Overconfidence is a Dangerous Thing: Mitigating Membership Inference Attacks by Enforcing Less Confident Prediction

Zitao Chen
University of British Columbia
zitaoc@ece.ubc.ca

Karthik Pattabiraman
University of British Columbia
karthikp@ece.ubc.ca

***Abstract*—Machine learning (ML) models are vulnerable to *membership inference attacks* (MIAs), which determine whether a given input is used for training the target model. While there have been many efforts to mitigate MIAs, they often suffer from limited privacy protection, large accuracy drop, and/or requiring additional data that may be difficult to acquire.**

This work proposes a defense technique, HAMP that can achieve both strong membership privacy and high accuracy, without requiring extra data. To mitigate MIAs in different forms, we observe that they can be unified as they all exploit the ML model's *overconfidence in predicting training samples* through different proxies. This motivates our design to *enforce less confident prediction by the model*, hence forcing the model to behave similarly on the training and testing samples. HAMP consists of a novel training framework with high-entropy soft labels and an entropy-based regularizer to constrain the model's prediction while still achieving high accuracy. To further reduce privacy risk, HAMP uniformly modifies all the prediction outputs to become low-confidence outputs while preserving the accuracy, which effectively obscures the differences between the prediction on members and non-members.

We conduct extensive evaluation on five benchmark datasets, and show that HAMP provides consistently high accuracy and strong membership privacy. Our comparison with seven state-of-the-art defenses shows that HAMP achieves a superior privacy-utility trade off than those techniques[1].

I. Introduction

Machine learning (ML) models are often trained with the sensitive or private user data like clinical records [23], financial information [32] and personal photos [22]. Unfortunately, ML models can also unwittingly leak private information [38], [11], [44], [13], [4]. One prominent example is *Membership inference attacks* (MIAs) [38], [31], [49], [39], [28], [48], [3], which determine whether an input is used for training the target model, Hence, MIAs constitute a fundamental threat to data privacy. For instance, by knowing that an individual's clinical record was used to train a hospital's diagnostic model, the adversary can directly infer his/her health status.

MIAs exploit the ML model's differential behaviors on members and non-members [38], [31], [49], [28], [39], [9], [3]. *Members* are the samples used to train the model (i.e., training samples) and *non-members* are the samples not used for training (e.g., testing samples). Existing MIAs can be divided into score-based [38], [31], [18], [49], [39], [3] and label-only attacks [9], [28], where the former requires access to the model's *output score* indicating the class probability, while the latter needs only the prediction label. These attacks all seek to learn distinctive statistical features from the model's predictions in different ways, such as training an attack inference model [31], [38], computing metrics like prediction loss [49], or using Gaussian likelihood estimate [3].

Defenses against MIAs can be categorized into provable and practical defenses. *Provable* defenses provide provable guarantees through differential privacy (DP) [2], but they often incur severe accuracy degradation. *Practical* defenses, instead, offer empirical membership privacy with the goal of maintaining high model accuracy [30], [42], [37], [20]. However, existing defenses still suffer from the following limitations: (1) limited privacy protection [20], [30]; (2) large accuracy drop [2], [30], [42]; (3) requiring additional public datasets that may not always be available in practice [33], [37]. To the best of our knowledge, no technique satisfies all these constraints, though they may address individual issues, e.g., high model accuracy but with limited privacy protection [20]; or strong privacy but with significant accuracy loss [2].

Our Approach. This paper proposes a practical defense called HAMP that can achieve both **H**igh **A**ccuracy and **M**embership **P**rivacy without requiring additional data. Existing MIAs employ diverse approaches in inferring membership, e.g., score-based MIAs may exploit prediction loss or entropy [49], [39], [31] while label-only MIAs [9], [28] can leverage adversarial robustness. Despite the different manifestations of these attacks, we identify a common exploitation thread among them - they are all learning to distinguish whether the model is *overly confident* in predicting the training samples via different proxies. Our defense is therefore to *reduce the model's overconfident prediction on training samples while preserving the model's prediction performance*, which can simultaneously reduce membership leakage (from different MIAs) and maintain model accuracy.

[1] Our code is available at https://github.com/DependableSystemsLab/MIA_defense_HAMP.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23014
www.ndss-symposium.org

HAMP consists of a training- and testing-time defense.

Training-time defense. Our key idea is to explicitly enforce the model to be less confident in predicting training samples during training. We first identify that the prevailing use of *hard labels* in common training algorithms is one of the main factors that lead to the model's excessive confidence in predicting training samples. Hard labels assign 1 to the ground-truth label class and 0 elsewhere. The model is trained to produce outputs that match the labels, i.e., near 100% probability for the ground-truth class and 0% otherwise. On the other hand, a non-member sample that is not seen during training, is usually predicted with lower confidence, and can hence be distinguished by the adversary from member samples.

We therefore propose a new training framework that gets rid of hard labels and instead uses (1) *High-entropy soft labels*, which are soft labels with high entropy that assign a much lower probability to the ground-truth class and non-zero probability for other classes. This explicitly enforces the model to make less confident prediction on training samples. (2) HAMP also consists of an *entropy-based regularizer*, which is to penalize the model for predicting any high-confidence outputs via regularizing the prediction entropy during training.

The proposed training framework is able to significantly reduce the model's overconfident prediction and improve membership privacy, without (severely) degrading the model accuracy. Section III-B explains how it prevents privacy leakage from different sources (output scores and prediction labels). On the other hand, stronger membership privacy can also be achieved (e.g., by increasing the strength of regularization), but it would be at the cost of accuracy, which is undesirable as both privacy and accuracy are important considerations. This motivates our testing-time defense, whose goal is to gain higher membership privacy without degrading accuracy.

Testing-time defense. We propose to uniformly modify *all* the outputs (from members and non-members) into low-confidence outputs, without changing the prediction labels. Our idea is to leverage the output scores from the *randomly-generated samples*, which are often predicted with low confidence due to the high dimensionality of the input space.

In our defense, all the values in each output score are replaced by those from random samples, and we keep the relative ordering of different classes unchanged to maintain the same prediction labels (e.g., a dog image is still predicted as a dog but with different output scores). Both the high-confidence outputs (on training samples) and low-confidence outputs (on testing samples) are uniformly replaced by such low-confidence outputs from random samples. This further reduces the membership leakage from the output scores.

Evaluation. We evaluate HAMP on five benchmark datasets (Purchase100, Texas100, Location30, CIFAR100 and CIFAR10), and perform comprehensive evaluation on a total of nine diverse MIAs (including the state-of-art LiRA attack [3]).

We compare HAMP with seven leading defenses: AdvReg [30], MemGuard [20], SELENA [42], DMP [37], Label Smoothing (LS) [41], Early-stopping [39], and DP-SGD [2].

An ideal privacy defense should offer strong protection for both members and non-members. Therefore, we follow Carlini et al. [3] to use attack true positive rate (TPR) controlled at low false positive rate (FPR), and attack true negative rate (TNR) at low false negative rate (FNR) to evaluate membership privacy. The former metric evaluates the privacy protection for members, and the latter for non-members.

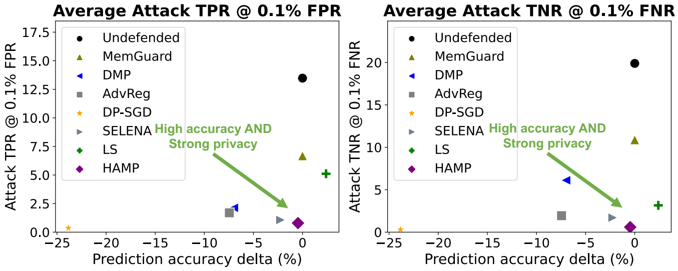


Fig. 1: Privacy and utility evaluation on each defense (results averaged across datasets). Negative accuracy delta means accuracy drop compared with the undefended models. DP-SGD is reported at $\epsilon = 4$. HAMP *simultaneously* achieve strong membership privacy (for both members and non-members) and high prediction accuracy, hence providing a better privacy-utility trade off than existing defenses.

Contributions. We summarize our contributions below.

- Develop a novel training framework with high-entropy soft labels and an entropy-based regularizer to enforce less confident prediction by the model, which can significantly mitigate diverse MIAs and incur minimal accuracy drop.
- Propose a novel testing time defense technique to modify all the output scores into low-confidence outputs, which further improves membership privacy without degrading accuracy.
- Integrate the training and testing framework as HAMP, and conduct rigorous evaluation under a wide range of attacks on five different datasets. We compare HAMP against seven leading defenses and show that HAMP outperforms existing defenses by achieving a superior privacy-utility trade off.

Fig. 1 summarizes the results of HAMP versus other defenses. We find that existing defenses often bias towards either privacy (e.g., DP-SGD) or utility (e.g., MemGuard). In contrast, HAMP is able to provide strong membership privacy for both members and non-members, and preserve model accuracy. HAMP reduces the attack TPR @0.1% FPR by 94% and the attack TNR @0.1% FNR by 97% respectively, with only 0.46% accuracy loss on average. This represents a much better privacy-utility trade off than other defenses.

II. Background

A. Machine Learning Primer

This work focuses on supervised training for classification problem. A ML model can be expressed as a function $F_\theta : X \to Y$, where $X \in \mathbb{R}^d$ denotes the input space and $Y \in \mathbb{R}^k$ the output space, and F is parameterized by weights θ. During training, the network is given a training set $(x, y) \in D_{tr}$ where y is the ground truth label. y is commonly expressed in the one-hot encoding format, where the ground-truth class is indicated with 1 and 0 elsewhere. The training objective is to minimize the prediction loss on the training set:

$$\min_{\theta} \frac{1}{|D_{tr}|} \sum_{x \in D_{tr}} \mathcal{L}(F_\theta(x), y), \tag{1}$$

where $|D_{tr}|$ denotes the size of the training set, and $\mathcal{L}$ the prediction loss such as cross-entropy loss. The model's output $F_\theta(x)$ indicates the probability of x belonging to each class with $\sum_{j=0}^{k-1} F_\theta(x)_j = 1$ that sums up to 1.

To prevent the model from overfitting on the training set, a separate validation set different from D_{tr} is commonly used to serve as an unbiased proxy of the testing set. One can use the accuracy on the validation set to assess how good the model will be when evaluated on test data and prevent overfitting.

Hereafter, we refer to F as the trained model F_θ, $F(x)$ as the output score of F on x, and D_{te} as the test set.

B. Threat Model

Attacker. Following prior work [20], [42], [30], we assume a black-box adversary who can query the target ML model with any input and observe the prediction output. The adversary's goal is to infer the membership of the training samples $(x, y) \in D_{tr}$ for a given model F. Like previous defenses [30], [42], [37], we assume a strong adversary with the knowledge of half of the training members and an equal number of non-members. Further, we assume the adversary has full knowledge of the defense technique and can therefore train shadow models in the same way as the target model is trained, which facilitates a strong adversary in evaluating the defenses.

Defender. We assume the defender has a private set D_{tr} and his/her goal is to train a model that can both achieve high classification accuracy and protect against MIAs. We do not assume the defender has access to any additional data.

C. Membership Inference Attacks

The attack model $h(x, y, F(x)) \rightarrow [0, 1]$ outputs the membership probability. We refer to D_{tr}^A, D_{te}^A as the set of members and non-members that are known to the adversary. The adversary's goal is to find a h that can best distinguish between D_{tr}^A and D_{te}^A. The empirical gain of the attack can be measured as:

$$\sum_{(x,y) \in D_{tr}^A} \frac{h(x, y, F(x))}{|D_{tr}^A|} + \sum_{(x,y) \in D_{te}^A} \frac{1 - h(x, y, F(x))}{|D_{te}^A|} \tag{2}$$

We categorize existing MIAs into *score-based* and *label-only* attacks as follows.

Score-based MIAs: This class of attacks either trains an inference model to infer membership [31], [38] or computes custom metrics such as prediction loss [49] to derive a threshold for distinction.

NN-based attack [31], [38] trains an neural network (NN) A, to distinguish the target model's prediction on members and non-members: $A : F(x) \rightarrow [0, 1], x \in [D_{tr}^A, D_{te}^A]$. By querying the target model with D_{tr}^A, D_{te}^A, the resulting output $(F(D_{tr}^A), 1)$, $(F(D_{te}^A), 0)$ forms the training set for A. In addition to output scores, other features like the ground-truth labels and prediction loss can also be used to train the inference model.

Loss-based attack [49] is based on the observation that the prediction loss on training samples is often lower than that on testing samples, as the loss on training samples are explicitly minimized during training. Specifically, the adversary can query the target model with D_{tr}^A, and obtain the average loss on D_{tr}^A as the threshold $\tau = -\frac{1}{|D_{tr}^A|} \sum_{(x,y) \in D_{tr}^A} \mathcal{L}(F_\theta(x), y)$. Any sample with loss lower than τ is considered as a member.

Entropy-based attack [38], [49] leverages that the output score of a training sample should be close to the one-hot encoded label, and hence its prediction entropy should be close to 0, which is lower than that on testing samples. Prediction entropy of a sample can be computed as $-\sum_j F(x)_j \log(F(x)_j)$, where j is the class index.

Modified-entropy-based attack [39] is an enhanced version of the entropy-based attack by computing the following metric: $-(1 - F(x)_y)\log(F(x)_y) - \sum_{j \neq y} F(x)_j \log(1 - F(x)_j)$. This attack improves by taking into account class-dependent thresholds, as well as the ground truth label y, which is shown to achieve higher attack effectiveness.

Confidence-based attack [49], [39] exploits the observation that the prediction confidence on training samples $F(x)_y$ is often higher than that on testing samples. The attack threshold can be derived similar to the entropy-based attacks, and samples predicted with high confidence are deemed as members.

Likelihood Ratio Attack (LiRA) [3] is a state-of-art attack that can successfully infer membership when calibrated at low false positive. In LiRA, the adversary trains N shadow models, half of which are trained with target sample (called IN models) and the remaining half are trained without the target sample (called OUT models). It then fits two Gaussian distributions to approximate the output distributions by the IN and OUT models (a logit scaling step on the logit values is taken to ensure the outputs follow a Gaussian). Finally, LiRA conducts a parametric likelihood-ratio test to conduct membership inference (e.g., a sample is deemed as a member if its output is estimated to come from the IN models with high probability).

Label-only MIAs: These attacks exploit training members' *higher* degree of robustness to different perturbations (like adversarial perturbations, random noise), and develop different proxies to distinguish the degree of robustness by members and non-members.

Prediction-correctness attack [49] is the baseline label-only attack that simply determines any samples that are correctly classified as members. This attack is effective when the training accuracy is higher than the testing accuracy.

Boundary attack [9], [28] is based on the observation that it is easier to perturb a testing sample to change the prediction label than a training sample. This is because testing samples are often closer to the decision boundary and therefore more susceptible to perturbations. Using common attacks such as CW2 attack [5], the adversary measures the magnitude of perturbation needed to perturb $x \in [D_{tr}^A, D_{te}^A]$, based on which τ can be derived. A sample is deemed as a member if the amount of perturbation needed to change the prediction label is higher than τ (i.e., more difficult to be perturbed).

The adversary can also inject random noise to the samples (instead of adversarial perturbations), which is more efficient and useful in the cases where constructing the adversarial sample is difficult (e.g., for inputs with binary features) [9].

Augmentation attack [9] makes use of the samples' robustness to data augmentation and the idea is that training samples are often more resilient to data augmentation than testing samples. For instance, if an image was used to train a model, it should still be classified correctly when it is slightly translated. For each input x, the adversary first generates multiple augmented versions of x, and computes how many of them are correctly classified. Based on the classification outcome, the adversary trains an attack inference model to predict whether or not x is a member.

D. Defenses against MIAs

This section presents an overview of representative defenses against MIAs (a comprehensive survey of existing defenses is in Section VI).

Adversarial regularization (AdvReg) [30] trains the model to both achieve good model performance and protection against a shadow MIA adversary. During training, the defender first trains an attack inference model that tries to maximize the MIA gain, after which the protected model is trained to minimize the MIA gain and maximize the classification accuracy. This is instantiated as a min-max game in [30].

Distillation for membership privacy (DMP) [37]. Shejwalkar et al. propose DMP to defend against MIAs based on knowledge distillation. The idea is to distill the knowledge from an undefended model (trained on a private dataset) into a new public model using a new reference set. Privacy protection is enabled by thwarting the access of the public model to the private dataset as the public model is trained on a separate reference set. Such a reference set can be curated by assuming the availability of a public dataset or by using synthetic data. We consider the latter since we do not assume access to external data. This is because in many domains such as healthcare, the training data is private/proprietary, and thus such a public dataset may not be available. We hence consider a more realistic scenario in which the defender has no access to external data (similar to [42]).

SELf ENsemble Architecture (SELENA) [42]. SELENA also uses knowledge distillation. Its key idea is to partition the private dataset into different subsets and train a sub model on each of the subset (another technique with similar idea is proposed in [10]). For each sub model, there exists a subset of private dataset that was not used in its training, i.e., "reference set" for that sub model. Each sub model assigns the output scores on its "reference set", which constitutes the knowledge to the distilled. The knowledge from the ensemble of sub models is finally distilled into a new public model.

Early stopping [39], [6]. As the training proceeds, the model tends to overfit the training data and become susceptible to MIAs. Early stopping is a general solution in reducing overfitting [6] by training models with fewer epochs. Song et al. [39] find that this is useful in mitigating MIAs and we follow to include it as as a benchmark defense mechanism.

Differential privacy (DP) based defenses [2]. DP-based defenses leverage the formal framework of differential privacy to achieve rigorous privacy guarantee. This is done via injecting noise to the learning objective during training such as DP-SGD that adds noise to the gradients [2]. However, DP-based defenses often produce models with considerable accuracy drop, resulting in a poor privacy-utility tradeoff.

MemGuard [20]. Jia et al. propose to defend against MIAs via obfuscating the prediction scores. The idea is to fool the MIA adversary by constructing a noise vector to be added to the input (analogous to constructing adversarial samples), and make the outputs on members and non-members indistinguishable by the adversary.

Label Smoothing [41]. LS is a common regularization technique to improve model accuracy by using soft labels. LS replaces the one-hot label with a mixture of the one-hot label and uniform distribution using a smoothing intensity parameter. E.g., for a smoothing intensity of 0.3, the soft label becomes 80% cat, 10% dog, 10% frog; and a smoothing intensity of 0.6 yields 60% cat, 20% dog, 20% frog. LS trains with different smoothing intensities to produce model with high accuracy.

Both LS and HAMP use soft labels in their training, but they are two techniques built with different principles that require different soft labels. LS is used to improve model performance, which necessitates training with *low*-entropy soft labels. Unlike LS, HAMP consists of *high*-entropy soft labels, an entropy-based regularizer and a novel testing-time defense (details in the next section), which is to improve membership privacy while preserving model accuracy. This consequently results in the different privacy implications by the two techniques: LS improves model performance but the resulting model still suffers from *high* MIA risk [21], while HAMP consistently contributes to very *low* MIA risk. We refer to detailed comparison in Section IV-G.

III. Methodology

The main insight behind HAMP in mitigating diverse MIAs is to identify a common exploitation thread among different MIAs. HAMP is designed to overcome this exploitation so that it can defend against different MIAs regardless of their specific approaches. We first explain how existing MIAs can be unified via a common thread in Section III-A, and then discuss how we build HAMP to overcome this exploitation.

A. Overconfident Prediction Leads to Membership Leakage

While existing MIAs employ diverse approaches to infer membership, we unify them by viewing them all as exploiting the model's overconfidence in predicting training samples. We explain below how different attacks can be viewed as different forms to quantify whether a model is overly confident in predicting a specific sample, in order to infer its membership.

Score-based MIAs leverage the prediction scores to infer membership through different proxies. The model's overconfident prediction on training samples can be exposed through high confidence scores [49], low prediction entropy [38], [39], low prediction loss [49], or using a neural network [38], [31]. For boundary and augmentation attacks, samples predicted with high confidence can be viewed as exhibiting high robustness against adversarial perturbations and data augmentation. Training samples can therefore be identified by the adversary based on whether they are more resilient to adversarial perturbation [9], [28] or data augmentation [9].

What leads to the model's overconfidence in predicting training samples? As mentioned before, common training algorithms make use of the one-hot hard labels to minimize the prediction loss. Minimizing the training objective function (1) is equivalent to encouraging the model to produce outputs that are consistent with the labels, i.e., 100% for the ground-truth class and 0% for any other classes.

While training with hard labels has achieved success in a broad class of classification problems, we find that it undesirably contributes to the model's overconfidence in predicting training samples, which eventually leads to membership leakage. For example, on Purchase100, the difference between the average prediction confidence on training and testing samples is >25%, which means the model is much more confident in predicting training samples. Such differential behavior can be identified by the adversary to obtain >14% attack TPR @0.1% FPR. This indicates training with one-hot hard labels undesirably enables the adversary to identify a large fraction of training samples with very low false positives (and similarly identifying testing samples with low false negatives). This inspires our design principle of enforcing less confident prediction to mitigate MIAs, based on which we introduce a novel training and testing defense that can achieve both strong membership privacy and high model accuracy.

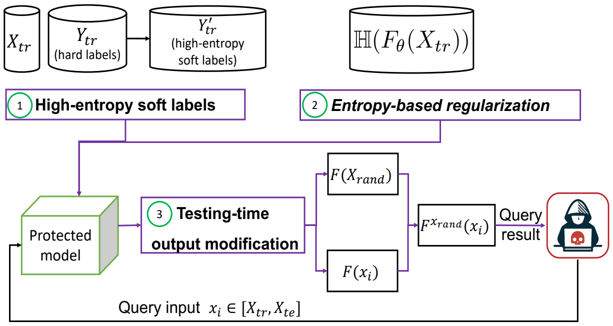
B. Overview

Fig. 2 shows an overview of HAMP. It has two parts.

Training-time defense. Inspired by the observation in Section III-A, our main idea is to *train the model to produce less confident prediction even on training samples*, thereby enforcing the model to behave similarly on training and testing samples. We achieve this by two innovations: (1) replacing the hard labels with *high-entropy soft labels*; and (2) introducing an *entropy-based regularizer*.

The first step is to generate soft labels with high entropy from the hard labels. These high-entropy soft labels explicitly induce the model to produce less confident output during training by assigning a much lower probability for the ground-truth class. For instance, a hard label of [0, 1] can be changed into a soft label of [0.4, 0.6], which guides the model to predict the ground-truth class with 60% probability (instead of 100%). The probability of each class is determined by an *entropy threshold* parameter, and a higher threshold generates a soft label with higher entropy (e.g., [0.5, 0.5] has the highest entropy) - details in the next section. The ground-truth class remains the same so that the model can learn correctly, e.g., a dog image is still trained to be predicted as a dog.

Second, we introduce an entropy-based regularizer to penalize the model for predicting any output with low entropy. Prediction entropy measures the prediction uncertainty, and can be used to regularize the confidence level of the prediction, e.g., low entropy indicates high-confidence output, and can be mitigated by the proposed regularizer to become low-confidence output.



X_{tr}
Y_{tr} (hard labels)
Y'_{tr} (high-entropy soft labels)
$\mathbb{H}(F_\theta(X_{tr}))$
1 High-entropy soft labels
2 *Entropy-based regularization*
Protected model
3 Testing-time output modification
$F(X_{rand})$
$F(x_i)$
$F^{x_{rand}}(x_i)$
Query result
Query input $x_i \in [X_{tr}, X_{te}]$

Fig. 2: Overview of our training- and testing-time defense.

The high-entropy soft labels encourages the model to produce outputs consistent with the labels, while the regularization term allows the model to produce any low-confidence outputs, even if the outputs do not closely match the labels. Both components are important for HAMP to mitigate overconfident prediction and achieve strong membership privacy.

How HAMP's training-time defense mitigates membership leakage from different sources? There are two sources leading to membership leakage, and we discuss below how HAMP can reduce leakage from both sources.

Output scores. With the high-entropy soft labels and entropy-based regularizer, HAMP forces the model to produce output scores on training samples with higher entropy (i.e., lower confidence), which resemble the output scores on testing samples. E.g., on Purchase100, the average prediction entropy on members and non-members are 0.389 and 0.576 on the undefended model, which are 4.485 and 4.490 on the HAMP model. HAMP therefore reduces the entropy difference by 31x (from 0.187 to 0.006) and effectively enforces the output scores on members and non-members to be indistinguishable (more details in Appendix A-B). Some score-based MIAs leverage both output scores *and* label information (e.g., [39], [31]) and we explain next how HAMP prevents membership leakage from the labels.

Prediction labels. HAMP's training-time defense mitigates privacy leakage from the prediction labels by pushing the training samples closer to the decision boundary, so that training samples lie *similarly close* to the decision boundary as the testing samples. We next use the boundary and augmentation attacks to explain (both attacks exploit label information in different manners to infer membership).

Boundary attack exploits the training samples' higher adversarial robustness than testing samples. Without HAMP, the adversary can discern that the training samples require more perturbations than the testing samples. With HAMP however, training samples are predicted with lower confidence, and therefore it takes a similar amount of perturbation to perturb training and testing samples. For instance, on CIFAR100, the average amount of perturbation to perturb the training samples on the undefended model is 0.342, and 0.226 on the testing samples. With HAMP, the perturbation on the training samples become 0.289 and 0.234 on the testing samples, which effectively reduces the perturbation difference between training and testing samples by >53%. This means the members and non-members become indistinguishable from the perspective of their adversarial robustness.

Augmentation attack exploits the training samples' higher resistance to data augmentation, i.e., the augmented variants of training samples are *more likely* to be classified correctly. Performing data augmentation on the original samples can be viewed as drawing neighboring variants around the original samples in the sample space. Since the training samples are closer to the decision boundary under HAMP, their augmented variants are more likely to cross the decision boundary, and hence be classified *incorrectly*, which is similar to how testing samples would behave. We also evaluate the model's performance on the inputs added with random augmentations. We find HAMP mainly reduces the performance on the augmented training samples (from 64.38% to 55.12%), and the performance on the augmented testing samples remain similar before and after HAMP (46.12% and 46.36%). This reduces the accuracy difference between members and non-members from 18.26% to 8.76% (a 52% reduction), and enables them to exhibit similar resistance to data augmentation.

HAMP's training-time framework is able to reduce the model's overconfident prediction on training samples *without* compromising the model's performance, i.e., strong membership privacy and prediction accuracy. Nevertheless, membership privacy can be further improved such as by pushing the training samples closer to the decision boundary, but at the cost of accuracy, which is undesirable. In light of this, we introduce a testing-time output modification defense that can attain higher membership privacy *without* degrading accuracy.

Testing-time defense. Our idea is to modify all the output scores to become low-confidence scores, hence making the output scores from members and non-members less distinguishable. The key observation that underpins the testing-time defense is that *randomly-generated samples are often predicted with low confidence, and the low-confidence output scores can be used for output modification.* Specifically, we first uniformly generate random samples, which are highly unlikely to be part of the training set due to the high dimensionality of the input space (e.g., the entire Texas100 dataset contains only $67,330$ samples while the input space has 2^{6170} samples). As these random samples are unlikely to be members of the training set, they are often predicted by the model with low confidence. We then replace all the entries in each output score with those from random samples, where the replacement is to keep the predicted labels unchanged (all top-*k* labels) and modify the output scores only. In essence, HAMP returns only the ordering of the confidence scores and the ordering is represented by the random output scores arranged in a specific order.

The random samples *do not* have any prerequisites (e.g., they do not need to come from a specific distribution, nor do they need to produce a specific prediction label), as long as they are valid inputs (e.g., pixel values are in [0, 255]).

In HAMP, the high-confidence outputs on members and low-confidence outputs on non-members, all become low-confidence outputs after being modified. This significantly increases the difficulty for the adversary to identify differential behaviors on members and non-members.

In Section V-A, we perform detailed ablation study to show that all three defense components in HAMP are crucial in achieving strong membership privacy and preserving high model accuracy. We next explain HAMP in details.

C. Training-time Defense

Generating high-entropy soft labels. The first step is to generate high-entropy soft labels for training, where the class probabilities in the soft labels are controlled by an *entropy threshold* parameter, denoted as γ. The entropy of a soft label y' can be calculated as:

$$\mathbb{H}(y') = -\sum_{j=0}^{k-1} y'_j * \log(y'_j) \tag{3}$$

A soft label with uniform probability on each dimension has the highest entropy, based on which we choose a smaller entropy threshold. For a k-class classification problem, our goal is to find a y' given γ such that,

$$\mathbb{H}(y') \geq \gamma \mathbb{H}(y), y = \{\frac{1}{k}, ... \frac{1}{k}\}^k, \gamma \in [0, 1], \tag{4}$$

where y' has the highest probability on its ground-truth class, and the probabilities on the remaining dimension are the same. For a hard label y whose ground-truth class is j_{truth} (k classes in total), the resulting soft label becomes:

$$\forall y'_j \in y', y'_j = \begin{cases} p & \text{if } j = j_{truth} \\ (1-p)/(k-1) & \text{if } j \neq j_{truth} \end{cases} \tag{5}$$

p is the probability on the ground-truth class, and a larger γ indicates higher prediction entropy, which leads to a smaller p (i.e., smaller probability on the ground-truth class).

Entropy-based regularization. In addition, we introduce an entropy-based regularizer that measures the prediction entropy during training, and penalizes predictions that have low entropy, as such predictions indicate high-confidence output and may be exploited by the adversary.

Finally, the overall training objective can be formulated as:

$$\mathcal{L}_{\text{KL}}(F_\theta(x), y) = \sum_{j=0}^{k-1} y_j \log(\frac{y_j}{F_\theta(x)_j}), \tag{6}$$

$$\min_\theta [\mathcal{L}_{\text{KL}}((F_\theta(X_{tr}), Y'_{tr}), \theta) - \alpha \mathbb{H}(F_\theta(X_{tr}))], \tag{7}$$

where Y'_{tr} is the high-entropy soft labels, L_{KL} the Kullback-Leibler divergence loss, α is to control the strength of regularization. Our goal is to train the model to mitigate the overconfident prediction on training samples while maintaining high prediction accuracy. We achieve this by using a large γ to train the model with soft labels in high entropy, and a α to regularize the prediction entropy. Section IV-A explains how to select the parameters γ, α in HAMP (p in Equation 5 is determined by γ).

D. Testing-time Defense

The testing-time defense uniformly modifies the runtime outputs to achieve stronger privacy without jeopardizing accuracy. We first generate uniform random samples x_{rand}, e.g., for Purchase100 with binary features, each feature is assigned with 0 or 1 with equal probability. For each runtime input $x \in [D_{tr}, D_{te}]$, all the entries in $F(x)$ (that indicate the

Algorithm 1 Training and testing phase of HAMP

```
Input: $(X_{tr}, Y_{tr}) \in D_{tr}$: Training set;
       $\gamma$: Entropy threshold;
       $\alpha$: Strength of regularization;
       $F$: an initialized ML model;

 1: function TRAINING($(X_{tr}, Y_{tr})$, $\gamma$, $\alpha$, $F$)
 2:     Generate high-entropy soft labels $y'$ given $\gamma$
 3:     Generate $Y'_{tr}$ from $Y_{tr}$ using $y'$, where $\forall(Y_{tr}[i], Y'_{tr}[i]) \in$
        $(Y_{tr}, Y'_{tr})$, argmax$(Y_{tr}[i])$ = argmax$(Y'_{tr}[i])$
 4:     for number of training epochs do
 5:         Minimize (7) using Stochastic Gradient Descent
 6:     end for
 7:     return $F$
 8: end function
 9:
10: function TESTING($F$, $x$)
11:     Generate $F(x)$
12:     Generate random uniform sample $x_{rand}$ and $F(x_{rand})$
13:     Generate $F^{x_{rand}}(x)$ by replacing $F(x)$ with $F(x_{rand})$, where
        argsort$(F^{x_{rand}}(x))$ = argsort$(F(x))$ /* top-$k$ labels unchanged */
14:     return $F^{x_{rand}}(x)$
15: end function
```

probability for each class) are replaced by those in $F(x_{rand})$, the resulting output is denoted as $F^{x_{rand}}(x)$. The replacement is to only modify the entries in $F(x)$ while ensuring $F(x)$ and $F^{x_{rand}}(x)$ give the same prediction labels. For example, let $x \in [D_{tr}, D_{te}], F(x) = [0.85, 0.05, 0.1]$, and $x' \in X_{rand}, F(x') = [0.2, 0.3, 0.5]$, then the final output produced by the model becomes: $F(x_i) = [0.5, 0.2, 0.3]$. This enforces the model to produce low-confidence outputs on both members and non-members, and reduces privacy leakage.

Overall Algorithm. Algorithm 1 gives the overall algorithm of HAMP. γ and α are the two parameters in HAMP to regulate the confidence level of the model's prediction, e.g., a high entropy threshold or strong regularization can enforce the model to become less confident in prediction. Line 2 generates a template of high-entropy soft labels of y', which is then used to generate soft labels for each of the hard labels. The condition in Line 3 ensures that the ground-truth labels remains unchanged so that the model can learn the correct labels.

At test time, each output is replaced by those from a random sample. The condition of argsort$(F^{x_{rand}}(x))$ = argsort$(F(x))$ in line 13 is to ensure both $F^{x_{rand}}(x)$ and $F(x)$ give the same labels (all top-*k* labels and not just the top-1 label). Line 11 and Line 12 are independent of each other, and hence can be executed independently to facilitate faster runtime inference (overhead evaluation in Appendix A-F).

IV. Evaluation

A. Experimental Setup

Datasets. We consider five common benchmark datasets.

Purchase100 [38] includes 197,324 shopping records of customers, each with 600 binary features indicating whether a specific item is purchased. The goal is to predict the customer's shopping habits (100 different classes in total).

Texas100 [38] contains 67,330 hospital discharge records, each containing 6,170 binary features indicating whether the patient has a particular symptom or not. The data is divided into 100 classes, and the goal is to predict the treatment given the patient's symptoms.

Location30 [38] contains the location "check-in" records of different individuals. It has 5,010 data records with 446 binary features, each of which corresponds to a certain loation type and indicates whether the individual has visited that particular location. The goal is to predict the user's geosocial type (30 classes in total).

CIFAR100 [24] is an image classification dataset that has 60,000 images (32×32×3) in 100 object classes.

CIFAR10 [24] is similar to CIFAR100 that also contains 60,000 images but with 10 different object classes.

We follow [37] to use the fully-connected (FC) networks on Purchase100, Texas100 and Location30, and a DenseNet-12 [17] on CIFAR100 and CIFAR10 (Appendix A-H conducts evaluation on more network architectures, including ResNet-18 [15], MobileNet [16] and ShuffleNet [51]). Purchase100 is trained with 20,000 samples, Texas100 with 15,000 samples, Location30 with 1,500 samples, CIFAR100 and CIFAR10 are with 25,000 samples. Section V-B reports additional experiments on more training sizes (from 2,500 to 50,000).

Attacks. We consider all nine attacks as in Section II-C. For NN-based attack, we use the black-box NSH attack from Nasr et al. [31], which uses the model loss, logit values from the target model, and the ground-truth label to train an attack inference model. We consider the loss-based attack from Yeom et al. [49] and confidence-, entropy- and modified-entropy-based attacks as in Song et al. [39]. For LiRA [3], we train 128 shadow models for each defense (64 IN and OUT models each), where each shadow model is trained following the *same* procedure as the targeted defense (as per our threat model). E.g., for HAMP, this means the shadow model is trained with the same high-entropy soft labels and the entropy-based regularization as the defense model, and the shadow model also performs the same output modification as HAMP does.

We consider the boundary and augmentation attacks from Choquette et al. [9]. For the boundary attack on the two image datasets, we use the CW2 attack [5] to generate adversarial samples and derive the perturbation magnitude threshold to distinguish members and non-members. Likewise, for the other three non-image datasets that contain binary features, we compute the sample's robustness to random noise instead of adversarial perturbation. For each sample x, we generate hundreds of noisy variants of x, and the number of correctly classified noisy variants of x is used to determine a threshold that best distinguishes between members and non-members. For augmentation attack, we consider image translation as the augmentation method, and we similarly consider different degrees of translation to find the best attack.

HAMP configuration. γ, α are the two parameters in configuring HAMP (for generating high-entropy soft labels and controlling the strength of regularization respectively). We perform grid search to select the parameters ($\gamma \in [0.5, 0.99], \alpha \in [0.0001, 0.5]$), and select the one with small train-validation gap and high validation accuracy. We also conduct evaluation to study how HAMP's performance varies under different parameters (please see Appendix A-E).

For the testing-time defense, we generate random samples (e.g., random pixels in [0, 255]) and perform output modification as in Section III-D. There are no any other requirements.

Related defenses. We consider seven major defenses: AdvReg [30], MemGuard [20], DMP [37], SELENA [42], Early stopping [39], [6], Label Smoothing (LS) [41] and DP-SGD [2]. We follow the original work to set up the defenses unless otherwise stated (more details in Appendix A-A).

Evaluation metrics. An ideal privacy defense should provide strong protection for both members and non-members, for which we follow the best practice [3] to consider (1) *attack true positive rate* (TPR) evaluated at 0.1% false positive rate (FPR), which evaluates the protection for members, and (2) *attack true negative rate* (TNR) at 0.1% false negative rate (FNR), which quantifies the protection for non-members.

Result organization. Table I reports the model accuracy for every defense. Fig. 3 compares each defense in terms of their membership privacy and model utility. Each defense is evaluated with multiple attacks, and we report the ones that achieve the highest attack TPR or TNR (detailed results for each attack are in Appendix A-K). Fig. 4 presents the average attack AUC (area under curve) by each defense, and the full ROC curves are in Appendix A-J. We leave the comparison with early stopping in Appendix A-D due to space constraint. Section V-A presents an ablation study, and Appendix A-F reports training and inference overhead evaluation. We next discuss the results by comparing HAMP with other defenses.

B. Comparison with Undefended Models

HAMP significantly reduces the MIA risk against both members and non-members. Compared with the undefended models, HAMP achieves significantly lower attack TPR and TNR. The average attack TPR on the undefended model is 13.48%, which is reduced to 0.8% by HAMP (a 94.1% reduction). Similarly, HAMP reduces the attack TNR by 97%, from 19.89% to 0.59%. This effectively thwarts the adversary in inferring members or non-members from the target model.

In addition, we find that NN-based attack yields the highest attack TPR on the undefended models in many cases (as in Fig. 3), and we explain the reason in Appendix A-G.

HAMP achieves strong membership privacy while preserving high model accuracy. Across the five diverse datasets, HAMP is able to consistently produce models with comparable accuracy as the undefended models. HAMP has an accuracy drop of at most 1.1% (on Location30), and the average accuracy drop by HAMP is only 0.46%.

C. Comparison with MemGuard [20]

Both MemGuard and HAMP are capable of preserving model accuracy. MemGuard does not incur any accuracy drop since it is a post-processing technique, and does not change the prediction label. Likewise, HAMP only incurs a minor accuracy drop of 0.46%.

HAMP achieves considerably stronger membership privacy than MemGuard. MemGuard offers very limited privacy protection because MemGuard only modifies the output scores without changing the prediction labels, which cannot prevent privacy leakage from the label information. On the contrary, HAMP consists of a training-time defense that can mitigate membership leakage from both output scores and label information (explained in Section III-B), and achieves much stronger membership privacy than MemGuard. The average attack TPR on MemGuard is 6.7%, which is 8.4x relative to that of HAMP. Similarly, the attack TNR by MemGuard is 10.9%, which is 18.3x relative to that of HAMP.

TABLE I: Model accuracy for each defense. Accuracy delta measures the accuracy difference with the undefended model.

Dataset	Defense	Training acy	Testing Acy	Acy delta
Purchase100	Undefended	99.36	80.85	0.00
	MemGuard	99.36	80.85	0.00
	AdvReg	93.97	75.75	-5.10
	DPSGD	61.06	54.05	-26.80
	LS	99.54	85.60	+4.75
	SELENA	85.19	76.50	-4.35
	HAMP	91.12	81.15	+0.30
CIFAR100	Undefended	86.21	59.56	0.00
	MemGuard	86.21	59.56	0.00
	AdvReg	55.78	44.36	-15.20
	DMP	53.37	47.52	-12.04
	LS	88.80	63.24	+3.68
	SELENA	62.15	57.64	-1.92
	HAMP	68.44	58.92	-0.64
Location30	Undefended	99.56	57.40	0.00
	MemGuard	99.56	57.40	0.00
	AdvReg	69.70	48.20	-9.20
	DPSGD	36.37	28.00	-29.40
	DMP	92.81	54.30	-3.10
	SELENA	67.41	55.80	-1.60
	HAMP	78.22	56.30	-1.10
CIFAR10	Undefended	98.72	86.72	0.00
	MemGuard	98.72	86.72	0.00
	AdvReg	86.73	82.16	-4.56
	DMP	91.08	85.56	-1.16
	SELENA	86.86	84.52	-2.20
	HAMP	95.88	86.28	-0.44
Texas100	Undefended	76.79	54.80	0.00
	MemGuard	76.79	54.80	0.00
	AdvReg	62.76	51.60	-3.20
	DPSGD	43.08	39.47	-15.33
	DMP	46.92	43.07	-11.73
	LS	75.52	56.33	+1.53
	SELENA	58.58	53.60	-1.20
	HAMP	68.56	54.40	-0.40
Average accuracy delta	Undefended	0.00	MemGuard	0.00
	AdvReg	-7.45	DPSGD	-23.84
	LS	+2.42	DMP	-7.01
	SELENA	-2.25	HAMP	-0.46

D. Comparison with AdvReg [30]

HAMP outperforms AdvReg with higher model accuracy and stronger membership privacy. In terms of accuracy, HAMP consistently achieves higher accuracy than AdvReg. AdvReg incurs an average 7.45% accuracy drop, while HAMP incurs only 0.46% (94% lower than AdvReg).

In terms of privacy, HAMP outperforms AdvReg with both much lower attack TPR and TNR. The attack TPR is 1.70% with AdvReg and 0.8% with HAMP, which translate to a 87% and 94% reduction from those of the undefended models. Similarly, AdvReg reduces the attack TNR by 90% while HAMP reduces it by 97%, which is much higher.

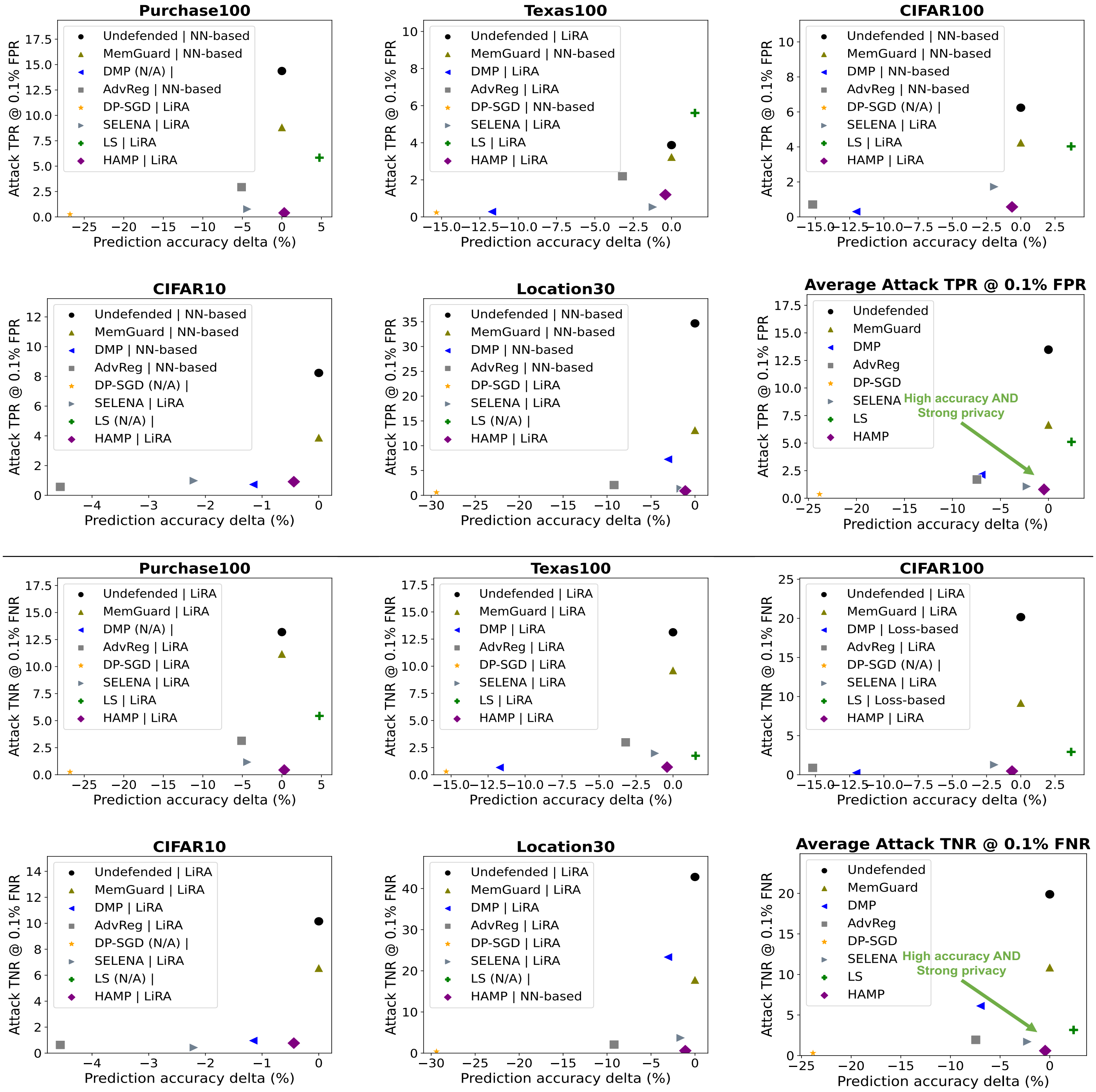


Fig. 3: **Attack TPR @ 0.1% FPR** (first two rows) and **Attack TNR @ 0.1% FNR** (last two rows) on different datasets. The legend indicates the attack that yields the highest attack TPR/TNR. Negative prediction accuracy delta means accuracy drop compared with the undefended models. DP-SGD is reported at $\epsilon = 4$, and it is not evaluated on CIFAR100 and CIFAR10 due to its significant accuracy drop (similar case as DMP on Purchase100). LS is not evaluated on CIFAR10 and Location30 as LS did not lead to accuracy improvement. *Overall, HAMP offers strong privacy protection for both members and non-members, while preserving high model accuracy, thereby yielding a superior privacy-utility trade off over other defenses.*

E. Comparison with DMP [37]

DMP [37] uses generative adversarial networks (GANs) trained on the private dataset to produce synthetic data as the reference set for knowledge distilation. We follow Shejwalker et al. [37] to train the two image datasets on DC-GAN [35]. The defender can generate unlimited data from the GAN, and hence he/she can create a reference set that is larger than the original training set. Therefore, we use 150K synthetic samples to train the model with higher accuracy (we do not consider more synthetic images as the improvement is negligible).

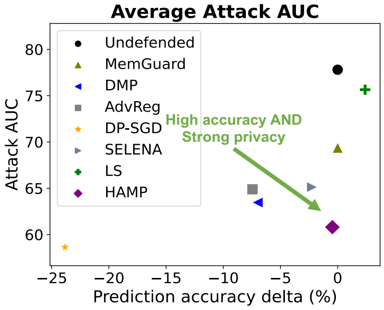


Fig. 4: Average attack AUC by each defense (detailed results for each dataset can be found in Appendix A-I).

For the three datasets with binary features, we use CT-GAN [46] for modeling tabular data. We use 100K synthetic samples for Texas100, 10k for Location30. We do not consider Purchase100 as it incurs significant accuracy drop (over 30%). To validate that synthetic samples are useful for the domain task, we compare the performance of the models trained with GAN-generated synthetic data and those with random data (i.e., all features are randomly selected as 0 or 1 with equal probability) using Texas100. We find that models trained with random data only achieve accuracy from 5.8% to 14.8%; while those with GAN-generated data achieve over 40% accuracy.

HAMP outperforms DMP by being able to consistently achieve strong privacy protection with high model accuracy across different datasets. In terms of membership privacy, we find that DMP is able to achieve strong results in many (but not all) cases, and it achieves an average attack TPR of 0.44% and TNR of 0.38% on Texas100, CIFAR100 and CIFAR10, where HAMP achieves 0.9% TPR and 0.65% TNR (DMP is slightly better). However, DMP's performance does not generalize across datasets. For instance, on Location30, DMP suffers from a much higher attack TPR of 7.26% and TNR of 23.33%. This is because the model is trained with limited data (1,500), and the GAN is *not* able to generate diverse data that are different from the original training data. As a result, the teacher model assigns high confidence to the synthetic data, from which the student model learns to predict the training members with high confidence that eventually leads to high MIA risk. To validate this, we compare the difference between the prediction confidence on members and non-members by the DMP models. On Location30, the average difference is $>$30%, and only $<$5% on the other datasets, which is why DMP exhibits poor privacy protection on Location30. On the same dataset, HAMP yields a low TPR of 0.89% and TNR of 0.59%, and this trend is consistent across datasets.

In terms of accuracy, DMP suffers from different degrees of accuracy loss that are much higher than HAMP's. DMP incurs $>$30% accuracy loss on Purchase100 (as mentioned earlier), $\sim$12% accuracy drop on Texas100 and CIFAR100, 3.1% on Location30, and 1.2% on CIFAR10 (smaller accuracy loss as CIFAR10 has 10 classes only). In contrast, HAMP incurs average accuracy drop of $<$0.5% (at most 1.1%), which is significantly better than DMP.

F. Comparison with SELENA [42]

Both SELENA and HAMP achieve similarly strong privacy protection. On average, HAMP has a slightly better membership privacy than SELENA, but neither technique has consistently better membership privacy overall (Fig. 3). The attack TPR of SELENA is $0.53\% \sim 1.72\%$, with an average of 1.1%, and that of HAMP is $0.4\% \sim 1.2\%$, with an average of 0.8%. They are able to reduce the attack TPR by 92% (SELENA) and by 94% (HAMP). In addition, the attack TNR of SELENA is $0.42\% \sim 3.7\%$, with an average of 1.7%, and that of HAMP is $0.44\% \sim 0.77\%$, with an average of 0.6%. This translates to a TNR reduction of 91% (SELENA) and 97% (HAMP), respectively.

While providing comparable privacy benefits, HAMP outperforms SELENA by having lower accuracy loss, hence providing a better privacy-utility trade off. The largest accuracy drop by SELENA is 4.4% and that by HAMP is only 1.1%. On average, SELENA incurs a 2.25% accuracy drop, while HAMP incurs a much smaller drop of 0.46%. Moreover, our additional experiment in Section V-B shows that HAMP continues to outperform SELENA with much lower accuracy drop when evaluated on a variety of different training sizes (2.2%$\sim$5.2% by SELENA and 0.04%$\sim$0.98% by HAMP).

G. Comparison with Label Smoothing (LS) [41]

Though LS is able to improve model accuracy, the model trained with LS still suffers from *high* MIA risk. In contrast, the model trained with HAMP can maintain high model accuracy and exhibit very *low* MIA risk. For LS, we follow prior work by Kaya et al. [21] to train with different smoothing intensities from 0.01 to 0.995, and select the model with the highest accuracy (we omit CIFAR10 and Location30 as LS did not lead to accuracy improvement). We first discuss the qualitative difference between LS and HAMP, and then quantitatively compare their privacy risk.

While LS and HAMP use soft labels in their training, they are built with different purposes that require different soft labels. LS is used as a regularization technique to improve model accuracy, which necessitates training with *low*-entropy soft labels, and is able to increase the accuracy by 2.4% on average. However, the resulting model still suffers from high MIA risk, as LS causes the model to overfit on the smooth labels and exhibit discernible behavior on the training samples [21]. In contrast, HAMP is built to improve membership privacy, which consists of *high*-entropy soft labels, an entropy-based regularizer and a novel testing-timd defense to force the model to make less confident predictions, and to behave similarly on the training and testing samples.

To quantitatively compare the different soft labels used by both techniques, we measure the soft label entropy in LS and HAMP, and find that the label entropy in HAMP is considerably higher than that in LS, and is 4x$\sim$50x relative to that in LS (average 9x). This contributes to the low membership privacy risk by HAMP, unlike LS.

The average attack TPR on the LS models is 5.1%, 7.1x relative to that by HAMP (on the same datasets). The attack TNR on LS is 6.3x relative to that by HAMP (we observe a similar trend even when we train LS with other smoothing intensities that have comparable accuracy improvement - see Appendix A-C). Moreover, our results reveal that LS may amplify the MIA risk and render the model *more vulnerable* than the undefended model. On Texas100, LS increases the attack

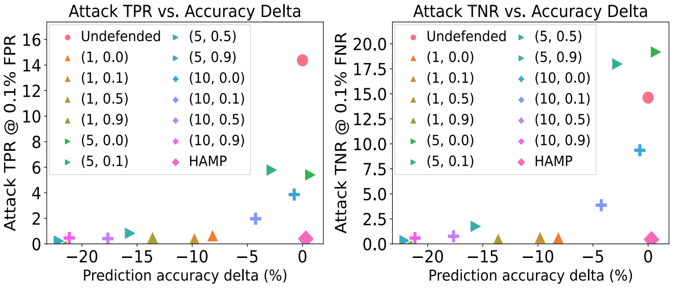


Fig. 5: Results on DP-SGD under different clipping norms $\in [1, 5, 10]$, and noise_multipliers $\in [0.0, 0.1, 0.5, 0.9]$.

TPR from 3.87% (on the undefended model) to 5.61%, which increases the MIA risk against training members by 45%. This suggests that LS may constitute a hidden privacy risk for the practitioners (a similar finding was identified recently by Kaya et al. [21]). On the contrary, HAMP consistently leads to low MIA risk and outperforms LS with significantly better membership privacy.

H. Comparison with DP-SGD [2]

We use the canonical implementation of DP-SGD using Pytorch Opacus [1]. We first consider a fixed privacy budget $\epsilon = 4$ as per Tang et al. [42], and then evaluate DP-SGD with different values of ϵ.

1) DP-SGD with fixed $\epsilon = 4$.: In this setting, the average attack TPR of the DP-SGD models is 0.36% and 0.3%, both of which are the lowest among all the defenses we evaluated. In comparison, HAMP yields 0.8% attack TPR and 0.6% TNR, which are slightly higher than DP-SGD. However, DP-SGD suffers from considerable accuracy loss, with an average loss of 23.84%, while HAMP a significantly smaller loss of 0.46%.

2) DP-SGD with different ϵ.: We next evaluate DP-SGD by considering different noise_multipliers and clipping norms. We consider Purchase100, on which we used a noise_multiplier of 1.7 and a clipping norm of 1, for $\epsilon = 4$ in the earlier evaluation. We select different noise_multiplier values of 0.0 (no noise injected), 0.1 ($\epsilon = 12069.1$), 0.5 ($\epsilon = 62.5$) and 0.9 ($\epsilon = 10.9$); and clipping norm values of 1, 5 and 10, totalling 12 different configurations. We report the results in Fig. 5.

Reducing the amount of injected noise and using a larger clipping norm allows DP-SGD to provide empirical privacy protection (but with a very large provable bound of ϵ), and reduce the amount of accuracy loss. For instance, by using a clipping norm of 10 *without* injecting any noise, DP-SGD is able to reduce the accuracy loss to be <1%, which can also reduce the attack TPR by 73% (from 14.37% to 3.86%), and the attack TNR by 36% (from 14.62% to 9.36%). Nevertheless, this performance is still considerably inferior to that of HAMP, which can reduce the attack TPR and TNR by 97.2% and 96.7%, respectively.

Using a tighter clipping norm or injecting more noise can improve the membership privacy even more, but this comes at the cost of accuracy loss (the earlier result has negligible accuracy loss). For example, by using a small clipping norm of 1, the attack TPR can be reduced to 0.67% and attack TNR to 0.62%. However, this results in 8.2% accuracy loss. Increasing the noise_multiplier can further reduce privacy leakage, e.g., using a noise_multiplier value of 0.5 can reduce the attack TPR to 0.5% and attack TNR to 0.49% (and with a large ϵ of 62.5), which are comparable to the 0.4% TPR and 0.44% TNR values by HAMP. However, DP-SGD degrades the accuracy by 13.6%, while HAMP incurs negligible accuracy drop.

Therefore, training a model with a small amount of noise or with a tight clipping norm is also a viable defense against MIAs, though it still incurs much larger accuracy loss than HAMP and results in large provable bounds ϵ.

TABLE II: Ablation study on different components of HAMP: ①: High-entropy soft labels; ②: Entropy-based regularizer; ③: Testing-time output modification.

Defense component	Training accuracy	Testing accuracy	Attack TPR @0.1% FPR	Attack TNR @0.1% FNR
None (undefended)	99.36	80.85	14.37	14.62
①	94.58	81.75	4.76	4.22
②	98.06	81.10	3.39	4.19
③	99.36	80.85	8.51	5.34
① + ②	91.12	81.15	1.86	1.07
① + ③	94.58	81.75	0.82	1.23
② + ③	98.06	81.10	2.90	3.76
① + ② + ③ (full defense)	91.12	81.15	0.40	0.44

V. Discussion

A. Ablation Study

HAMP consists of three components, and we perform a detailed ablation study to investigate the effectiveness of each of these components - this includes a total of six configurations. We present the results in Table II.

The second to fourth rows in Table II shows the results on models using a single component in HAMP. For instance, training with high-entropy soft labels alone is able to produce a model with similar accuracy as the undefended model (trained with the one-hot hard labels), and reduce the attack TPR from 14.37% to 4.76%, and attack TNR from 14.62% to 4.22%. This also validates our earlier observation in Section III-A that training with one-hot hard labels could lead to high MIA risk, and the proposed high-entropy soft labels can be used to mitigate the high MIA risk. However, this is not enough as the model still suffers from relatively high TPR and TNR. We observe similar trends in the other two settings where we either train with the entropy-based regularizer alone, or directly perform output modification on the undefended model.

Strengthening the model with more defense components can further reduce the MIA risk while preserving model accuracy. For example, training with high-entropy soft labels and the entropy-based regularizer (fifth row in Table II) achieves a low TPR of 1.86% and a low TNR of 1.07%. We observe a similar trend even if we change to different configurations, as in the sixth and seventh rows in Table II, both of which exhibit better privacy protection than models equipped with a single component. Furthermore, we find that the resulting model continues to maintain high model accuracy, which

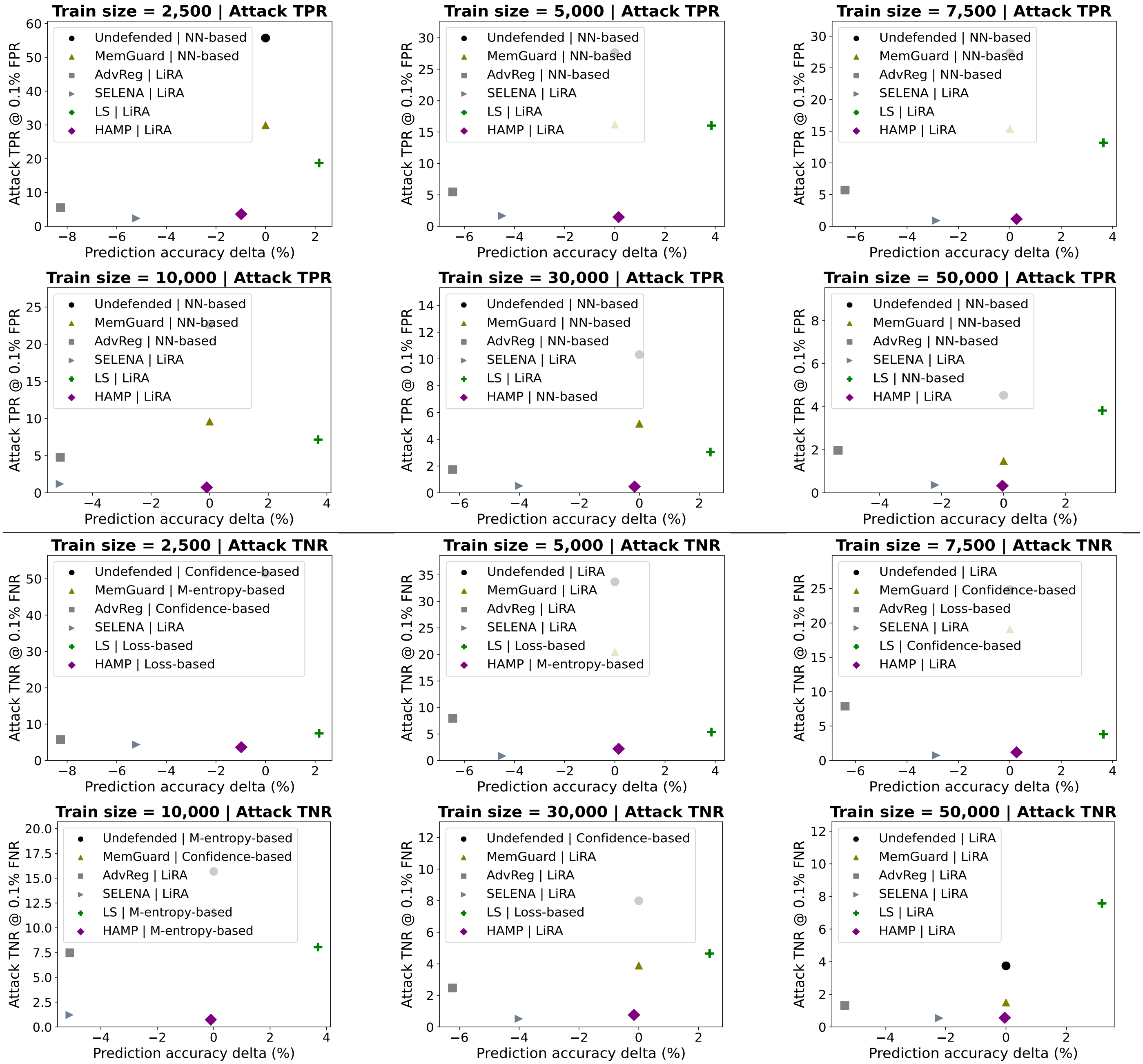


Fig. 6: Defense evaluation on models trained with different amounts of training data. The first two rows evaluate attack TPR and the last two rows evaluate attack TNR. HAMP consistently achieves strong privacy protection while preserving model accuracy.

means the different defense components in HAMP can be used together to improve membership privacy without jeopardizing model accuracy. Finally, the full defense consisting of all three defense components, as in HAMP, exhibits the best privacy protection while maintaining competitive model accuracy.

B. Evaluation on Different Training Sizes

This section reports additional experiments where we vary the size of the training set. We evaluate six more different sizes on Purchase100, which is the largest dataset in our evaluation and allows us to comprehensively evaluate a wide range of sizes, namely: 2,500, 5,000, 7,500, 10,000, 30,000, 50,000 (up to 20x difference). We trained 64 shadow models in the LiRA attack for each defense, with over 2,300 different shadow models in total. Fig. 6 shows the results.

We find that even when evaluated under a broad range of training sizes, HAMP consistently achieves superior performance on both privacy protection and model utility. The average attack TPR on the undefended model is 24.7% and attack TNR 22.9%. MemGuard achieves an average attack TPR of 13% and attack TNR 17.4%, both of which are significantly higher than the 1.3% and 1.5% by HAMP. AdvReg incurs an average accuracy loss of 6.3% while HAMP incurs only 0.2%. HAMP also outperforms AdvReg with better privacy protection: AdvReg reduces the attack TPR by 83% and attack TNR by 76.1%, while HAMP reduces them by 94.8% and 93.4%, respectively. LS improves the accuracy by 3.2%, but it still suffers from high MIA risk: its attack TPR and TNR are 8x and 4.1x relative to that of HAMP. Both SELENA and HAMP have similarly strong membership privacy: the average attack TPR on SELENA is 1.2%, and 1.3% on HAMP; the attack TNR are 1.4% and 1.5%, respectively. Under a similar privacy protection, HAMP still outperforms SELENA with a much lower accuracy drop. On average, SELENA degrades the accuracy by 3.97% (up to 5.2%), while HAMP degrades accuracy by only 0.15% (up to 0.98%).

C. Evaluation against Data-poisoning-based MIA [43]

Recent work by Tramer et al. [43] shows that a more capable adversary can significantly amplify the MIA risk through data poisoning. Therefore, we conduct additional evaluation on whether HAMP can protect against such more capable attack.

The Tramer et al. attack increases the membership leakage against target points, by poisoning the training set to transform the target points into outliers. Each target point is replicated n times with a wrong label, and these replicas are added as the poison samples. If the target point is a member in the training set, the model will be fooled into believing that the correctly-labeled target point is "mislabeled" (due to the presence of other poisoned replicas), which would have a large influence on the model's output and can be identified by the adversary.

We follow [43] to conduct the evaluation on CIFAR10, and select 250 random target points (containing both members and non-members), each replicated 8 times. We train 128 shadow models, which include a total of 32,000 target points. Without data poisoning, the adversary achieves 8.23% attack TPR and 10.15% attack TNR on the undefended model. These are increased to 52.44% and 24.52% after data poisoning, respectively. Even under such a powerful attack, HAMP is able to reduce the attack TPR from 52.44% to 0.34%, and attack TNR from 24.52% to 0.71%. Further, HAMP achieves such strong protection with a negligible accuracy drop of 0.6%.

D. Limitation

First, it requires re-training and hence incurs additional training overhead. Nevertheless, re-training is commonly required by many existing defenses [30], [37], [42], and training is a one-time effort prior to deployment. Further, our evaluation shows that HAMP incurs only a modest training overhead compared with other defenses (see Appendix A-F).

The second limitation is that HAMP's testing-time defense incurs an overhead in every inference, which may be undesirable for the computations that have stringent real-time constraints. Nevertheless, HAMP incurs a low latency of only 0.04∼0.38*ms* per inference. In comparison, MemGuard, the other defense that also contains post-processing modification, introduces a latency of 335.42∼391.75*ms*. In addition, this process also changes the output scores to be randomized scores, which may affect the usefulness of the output scores. Nevertheless, we try to reduce the impact by ensuring the prediction labels derived from the output scores remain unchanged (all top-k labels), and thus the model accuracy is unaffected. This can still provide meaningful information in the output scores without leaking membership privacy.

Finally, though HAMP empirically provides superior privacy-utility tradeoff, it does not offer provable guarantees. This is a limitation common to all practical defenses [30], [37], [42], [20]. Hence, a more capable adversary may mount stronger attacks, such as the data poisoning attack by Tramer et al. [43]. Our preliminary evaluation shows that HAMP still exhibits strong privacy protection and preserves model accuracy even under the presence of such a data-poisoning adversary, but we leave further investigation to future work.

VI. Related Work

Membership inference attacks. Depending on the adversary capabilities, MIAs can be divided into black-box [38], [49], [18], [3], [39], [9], [48], [28] and white-box attacks [26], [19], [31]. The former has access only into the output of the target model while the latter has visibility into information such as the internal model gradients to facilitate membership inference. Black-box MIA assumes a more realistic adversary, and hence is hence widely adopted in prior defense studies [20], [42], [30] (and in HAMP). Such attacks can be mounted by either shadow-training [38], [30], [49] or computing statistical metrics based on the partial knowledge of the private dataset [39], [9], [28]. Many of those attacks require full or partial access to the output scores by the model, and may be defeated if the model only reveals the prediction label. This motivates a new class of attacks called, label-only attacks, which can be launched either with [9] or without [28] partial knowledge of the membership information. Carlini et al. [3] introduce the LiRA attack that can succeed in inferring membership when controlled at low false positive or false negative, through a well-calibrated Gaussian likelihood estimate.

In addition to supervised classification, MIAs have also been explored in other domains, including contrastive learning [29], generative models [7], [14], federated learning [31], graph neural networks [52], and recommender systems [50].

Defenses against membership inference attacks. These defenses can be divided into provable and practical defenses. The former can provide rigorous privacy guarantee, such as DP-SGD [2], PATE [33]. Nevertheless, these defenses often incur severe accuracy drop when used with acceptable provable bounds [36], [34]. Another line of practical defenses aim to achieve empirical privacy without severely degrading accuracy. Common regularization techniques such as dropout [40], weight decay [25] are shown to be able to reduce privacy leakage, but with limited effectiveness [38], [37]. Other defenses enforce specific optimization constraint during training to mitigate MIAs [30], [27], or perform output obfuscation [20], [47]. Knowledge distillation is used by different techniques to mitigate MIAs, including PATE [33], DMP [37], SELENA [42] and KCD [10]. However, existing defenses are often biased towards either privacy or utility. In contrast, HAMP both achieves strong membership privacy and high accuracy, which offers a much better privacy-utility trade off.

Other privacy attacks. In addition to membership privacy, common ML models are found to leak different private properties [44], [45], [12], [11], [13], [4]. Model extraction attacks can duplicate the functionality of a proprietary model [44],

[45]. Model inversion attacks are capable of inferring critical information in the input features such as genomic information [12], [11]. Property inference attacks are constructed to infer sensitive properties of the training dataset [13].

VII. Conclusion

This work introduces HAMP, a defense against Membership Inference Attacks (MIAs) that can achieve both high accuracy and membership privacy. HAMP has two innovations: (1) a training framework that consists of high-entropy soft labels and an entropy-based regularizer; and (2) an output modification defense that uniformly modifies the runtime output. HAMP significantly constrains the model's overconfidence in predicting training samples, and forces the model to behave similarly on both members and non-members, thereby thwarting MIAs. Our evaluation shows that HAMP outperforms seven leading defenses by offering a better trade off between utility and membership privacy.

Acknowledgment

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a Four Year Fellowship and a Public Scholar Award from the University of British Columbia (UBC).

References

[1] "Pytorch opacus," https://github.com/pytorch/opacus.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[3] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, "Membership inference attacks from first principles," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1897–1914.

[4] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson *et al.*, "Extracting training data from large language models," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2633–2650.

[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57.

[6] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in neural information processing systems*, pp. 402–408, 2001.

[7] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "Gan-leaks: A taxonomy of membership inference attacks against generative models," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 343–362.

[8] Z. Chen and K. Pattabiraman, "Overconfidence is a dangerous thing: Mitigating membership inference attacks by enforcing less confident prediction," *arXiv preprint arXiv:2307.01610*, 2023.

[9] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, "Label-only membership inference attacks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1964–1974.

[10] R. Chourasia, B. Enkhtaivan, K. Ito, J. Mori, I. Teranishi, and H. Tsuchida, "Knowledge cross-distillation for membership privacy," *arXiv preprint arXiv:2111.01363*, 2021.

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

[12] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, pp. 17–32.

[13] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 619–633.

[14] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, "Logan: Membership inference attacks against generative models," in *Proceedings on Privacy Enhancing Technologies (PoPETs)*, vol. 2019, no. 1. De Gruyter, 2019, pp. 133–152.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[18] B. Hui, Y. Yang, H. Yuan, P. Burlina, N. Z. Gong, and Y. Cao, "Practical blind membership inference attack via differential comparisons," *arXiv preprint arXiv:2101.01341*, 2021.

[19] B. Jayaraman, L. Wang, K. Knipmeyer, Q. Gu, and D. Evans, "Revisiting membership inference under realistic assumptions," *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 2, 2021.

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

[21] Y. Kaya and T. Dumitras, "When does data augmentation help with membership inference attacks?" in *International Conference on Machine Learning*. PMLR, 2021, pp. 5345–5355.

[22] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard, "The megaface benchmark: 1 million faces for recognition at scale," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4873–4882.

[23] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[24] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[25] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, 1992, pp. 950–957.

[26] K. Leino and M. Fredrikson, "Stolen memories: Leveraging model memorization for calibrated white-box membership inference," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 1605–1622.

[27] J. Li, N. Li, and B. Ribeiro, "Membership inference attacks and defenses in classification models," in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 5–16.

[28] Z. Li and Y. Zhang, "Membership leakage in label-only exposures," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 880–895.

[29] H. Liu, J. Jia, W. Qu, and N. Z. Gong, "Encodermi: Membership inference against pre-trained encoders in contrastive learning," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2081–2095.

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

[31] ——, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated

learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[32] E. W. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision support systems*, vol. 50, no. 3, pp. 559–569, 2011.

[33] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[34] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson, "Tempered sigmoid activations for deep learning with differential privacy," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9312–9321.

[35] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[36] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, and Y. Wang, "Membership inference attack against differentially private deep learning model." *Trans. Data Priv.*, vol. 11, no. 1, pp. 61–79, 2018.

[37] V. Shejwalkar and A. Houmansadr, "Membership privacy for machine learning models through knowledge transfer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, pp. 9549–9557, May 2021.

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

[39] L. Song and P. Mittal, "Systematic evaluation of privacy risks of machine learning models," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[42] X. Tang, S. Mahloujifar, L. Song, V. Shejwalkar, M. Nasr, A. Houmansadr, and P. Mittal, "Mitigating membership inference attacks by {Self-Distillation} through a novel ensemble architecture," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1433–1450.

[43] F. Tramèr, R. Shokri, A. San Joaquin, H. Le, M. Jagielski, S. Hong, and N. Carlini, "Truth serum: Poisoning machine learning models to reveal their secrets," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, p. 2779–2792.

[44] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[45] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4771–4780.

[46] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[47] Z. Yang, B. Shao, B. Xuan, E.-C. Chang, and F. Zhang, "Defending model inversion and membership inference attacks via prediction purification," *arXiv preprint arXiv:2005.03915*, 2020.

[48] J. Ye, A. Maddi, S. K. Murakonda, V. Bindschaedler, and R. Shokri, "Enhanced membership inference attacks against machine learning models," *arXiv preprint arXiv:2111.09679*, 2021.

[49] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 2018, pp. 268–282.

[50] M. Zhang, Z. Ren, Z. Wang, P. Ren, Z. Chen, P. Hu, and Y. Zhang, "Membership inference attacks against recommender systems," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 864–879.

[51] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[52] Z. Zhang, M. Chen, M. Backes, Y. Shen, and Y. Zhang, "Inference attacks against graph neural networks," in *USENIX Security Symposium (USENIX Security). USENIX*, vol. 2022, 2021, p. 13.

Appendix A
Appendix

A. Details of Defense Setup

This section provides details of the defense setup in our evaluation. For each dataset, we use 10% of the training set as a separate validation set (20% for Location30 as it has a smaller training size), and select the model with the highest validation accuracy.

HAMP. The values of entropy threshold γ and α parameter (for controlling the regularizer) are given in Table III. For model training on the two image datasets, in addition to the requirement of yielding high validation accuracy, we empirically set an additional condition that the model needs to gain at least 1% improvement on validation accuracy in order to be considered the best model. This is to prevent the model gaining a marginal improvement on validation accuracy at the cost of significant overfitting on training samples, which could result in a large generalization gap.

TABLE III: Parameter setup in HAMP.

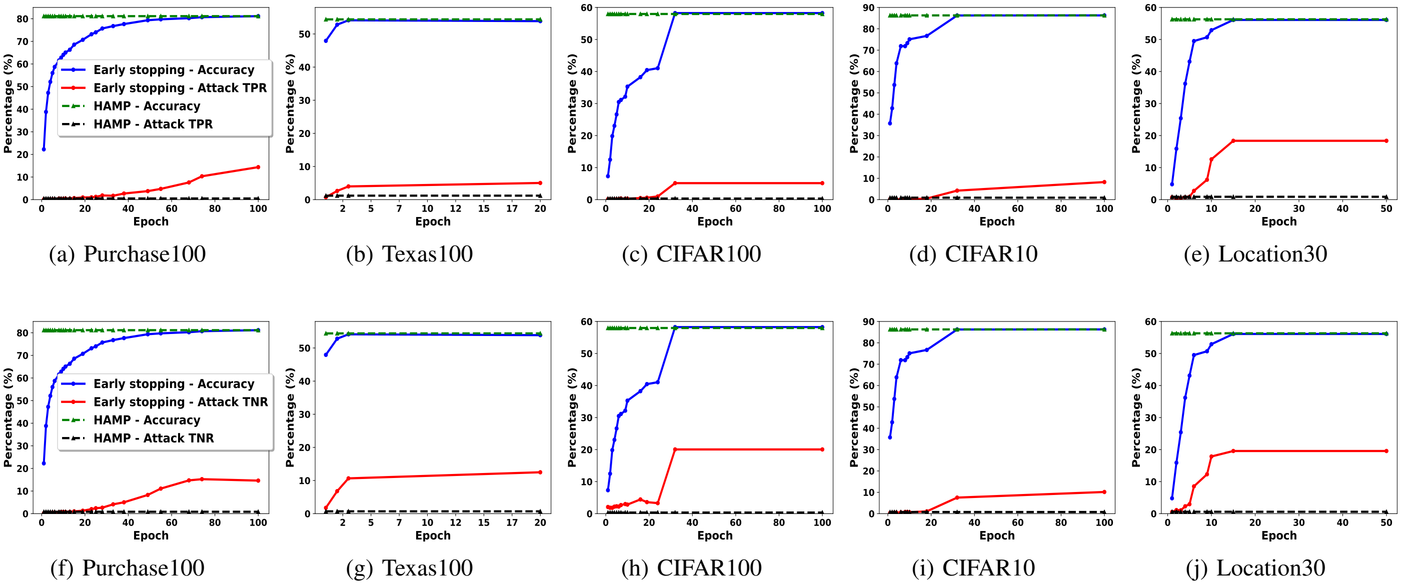
Dataset	Entropy threshold	Regularization strength
Purchase100	0.8	0.01
Texas100	0.6	0.01
CIFAR100	0.5	0.005
CIFAR10	0.95	0.001
Location30	0.5	0.001

Adversarial regularization [30]: The alpha parameter is for balancing classification accuracy and privacy protection. We set alpha as 3 for Purchase100 [30], 10 for Texas100 [39], 6 for CIFAR100 and CIFAR10 [30], and 10 for Location30.

SELENA [42]: We follow the original authors to set K=25 and L=10, where K is the total number of sub models, and L means for a given training sample, there are L sub models whose training sets do not contain that particular sample. For these L models, the given training sample can be viewed as an instance in their "reference set" for distillation.

Label Smoothing (LS) [41]: We follow [21] to train LS with different smoothing intensities and select the model with the highest accuracy. Purchase100 is trained with a smoothing intensity of 0.03, Texas with 0.09 and CIFAR100 with 0.01.

DP-SGD[2]: We use PyTorch Opacus [1] to train the DP-SGD model. We set microbatch size to be 1. Purchase100 is trained with a noise_multiplier of 1.7, a norm clipping bound of 1.0 and with 200 epochs. Texas100 is trained with a noise_multiplier of 1.44, a norm clipping bound of 1.0 and with 200 epochs. Location30 is trained with a noise_multiplier of 2.91, a norm clipping bound of 3.0 and with 50 epochs.



(a) Purchase100 (b) Texas100 (c) CIFAR100 (d) CIFAR10 (e) Location30

(f) Purchase100 (g) Texas100 (h) CIFAR100 (i) CIFAR10 (j) Location30

Fig. 7: Comparison of our defense (HAMP) with early stopping. First row compares the attack TPR @0.1% FPR and second row the attack TNR @0.1% FNR at different epochs. Dashed lines indicate the results by HAMP, while solid lines are those for early stopping (HAMP is trained till it converges, while early stopping is trained with different epoch sizes).

B. *Measuring Prediction Entropy by HAMP*

Please refer to the details in Appendix B in the extended version of this paper in [8], available at this [link].

C. *Label Smoothing with Different Smoothing Intensities*

In Section IV-G, we compare HAMP with LS using the smoothing intensity that achieves the highest accuracy, and we found that HAMP achieves significantly lower MIA risk than LS. In this section, we evaluate LS with other intensities that achieve similar accuracy improvement. On Purchase100, we select a smoothing intensity of 0.03, which yields the highest accuracy improvement of 4.75%, and we consider all seven other intensities that achieve comparable accuracy improvement (3.8%∼4.4%). Fig. 8 presents the results, which show that LS trained with different intensities still exhibit very high MIA risk. For example, the attack TPR @ 0.1% FPR by LS are 13.7x∼15.5x higher than that of HAMP, and the attack TNR are 8.2x∼12.4x higher than that of HAMP.

D. *Comparison with Early Stopping*

Early stopping produces models trained with fewer epochs to prevent overfitting. In our evaluation, we benchmark the classification accuracy and attack TPR/TNR of the models trained with different epochs before convergence, and compare them with HAMP. The results are shown in Fig. 7.

When the model is trained with a few epochs in early stopping, the model is able to achieve comparable privacy protection as HAMP, but with a large accuracy drop. For example, on Purchase100, the model trained with 15 epochs yields an attack TPR of 0.67% and attack TNR of 1.01%, which are slightly higher than the 0.4% and 0.44% by HAMP. However, its prediction accuracy is only 68.6%, which is much lower than the 81.15% achieved by HAMP.

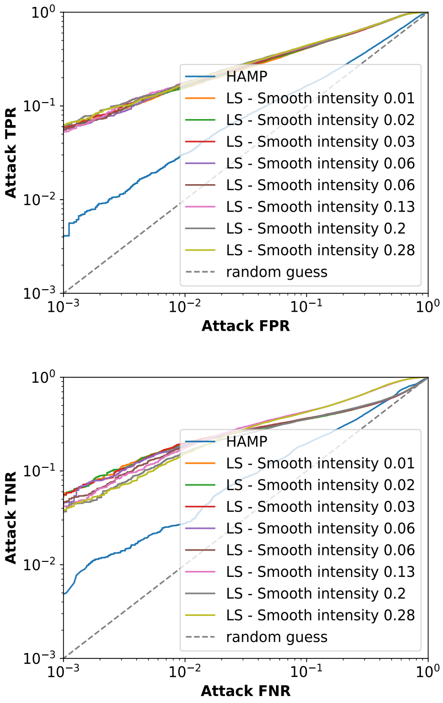


Fig. 8: Comparing HAMP with Label Smoothing under different smoothing intensities.

The model's accuracy improves with more training epochs, but so does the attack TPR and TNR. When the models derived by early stopping converge, there is a substantial gap between the attack TPR and TNR of HAMP and early stopping (black dashed line vs. red solid line in Fig. 7).

Overall, under a similar attack TPR, HAMP achieves an average 12.5% higher accuracy than early stopping; and 28.6%

TABLE IV: Performance of HAMP under different parameters.

	Parameter	Training Acy	Testing Acy	Attack TPR @0.1% FPR	Attack TNR @0.1% FNR
γ (Entropy threshold)	0.9	73.79	66.7	0.38	0.26
	0.8	91.12	81.15	0.4	0.44
	0.7	94.56	82.35	0.53	0.68
	0.6	96.73	83.4	0.87	1.15
	0.5	97.93	83.55	0.98	1.41
	0.4	98.49	83.5	0.9	1.97
	0.3	98.67	83.9	1.23	1.47
	0.2	98.85	84.55	1.17	2.09
	0.1	99.06	84.45	2.02	1.91
α (Regularization strength)	0.5	31.81	29.1	0.31	0.19
	0.1	34.27	33.25	0.15	0.2
	0.05	74.98	68.05	0.22	0.36
	0.01	91.12	81.15	0.4	0.44
	0.005	92.53	81.45	0.44	0.46
	0.001	93.8	82.1	0.56	0.53
	0.0005	94.22	81.85	0.7	0.78
	0.0001	94.69	82.6	0.81	0.91

higher than early stopping when under similar attack TNR.

E. Varying the Parameters of HAMP

This section evaluates the performance of HAMP under different parameters, $\gamma \in (0.1, 0.9), \alpha \in (0.0001, 0.5)$. We use Purchase100 and present the results in Table IV.

Entropy threshold. A higher entropy threshold assigns lower probability to the ground-truth class in the labels and enforces the model to become less confident in predicting training samples. For instance, for the entropy threshold of 0.9, the probability of the ground-truth class is only 20%, while with a threshold of 0.1, the probability is 94%. Table IV shows that a higher entropy threshold leads to a model with lower classification accuracy and also lower MIA risk (on both attack TPR and attack TNR). The highest entropy threshold, 0.9, produces a model with the lowest test accuracy of 66.7% and the lowest attack TPR of 0.38% and attack TNR of 0.26%

Strength of regularization. Stronger entropy-based regularization forces the model to produce outputs with higher uncertainty (uncertainty is measured by the prediction entropy), and is useful in preventing the model's overconfidence in predicting training samples. The model exhibits strong resistance against MIAs when α is large (e.g., 0.05)

On the other hand, strong regularization results in a model with low classification accuracy. This is because, when α is large, the overall loss term in objective (7) is dominated by the second regularization term, while the first loss term for improving classification accuracy is not optimized sufficiently.

F. Overhead Evaluation

Please refer to the details in Appendix F in the extended version of this paper in [8], available at this [link].

G. Understanding the High Attack Performance by the NN-based Attack [31]

Please refer to the details in Appendix G in the extended version of this paper in [8], available at this [link].

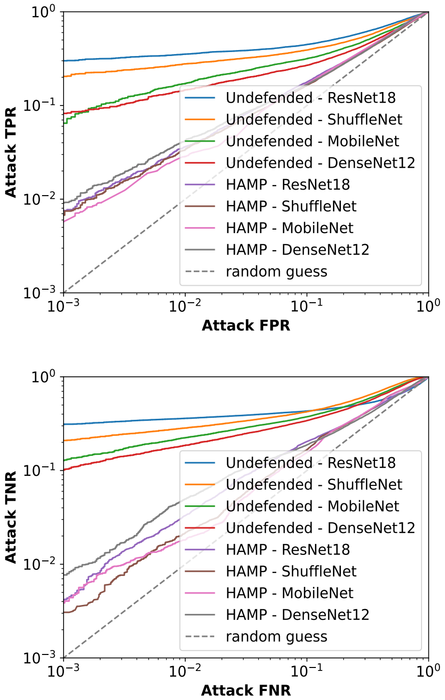


Fig. 9: Evaluation under different network architectures. While models trained with different architectures exhibit varied degree of MIA risk, HAMP *consistently* contributes to low MIA risk despite the specific architecture.

H. Evaluation on Different Network Architectures

This section reports additional evaluation on models trained with different network architectures (using CIFAR10), including DenseNet-12 [17], ResNet-18 [15], MobileNet [16], ShuffleNet [51]. The results are shown in Fig. 9.

Overall, models trained with different architectures exhibit disparate degrees of MIA risk, with the attack TPR @0.1% FPR being 6.47%∼30%, and the attack TNR @0.1% FNR 10.15%∼ 31.12%. This gives an average attack TPR of 16.29% and attack TNR of 18.75%. HAMP is able to consistently reduces the MIA risk, with the attack TPR on HAMP being 0.52%∼0.92% and the attack TNR 0.31%∼0.77%. On average, HAMP reduces the attack TPR by 95.6% (from 16.29% to 0.72%) and the attack TNR by 97.5% (from 18.75% to 0.47). Further, HAMP achieves such strong privacy protection with only a minor accuracy drop of 0.59% (at most 1.28%).

I. Detailed Attack AUC comparison

Please refer to the details in Appendix I in the extended version of this paper in [8], available at this [link].

J. Full ROC Curves

The ROC curves corresponding to Fig. 3 are in Fig. 10.

K. Detailed Results for Each Attack

In Section IV, we reported the highest attack results among all evaluated attacks. We provide the detailed results for each attack for completeness - the detailed results can be found in Appendix G in the extended version of this paper in [8] [link].

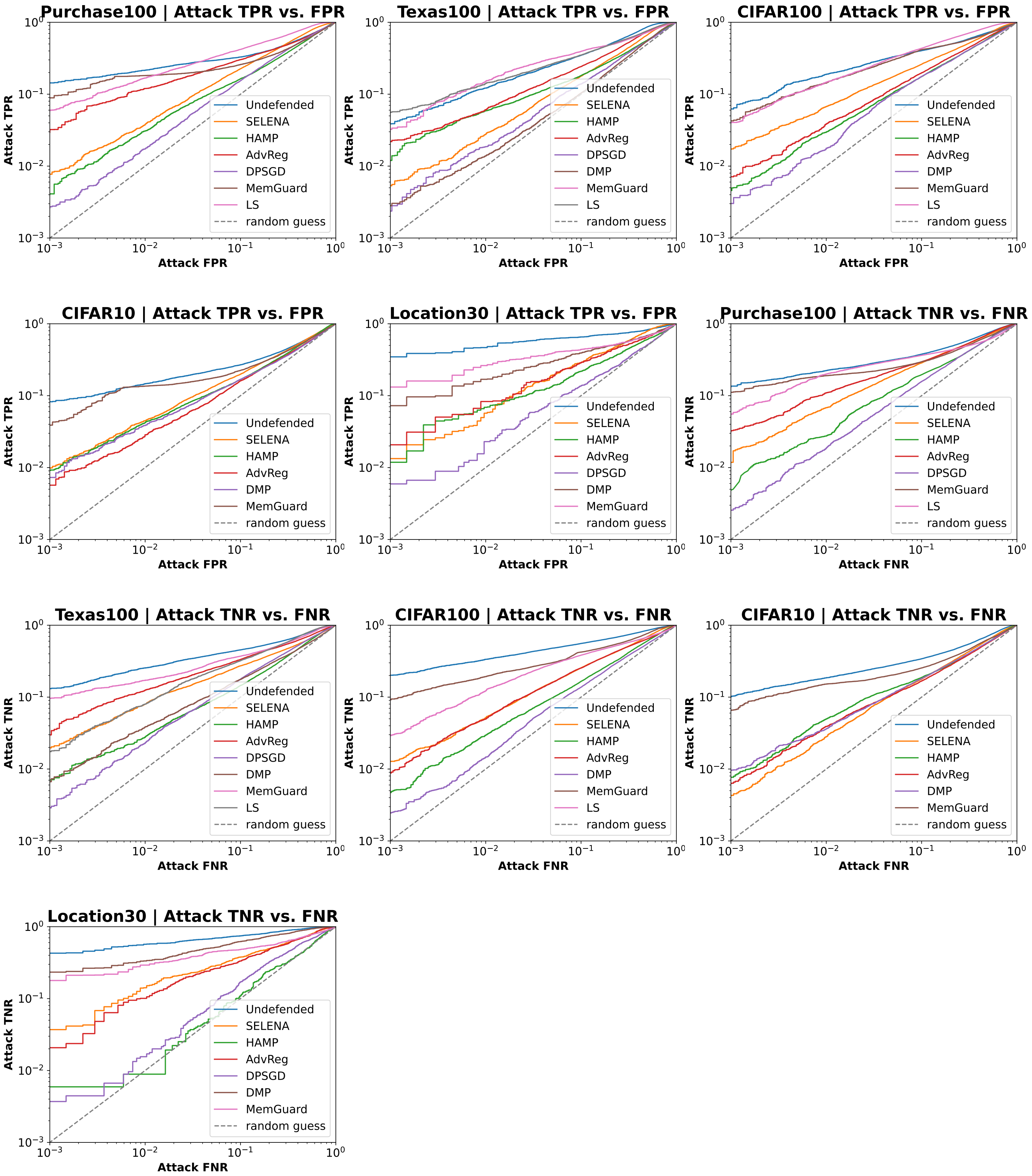


Fig. 10: Full ROC curves, showing attack TPR Vs. FPR, and attack TNR Vs. FNR.

Appendix B
Artifact Appendix

A. Description & Requirements

1) How to access: The artifact is available at https://doi.org/10.5281/zenodo.8271276, and the corresponding GitHub repo can be accessed at https://github.com/DependableSystemsLab/MIA_defense_HAMP.

2) Hardware dependencies: Commodity GPUs (e.g., we used Nvidia RTX 3090, A5000, V100 GPUs in our evaluation).

3) Software dependencies: PyTorch, torchvision, tensorflow, pandas, sklearn, rdt, numpy, scipy, tpdm, numba, matplotlib.

4) Benchmarks: (1) *Datasets*: Purchase100, Texas100, Location30, CIFAR100 and CIFAR10. (2) *Models*: Undefended models, models trained with different privacy defense techniques, including AdvReg, SELENA, DMP, DPSGD, MemGuard, Label Smoothing (LS), and HAMP (our technique).

We provide all datasets, and models used in our evaluation - the links to download these data are available in the artifact's README file.

B. Artifact Installation & Configuration

Please go to the README file in the artifact to:

(1) Install the software dependencies listed in A-3 above.

(2) Download the dataset and models listed in A-4 above.

C. Experiment Workflow

Please see the Evaluation section.

D. Major Claims

- (C1): HAMP achieves strong membership privacy protection and high model accuracy, without requiring additional data, which outperforms existing defenses by achieving a superior privacy-utility trade off. This is proven by the experiment (E1) whose results are reported in Table-I and Fig. 3.

E. Evaluation

Reference to the key results (at the end of Section I): HAMP is able to provide strong membership privacy for both members and non-members, and preserve model accuracy. HAMP reduces the attack TPR @0.1% FPR by 94% and the attack TNR @0.1% FNR by 97% respectively, with only 0.46% accuracy loss on average.

NOTE. Given the time limit of the artifact evaluation, we recommend evaluating the undefended model and the HAMP model, while omitting the evaluation on other defense models.

Such an evaluation can still be used to conclusively demonstrate the models trained by our technique (HAMP) can achieve strong privacy protection and high model accuracy, compared with the undefended models.

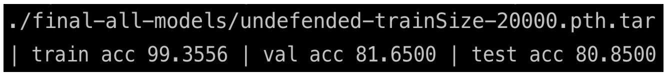
Finally, we have also provided the code and pre-trained models for the other defense models, for anyone interested in performing the full-scale evaluation.

1) Experiment (E1): We consolidate all experimental steps within a single script and you can run it to reproduce the key results on all datasets.

Execution. `bash ./run-all.sh` . This script involves three different evaluation parts (per dataset) as follows, and we use results on the Purchase100 dataset as an example to explain, i.e., `cd ./purchase`

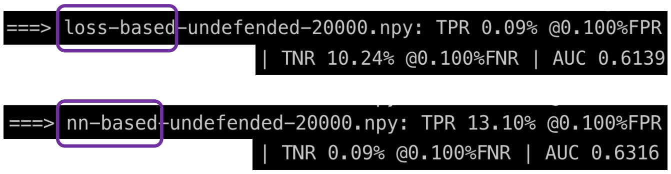
Step 1: Evaluate score-based attacks (except LiRA), and compute model accuracy.

Results. Read the output file `R-atk` , where you can find the results on model accuracy (e.g., Fig. 11), and privacy risk (e.g., Fig. 12). The privacy risk is evaluated by using multiple attacks (e.g., loss-based, NN-based attack).



```
./final-all-models/undefended-trainSize-20000.pth.tar
| train acc 99.3556 | val acc 81.6500 | test acc 80.8500
```

Fig. 11: Example output showing **model accuracy** on the undefended model.



```
===> loss-based-undefended-20000.npy: TPR 0.09% @0.100%FPR
                    | TNR 10.24% @0.100%FNR | AUC 0.6139

===> nn-based-undefended-20000.npy: TPR 13.10% @0.100%FPR
                    | TNR 0.09% @0.100%FNR | AUC 0.6316
```

Fig. 12: Example output showing the **privacy risk** on the Undefended model using the loss-based attack (first row) and NN-based attack (second row).

Step 2: Evaluate Likelihood-ratio attack (LiRA).

NOTE. The LiRA evaluation requires training multiple shadow models (128 in our experiments) to compute the logit-scaled scores for performing the hypothesis test, and shadow model training is a very time-consuming process.

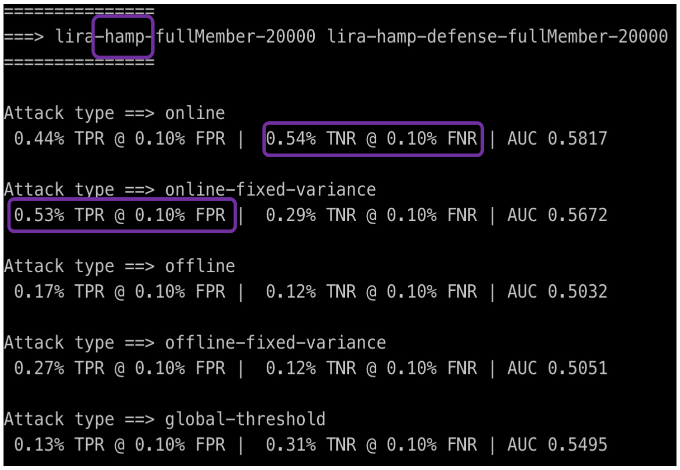
To facilitate the evaluation within the limited time, we provide the full logit-scaled scores computed from the shadow *undefended* models and shadow *HAMP* models on every dataset, which saves the high cost of shadow model training. We also provide the instruction on how to run the full-scale evaluation by training the shadow models from scratch - please refer to the README file in the artifact.

Results. Read the output file `R-undefended-lira` or `R-hamp-lira` . An output snapshot is in Fig. 13.

Step 3: Re-train the models from scratch.

Results. Read the output file `R-train` , where you can find the accuracy of the undefended model and HAMP model trained from scratch.

2) Interpreting the results.: The model accuracy reported from step 1 (for using the pre-trained models we provide) or step 3 (for using the newly trained models), can be used to compute the accuracy loss incurred by HAMP. These results can be compared with those in Table I, which reports an



```
===============
===> lira-hamp-fullMember-20000 lira-hamp-defense-fullMember-20000
===============

Attack type ==> online
 0.44% TPR @ 0.10% FPR |  0.54% TNR @ 0.10% FNR | AUC 0.5817

Attack type ==> online-fixed-variance
 0.53% TPR @ 0.10% FPR |  0.29% TNR @ 0.10% FNR | AUC 0.5672

Attack type ==> offline
 0.17% TPR @ 0.10% FPR |  0.12% TNR @ 0.10% FNR | AUC 0.5032

Attack type ==> offline-fixed-variance
 0.27% TPR @ 0.10% FPR |  0.12% TNR @ 0.10% FNR | AUC 0.5051

Attack type ==> global-threshold
 0.13% TPR @ 0.10% FPR |  0.31% TNR @ 0.10% FNR | AUC 0.5495
```

Fig. 13: Example output showing the **privacy risk** on the HAMP defense model using LiRA with 128 shadow models.

average accuracy loss of 0.46% by HAMP[2]. Note that model training is a statistical learning process, and thus the re-trained models may have different accuracy compared with the ones reported in the paper (this applies to both the undefended and HAMP models). If you find the difference to be major[3], please re-do the training again.

For the privacy risk evaluation, the highest attack TPR@0.1% FPR and highest attack TNR@0.1% FNR can be selected from the overall results by step 1 *and* step 2. These results can be compared with those in Fig. 3, which reports HAMP reduces the TPR@0.1% FPR by 94% and the TNR@0.1% FNR by 97%[4].

The overall results can be used to demonstrate the strong privacy protection and high model accuracy offered by HAMP.

[2] We re-executed the experiments and found that HAMP incurred a similar accuracy loss of 0.68%.

[3] For example, we find training HAMP on the Location30 dataset sometimes produced a model with accuracy much lower than the one reported in the paper (51.5% vs. 56.3%). By doing the training again, we were able to produce a model with similar accuracy.

[4] We re-executed the experiments and found that HAMP achieved similarly strong results - it reduced the attack TPR by 92% and the attack TNR by 98%.