Philip Yu, and Kui Ren. Learn to Forget: Memorization Elimination for Neural Networks. *CoRR abs/2003.10933*, 2020.

[43] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning Attack on Neural Networks. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[44] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. *CoRR abs/2102.02551*, 2021.

[45] 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. *CoRR abs/1802.04889*, 2018.

[46] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 497–512. IEEE, 2019.

[47] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646. ACM, 2018.

[48] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy*

(S&P), pages 1021–1035. IEEE, 2019.

[49] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2021.

[50] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-Delete: Gradient-Based Methods for Machine Unlearning. In *International Conference on Algorithmic Learning Theory (ICALT)*, pages 931–962. PMLR, 2021.

[51] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[52] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. SoK: Towards the Science of Security and Privacy in Machine Learning. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 399–414. IEEE, 2018.

[53] Nicolas Papernot, Patrick D. McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 506–519. ACM, 2017.

[54] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (Euro S&P)*, pages 372–387. IEEE, 2016.

[55] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable Private Learning with PATE. In *International Conference on Learning Representations (ICLR)*, 2018.

[56] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock Knock, Who's There? Membership Inference on Aggregate Location Data. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018.

[57] Erwin Quiring, Alwin Maier, and Konrad Rieck. Misleading Authorship Attribution of Source Code using Adversarial Learning. In *USENIX Security Symposium (USENIX Security)*, pages 479–496. USENIX, 2019.

[58] Shadi Rahimian, Tribhuvanesh Orekondy, and Mario Fritz. Differential Privacy Defenses and Sampling Attacks for Membership Inference. In *PriML Workshop (PriML)*. NeurIPS, 2020.

[59] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security Symposium (USENIX Security)*, pages 1291–1308. USENIX, 2020.

[60] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[61] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 6103–6113. NeurIPS, 2018.

[62] Dongdong She, Yizheng Chen, Abhishek Shah, Baishakhi Ray, and Suman Jana. Neutaint: Efficient Dynamic Taint Analysis with Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 364–380. IEEE, 2020.

[63] Reza Shokri, Martin Strobel, and Yair Zick. Exploiting Transparency Measures for Membership Inference: a Cautionary Tale. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI)*. AAAI, 2020.

[64] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.

[65] David Marco Sommer, Liwei Song, Sameer Wagh, and Prateek Mittal. Towards Probabilistic Verification of Machine Unlearning. *CoRR abs/2003.04247*, 2020.

[66] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine Learning Models that Remember Too Much. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 587–601. ACM, 2017.

[67] Congzheng Song and Vitaly Shmatikov. Auditing Data Provenance in Text-Generation Models. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 196–206. ACM, 2019.

[68] Congzheng Song and Vitaly Shmatikov. Overlearning Reveals Sensitive Attributes. In *International Conference on Learning Representations (ICLR)*, 2020.

[69] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations (ICLR)*, 2017.

[70] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX, 2016.

[71] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans Forget, Machines Remember: Artificial Intelligence and the Right to Be Forgotten. *Computer Law & Security Review*, 2018.

[72] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 36–52. IEEE, 2018.

[73] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 707–723. IEEE, 2019.

[74] Minhui Xue, Gabriel Magno, Evandro Cunha, Virgilio Almeida, and Keith W. Ross. The Right to be Forgotten in the Media: A Data-Driven Study. *Symposium on Privacy Enhancing Technologies Symposium*, 2016.

[75] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

A DATASETS

- **UCI Adult[4].** This is a widely used categorical dataset for classification. It is a census dataset that contains around 50, 000 samples with 14 features. The classification task is to predict whether the income of a person is over \$50k, which is a binary classification task.
- **US Accident[5].** This is a countrywide traffic accident dataset, which covers 49 states of the United States. This dataset contains around 3M samples. We filter out attributes with too many missing values and obtain 30 valid features. The valid features include temperature, humidity, pressure, etc. The classification task is to predict the accident severity level which contains 3 classes.
- **Insta-NY [5].** This dataset contains a collection of Instagram users' location check-in data in New York. Each check-in contains a location and a timestamp; and each location belongs to a category. We use the number of check-ins that happened at each location in each hour on a weekly basis as the location feature vector. The classification task is to predict each location's category among 9 different categories. After filtering out locations with less than 50 check-ins, we get 19,215 locations for Insta-NY dataset. Later in the section, we also make use of check-ins in Los Angeles, namely Insta-LA [5], for evaluating the data transferring attack. This dataset includes 16,472 locations.
- **MNIST[6].** MNIST is an image dataset widely use for classification. It is a 10-class handwritten digits dataset which contains 42,000 samples, each being formatted into a 28×28-pixel image.
- **CIFAR10[7].** CIFAR10 is the benchmark dataset used to evaluate image recognition algorithms. This dataset contains 60,000 colored images of size 32×32, which are equally distributed on the following 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 50,000 training images and 10,000 testing images.
- **STL10 [14].** STL10 is a 10-class image dataset with each class containing 1,300 images. Classes include airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck.

B HYPERPARAMETER SETTINGS OF SIMPLE MODELS

We use multiple ML models in our experiments. All models are implemented by sklearn version 0.22 except for the logistic regression

[4] https://archive.ics.uci.edu/ml/datasets/adult.

[5] https://www.kaggle.com/sobhanmoosavi/us-accidents

[6] http://yann.lecun.com/exdb/mnist/

[7] https://www.cs.toronto.edu/~kriz/cifar.html

classifier. For reproduction purpose, we list their hyperparameter settings as follows:

- **Logistic Regression.** We implement a single linear logistic regression classifier with PyTorch. Training with Adam optimizer for 100 epochs.
- **Decision Tree.** We use Gini index as criterion, set parameter max_leaf_nodes as 10, and set other hyperparameters as default.
- **Random Forest.** We use Gini index as criterion, use 100 estimators, set min_samples_leaf=30, and set other hyperparameters as default.
- **Multi-layer Perceptron.** For multi-layer-perceptron classifier, we use Adam optimizer and Relu activation function. And set the hidden layer size and learning rate to 128 and 0.001, respectively.

C IMPLEMENTATION OF SIMPLECNN

The architecture of our SimpleCNN is illustrated in Table 5. We train the SimpeCNN for 100 epochs, and use SGD optimizer with learning rate of 0.001.

Table 5: SimpleCNN structure and hyperparameter. For the MNIST dataset, input_channel $C_i = 1$, image width W and height H are both 28. The kernel_size of convolution layer K_c and Max-pooling layer K_m are 3 and 2, respectively.

Layer	Hyperparameters
Conv2D_1	(C_i, 32, K_c=3, 1)
Relu	-
Conv2D_2	(32, H, K_c, 1)
Maxpolling2D	K_m=2
Dropout_1	(0.25)
Flatten	1
Linear_1	$(H \times (W/2 - K + 1) \times (H/2 - K + 1)$, 128)
Relu	-
Dropout_2	0.5
Linear_2	(128, #classes)
Softmax	dim=1