NeuGuard: Lightweight Neuron-Guided Defense against Membership Inference Attacks

Anonymous Author(s)

ABSTRACT

Membership inference attacks (MIAs) against machine learning models lead to serious privacy risks for the training dataset used in the model training. The state-of-the-art defenses against MIAs often suffer from poor privacy-utility balance and defense generality, as well as high training or inference overhead. To overcome these limitations, in this paper, we propose a novel, lightweight and effective Neuron-Guided Defense method named *NeuGuard* against MIAs. Unlike existing solutions which either regularize all model parameters in training or noise model output per input in real-time inference, *NeuGuard* aims to wisely guide the model output of training set and testing set to have close distributions through a fine-grained neuron regularization. That is, restricting the activation of output neurons and inner neurons in each layer simultaneously by using our developed class-wise variance minimization and layer-wise balanced output control. We evaluate *NeuGuard* and compare it with state-of-the-art defenses against two neural network based MIAs, five strongest metric based MIAs including the newly proposed label-only MIA on three benchmark datasets. Extensive experimental results show that *NeuGuard* outperforms the state-of-the-art defenses by offering much improved utility-privacy trade-off, generality, and overhead.

1 INTRODUCTION

Machine learning (ML) has achieved extraordinary success in many fields, spanning daily image classification, object detection, and privacy- and security- sensitive medical diagnosis [33], biometric authentication [27], and autonomous vehicles [52]. Such success mainly depends on training ML models with the large-scale domain-specific datasets that may contain crucial confidential and private information, e.g. personal medical records, human faces. Unfortunately, many recent studies have revealed that such data information can be retrieved from the trained ML models by various attacks, such as attribute inference attack, model inversion attack, and membership inference attack [5, 8, 9, 17, 24, 28, 40, 49–51, 59].

Among these attacks, membership inference attack (MIA) [40] has been attracting ever increasing attention, with the goal of identifying whether a given data sample is used for training the target ML model or not. For instance, if a model is trained using a specialized medical database, e.g. data pertaining to individuals' disease, a successful MIA could reveal the identity of a person having the disease. In essence, MIAs are built upon the fact that a model's response to the member (training set) and non-member (testing set) can be different by nature. This difference allows the attacker to either train neural network (NN) based binary classifiers [31, 40] or use non-NN metric based approaches [5, 38, 41] for accurate member inferring. Given the difficulty of fundamentally eliminating such a difference, defending against MIAs can be challenging.

There exist many studies to address MIAs, including provable defense like differential privacy (DP) [1, 36], and non-provable defense like training time defense with dropout [38], L_2 norm [40], MIA-dedicated min-max adversary regularization [31], distillation based defense [39] and inference time defense via output perturbation [16]. Compared with provable defense like DP that is known for provable privacy but extremely low model utility [1, 15, 39] (see Section 8), the non-provable defenses offer privacy empirically but much better utility. However, as we shall show in Section 3.1, these solutions are still far from satisfactory in terms of model utility-defense effectiveness balance, defense generality against a wide range of MIAs, and training and inference overhead. *The underlying reasons are that existing solutions either regularize and control all model parameters in a coarse-grained manner with no guarantee of minimizing the model output distribution difference between training set and testing set, or directly manipulate output scores for achieving similar distributions by expensively searching and adding sample-specific perturbation noise during the real-time inference.* To this end, there has been lack of systematic studies on an important facet, that is, effectively guiding the model to produce output that would ensure that the output distribution for training set and test set are very close, with much better defense effectiveness and generality at marginal utility loss and overhead.

To achieve this goal, in this paper, we propose to create a new paradigm of safeguarding MIAs from a radically different perspective. Our basic idea is to **develop lightweight and fine-grained neuron-level regularization to simultaneously guide and orchestrate the final output neurons and hidden neurons (or intermediate features) for producing an output confidence score distribution indistinguishable between the training set and testing set.** *The key rationale is that*: if we can deliberately and largely reduce the space of distributions obtained over confidence score for training data by developing privacy-dedicated neuron-level training regularization, then such trained model will naturally confine output score distribution to a very limited space regardless of input (either training or testing data). In this way, the output score distribution of testing data produced by this model is also restricted to the similar space with a much smaller variance (as confirmed in Table 5/ 7), and thus is close to that of training data. While there still exist some score distribution differences between the training and testing data, the gap between them can be much smaller compared with existing solutions, thereby offering much better defense effectiveness against a variety of MIAs. However, the challenges lie in designing a learnable target output distribution with much reduced space and neuron regularizations workable for output and inner neurons with minimized utility loss and overhead.

To overcome these challenges, we design a class-wise variance minimization regularization that directly acts on the final output neurons to minimize the variance of output score in training, along with layer-wise balanced output control regularization for inner neurons to further guide the learning of intermediate features. As a result, the output confidence score of any input can be more evenly distributed with all label's confidence values close to the mean value

while still maintaining the prediction correctness (see Fig. 2(a)). In this way, we successfully restrict the value range and distribution of the output score, and hence reduce the model response difference between the training set and testing set. We name such a Neuron-Guided Defense against MIAs as ***NeuGuard*** in this work.

We evaluate our proposed *NeuGuard* against both NN based attacks and the latest metric based attacks (including the label-only attack), and compare it with state-of-the-art defenses [16, 31, 41] on three datasets. Experimental results show *NeuGuard* delivers by far the best utility-defense trade-off among the known defense mechanisms. Our *NeuGuard* achieves lower attack accuracy against all attacks at marginal utility drop and much lower training and inference overhead. Moreover, *NeuGuard* also provides a much better generality comparing with prior solutions, that is, models protected by *NeuGuard* are resilient to all kinds of NN and Non-NN metric based MIAs. Our contributions are summarized as follows:

- We, for the first time, investigate the difference between neural network based membership inference attacks that use sorted output confidence scores or unsorted scores with label information as two different attack inputs, and find that existing defenses workable for one attack often do not work for the other.
- We develop a novel, simple and effective neuron-guided defense–, to explicitly reduce the model output distribution difference between training set and testing set. It promises the best privacy and utility trade-off.
- We extensively evaluate our defense against NN based membership inference attacks and metric based attacks, outperforming existing defenses on three real-world datasets.
- We explore why defenses cannot reduce the metric-based attacks to random guessing, and perform analytical and experimental studies on the upper bound of defense effectiveness.

2 BACKGROUND

2.1 Membership inference attacks

The MIA attempts to infer whether a given data is from the training set or not that is used to train a target model [5, 38, 40–42, 57]. There exist many MIAs with different evaluation measurements and attack capabilities. In this work, we discuss two major classes of MIAs: *neural network based attacks* and *metric based attacks*.

2.1.1 Neural network based attacks. They aim to identify the given data's membership using neural networks and are performed in the white-box or black-box fashion. White-box MIAs assume the attacker has full access to the target model [22, 32], e.g. model architecture and parameters. Black-box MIAs assume the attacker can only observe the output confidence scores of the target model, which are more realistic and the focus of this work.

Given a data sample and its confidence scores outputted by a target model, neural network based attack trains a binary classifier to determine whether the data belongs to the training set or not. In particular, there are two kinds of black-box attacks, and they major differ in whether the output confidence scores are sorted before the attack or not. *We would like to emphasize that the existing research does not explicitly differentiate these two kinds of attacks, and we notice that a defense being effective to one attack is often not effective to the other type (Please see Section 6).* We, for the first time, notice the difference of these two attacks, and our proposed defense *NeuGuard* shows better defense generality against both the attacks.

NN based attack with sorted input: The first NN based attack takes the sorted output confidence scores as the attack input. We name this attack *sorted NN attack* in short. The binary attack classifier focuses on learning the difference between training set and testing set while ignoring the correctness of the class prediction. Training sorted NN attack can be realized by different approaches: 1) using a shadow model to represent the target model and training it by performing MIA with multiple class-wise attack classifiers[40]; 2) using a general attack classifier for all classes and achieve similar attack accuracy [38].

NN based attack with unsorted input: The second NN based attack uses both the unsorted output confidence scores and label information as attack inputs [31]. We call it *unsorted NSH attack* for short in this paper. This attack uses three neural network models to construct the binary classifier. One model is used to receive the unsorted confidence scores, and one takes the one-hot encoded class label as input. The last one concatenates the outputs from the other two models and generates a single probability, to determine whether the given data is a member of the training set or not.

2.1.2 Metric based attacks. Unlike neural network based attacks, metric-based attacks directly compute customized metrics using prediction confidence scores or only the label information to infer membership or non-membership. We introduce the four state-of-the-art (strongest) metric based attacks and the newly proposed label only attack for our later evaluation [5, 41]. Let (x, y) be a data sample with features x and label y and F be a NN model.

Prediction correctness based MIA. This attack infers the membership based on whether the given input data is correctly classified by the target model or not [57]. The associated metric $\mathcal{M}_{\text{corr}}$ is defined as:

$$\mathcal{M}_{\text{corr}}(F; x) = I(\text{argmax}\, F(x) = y) \tag{1}$$

where $I(\cdot)$ is an indicator function.

Prediction confidence based MIA. This attack determines whether an input data is from the training set or not by comparing the most significant confidence score with a preset threshold. The attack is first designed by [38] using a single threshold for all classes and [41] improves it by applying class-wise thresholds to reduce the impact of confidence differences among classes. The associated metric $\mathcal{M}_{\text{conf}}$ is defined as:

$$\mathcal{M}_{\text{conf}}(F; x) = I(F(x)_y \geq \tau_y), \tag{2}$$

where τ_y represents the threshold for the class y.

Prediction entropy based MIA. The entropy based MIA attack [41] is based on the fact that the testing set prediction entropy should be much larger than the training set. It identifies the input data x as a member if the prediction entropy is smaller than a preset threshold (e.g., $\hat{\tau}_y$). The associated metric $\mathcal{M}_{\text{entr}}(F; x)$ is defined as:

$$\mathcal{M}_{\text{entr}}(F; x) = I(-\textstyle\sum_{i=0}^{k} F(x)_i \log\left(F(x)_i\right) \leq \hat{\tau}_y). \tag{3}$$

Modified prediction entropy based MIA. Prediction entropy attack has a major limitation in that it doesn't contain any label information [41]. As a result, both correct and incorrect predication with high score can lead to zero entropy values. Modified prediction entropy [41] fixes this issue that only high probability correct prediction can lead modified entropy to 0. Then such modified entropy $\text{ME}(F(x), y)$ is presented as: $\text{ME}(F(x), y) =$

$-\left(1-F(x)_y\right)\log\left(F(x)_y\right)-\sum_{i\neq y}F(x)_i\log\left(1-F(x)_i\right)$. The adversary determines an input data as a member if the above modified value is smaller than the preset class-related threshold–$\check{\tau}_y$ for class y. The associated metric $\mathcal{M}_{\text{Mentr}}(F;x)$ is defined as:

$$\mathcal{M}_{\text{Mentr}}\ (F;x) = I(\text{ME}(F(x),y) \leq \check{\tau}_y) \tag{4}$$

Label-only MIA [5]. It is a newly proposed MIA that tries to infer membership or non-membership with only the label information. Specifically, for a data example (x,y), it first generates an adversarial example x' (an example close to x but has a wrong predicted label) and calculates the l_2-distance to a model's decision boundary as $dist_F(x',y)$. Then x is predicted as a member if $dist_F(x',y) > \tau$ (τ is a threshold) since a training data should exhibit higher robustness than testing data. In our evaluation, we adopt the strongest C&W [3] based label-only MIA to generate adversarial examples. This attack serves as the upper bound of the label-only MIAs, as it uses the actual model's information (e.g. gradient) to generate ideal adversarial examples for MIAs (white-box setting) [5].

2.2 The state-of-the-art defense methods

We mainly introduce two kinds of most representative and powerful defenses for detailed experimental comparisons in Section 6 and 7: training regularization and inference output perturbation.

2.2.1 Training regularization defense. Many works [4, 22, 38, 40, 57] have pointed out that model overfitting is the main reason that makes MIAs effective. Based on this observation, adding a regularizer during training to reduce overfitting can be an effective way to defend against MIA. The common regularization methods include L2-norm regularization[40], dropout [44], model stacking[38], early stop[41]. However, they mainly focus on model utility, and thus are unable to greatly reduce membership inference attack accuracy.

Besides, adversary regularization[31] (**AdvReg** in short) is specifically designed to defense against MIAs. The method introduces an NN based membership inference classifier during training to achieve the defense. The training process needs to simultaneously minimize the target model's loss and the attack classifier's accuracy over the training set D_{tr}:

$$\underset{\theta}{\text{argmin}}\frac{1}{|D_{\text{tr}}|}\sum_{\mathbf{x}\in D_{\text{tr}}}L(F(x),y)+\lambda\log(I(F(x),y)), \tag{5}$$

where λ is a parameter to balance the privacy risk and original classification task, and $I(\cdot)$ is the attack classifier.

2.2.2 Inference output perturbation defense. Another effective way is to directly modify the output confidence scores such that the information is hidden from the adversary. The basic idea of the best existing defense method **MemGuard** [16] is to add carefully calculated noise into the output confidence scores and turn the output into adversarial examples [2] to mislead the attack classifier for each input. Since the noise is crafted and added at the inference stage, this method does not influence the training process and maintains the target model's accuracy.

3 MOTIVATION AND THREAT MODEL

3.1 Motivation

We aim to satisfy the following requirements to achieve a good defense against MIAs. Existing solutions, however, are incapable of addressing challenges to meet such requirements:

Table 1: Comparing existing MIA defenses.

Method	Defense effectiveness	Defense Generalizability	Model utility	Overhead
Normal training	−−	−−	++	No
Dropout [38]	−	−	++	Low
Early stopping [41]	+	+	−	No
AdvReg [31]	+	+	−	Medium (Training)
MemGuard [16]	++	−	++	High (Inference)

'++' indicates the best , '−−' means the worst.

- **Defense effectiveness:** A good defense shall reduce MIA attack accuracy as close as to a random guess, i.e., 50%.
- **Defense generalizability:** A good defense shall be able to defend against different types of attacks considering the uncertainty of experiencing which MIA in practice.
- **Utility loss:** A good defense shall maintain the target model's accuracy on unseen data as much as possible. A defense causing a large accuracy drop is not desirable.
- **Overhead:** A good defense shall be lightweight and not incur significant overhead in training or inference.

Table 1 briefly evaluates the state-of-the-art defense methods discussed in Section 2.2 using the above four criteria. These results are summarized based on previous works [16, 31, 41] and our experiment results (details in Section 6.1 and 7). According to the table, existing defenses all have limitations in some aspects.

Dropout reduces model overfitting, but exhibits limited MIA defense, e.g. only slightly better than the baseline. **Early stopping** sacrifices model utility to improve defense efficiency, but obtains a sub-optimal utility-defense trade-off.

AdvReg leverages a min-max game theoretic method to train the model against MIAs. However, it cannot provide an effective defense with an acceptable utility in practice[39, 41]. As Table 2 shows, the defense takes effects on the Texas100 when $\lambda \geq 3$, but it leads to very unacceptable model accuracy drop, e.g. $\sim 8\% - 18\%$. Furthermore, [41] evaluates AdvReg in metric based MIAs and shows it is no better than the early stopping model under a similar test accuracy. The reason is that AdvReg directly incorporates the MIA classifier as a regularization term into the training but it may not have an explicit goal to regularize the model towards lowering MIAs. Besides, the difficulty of jointly optimizing the MIA classifier and original task also leads to considerable training overhead comparing to the regular training.

MemGuard performs defense at the inference stage such that the model utility is not affected and there is no training overhead. However, it suffers from the following limitations which hinder it from satisfying the given defense requirements: **First**, MemGuard causes huge inference overhead as it needs to run calculations for many times (e.g. up to 300×) for each input sample to find out the best noise output to defend against the attack. **Second**, its defense performance is highly dependent on the given trained model. It may fail to find the noisy outputs for the target model to effectively defend against the attack, leaving a high attack accuracy. **Third**, MemGuard cannot provide general protection against different attacks. If it uses a sorted NN classifier as the defense classifier, it cannot defend the attack with the unsorted NSH model and vice versa (details in Section 6). [41] further evaluates MemGuard under metric based attacks and shows that the defense works to a limited degree and is not much better than the baseline model. In addition, as demonstrated in [5], post-processing output scores

Table 2: Model accuracy and NN based MI accuracy with different λ values for the adversary regularization on Texas100.

λ		1	2	3	5
Training set accuracy		85.99	86.08	66.67	47.77
Testing set accuracy		58.51	58.17	50.92	40.59
MI accuracy	Sorted NN	68.56	68.31	64.18	55.81
	Unsorted NSH	60.41	60.44	53.48	52.24

like MemGuard fails to defend against label-only attacks as model produced raw scores remain unchanged (i.e., unchanged distance between input to decision boundary to infer members in label-only).

Based on the above limitation analysis of existing defenses, we believe a better way to defend against MIAs is to develop a method that is unlike AdvReg and has fine-grained regularization control guided by more specific objectives with the guarantee of lowered MIAs. The defense method should also maintain the model utility and provide a good utility-defense trade-off, work under different MIAs (e.g., sorted NN, unsorted NSH, metric-based attacks), and has much lower overhead than existing defenses (e.g. MemGuard).

3.2 Threat model

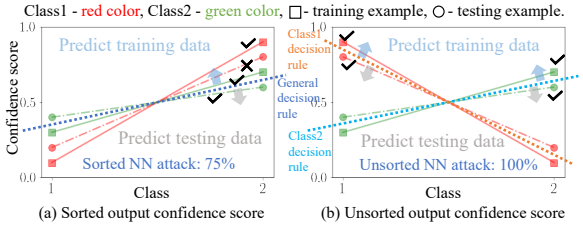
In this work, we adopt a threat model consistent with previous defenses [16, 31]. We assume model providers have a private training dataset (e.g., financial records, healthcare dataset, or location dataset). They train a machine learning model with the private dataset and deploy the trained model as a service that only provides prediction results to other users. We also assume the model providers can apply defense methods in the training and/or inference stages. The providers' goal is to make the model have satisfactory test performance and capable of defending against various membership inference attacks. The attackers aim to infer the membership of the private data in the training set from the deployed model. We assume the attacker could receive the output confidence scores or predicted labels directly generated by the target model from the server, instead of the shadow models estimated by attackers locally, to maximize MIA effectiveness for attackers. However, attackers cannot access the model itself or the model parameters.

Partial knowledge of the training/testing data: To consider a strong attack, we assume the attacker also has access to part of the training/testing data and the attacker has the capability to query sufficient data samples from the target model to perform MIAs including label-only attacks. The attack classifier is trained based on the output of these known data samples from the target model. This assumption allows more reasonable and realistic attacks and aligns with the state-of-the-art defense studies on MIAs [16, 31, 41].

4 DESIGN

4.1 Design overview

To develop a defense that can fulfill all requirements in Sec. 3.1, it is important to understand the differences of MIAs and the impact of these differences on the defense. In particular, we notice that the sorted and unsorted NN attacks learn different information from the different input formats. The former learns **ONLY sorted** output confidence scores with label information eliminated *(unified highest to lowest scores via sorting for any class)*, while the latter learns *original unsorted scores together with label information*. The former relies on a general rule among all classes to infer members, thus attack accuracy is greatly impacted when the score difference between training and testing data *at different classes* is different, as



(a) Sorted output confidence score (b) Unsorted output confidence score

Figure 1: (a) Sorted, (b) Unsorted output confidence distribution for binary classification model. (Class1: D_{train}:[0.9, 0.8], D_{test}: [0.8, 0.2]; Class2: D_{train}:[0.3, 0.7], D_{test}: [0.4, 0.6])

Fig. 1(a) shows. The latter learns a different output distribution **for each class** to attack like Fig. 1(b), however, incorrect prediction results introduce noises into attack training to reduce attack accuracy. Consequently, one attack is not uniformly stronger than the other, and it is necessary for a good defense to work against both.

With the understanding of the difference of MIAs, we now explore the ordinary case where the training accuracy and testing accuracy have a large gap, leading to a huge difference in *distributions* of confidence scores over the training and testing sets. For example, Fig. 2(b) shows the Alexnet model's output distribution on a real-world dataset CIFAR10 with 10 classes, where we can observe a substantial difference between the two output distributions of the training set and testing set. It is indeed the difference between these distributions that is exploited by various attacks in different ways to infer membership in the training set. Existing defenses either propose to add an adversarial regularization term during the training or to manipulate the model's outputs during the inference stage to reduce the distribution gap. However, they either cannot provide enough distortions in training to suppress the difference (see Fig. 4(b)) or can only defend against one type of attacks (see Fig. 4(a), where the perturbed output cannot defend against unsorted NN attack, because the false predicted testing set can be easily distinguished by the attack classifier).

In summary, we can learn that **for models with overfitting and a large accuracy gap between training and testing data, only a uniform distribution with marginal difference between classes can deceive both sorted and unsorted NN attacks and serve as an effective distribution against MIAs.** One possible path to this end is to restrict all confidence scores to be evenly distributed and thus reduce the output distribution gap between the training set and testing set. As the example in Fig. 2(a) shows, in a 10 class output, when all of the confidence scores of each class are close, the value of each confidence score should be close to the mean value (e.g. 0.1), where the score of correct label is slightly higher than the mean value and that of others are lower than it. Note that while it is not possible to "know" the distribution of scores for the testing set, limiting the distribution space of the model output is an indirect way to ensure closeness of the distributions arising from training and testing sets.

To achieve this goal, we propose a novel framework to control the output confidence score through dedicated output and inner (hidden) neuron control. As Fig. 2 shows, we seamlessly integrate our class-wise variance minimization (for output neurons) into the customized layer-wise balanced output control (for hidden neurons) intermediate results to construct the privacy preserving neural network model. In the following, we will first detail the class-wise

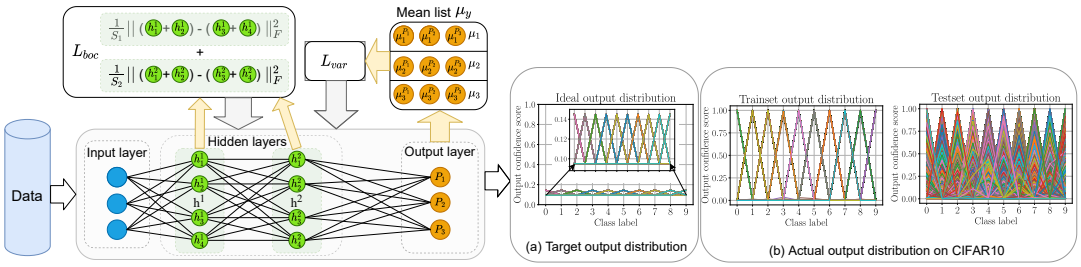


Figure 2: An overview of our proposed neuron guided defense method with *class-wise variance minimization* L_{var} and *layer-wise balanced output control* L_{boc}. (a) is the ideal output distribution that we aim to achieve by our proposed method (value space limited). (b) shows the actual output score distributions of training and testing set on the CIFAR10 by normal training.

variance minimization regularization, then the layer-wise balanced output control regularization, and finally present the combined training process to generate defense-efficient outputs.

4.2 Class-wise variance minimization

We propose to enforce the output confidence scores of all classes close to their mean distribution during the training. In this way, we expect that the output distribution can be in a similar range in both the training and testing set. In particular, we choose to add the class-wise variance as a regularization term L_{var} into the loss function. Since variance directly measures the spread around the mean value, minimizing the variance allows us to control the space of output distributions such that the training and test output distributions are close to an "ideal" distribution.

$$L_{var} = \frac{1}{N} \sum_{i=0}^{N} (F(x) - \mu_y)^2 \tag{6}$$

where $F(x)$ is the target model output confidence score for the training data (x, y), and N is the batch size. μ_y represents the mean list of corresponding class y that is used to calculate the expectation of the squared deviation of the output. For example, as Fig. 2 shows, the mean list μ_1 is a three dimensional vector–$(\mu_1^{P_1}, \mu_1^{P_2}, \mu_1^{P_3})$ corresponding to three output neurons, which is an average output confidence score vector for data samples with the class label 1.

There are different ways to calculate the mean vector μ for the variance minimization to train neural networks. In the following we discuss three of them and provide a reasoning for choosing the class-wise approach. We briefly show the model accuracy and the neural network based MIA of Cifar10 classification task with three different variance calculation methods on Table 3.

Batch-wise: The simplest way is to perform the batch-wise variance calculation that minimizes the variance of each batch's output. The mean list calculated in each batch can have different results and the variance minimization may not have a consistent goal. Thus, this method can maintain the model accuracy at the standard level, but it does not provide the necessary defense.

Single-sort: Another way is to calculate variance with a single sorted mean list, which means to sort all outputs and maintain a sorted mean list for all data to calculate the variance of each output. This approach provides a uniform goal unlike the batch method, however the distribution and accuracy for each class is highly variable. As a result, training with this regularization term cannot converge and leads to a huge model accuracy loss.

Class-wise: In order to set a consistent goal for the variance minimization as well as consider the effects of each class, we proposed a fine-tuned approach to calculate the output variance across multiple predictions from the same class via updating the class-wise mean list. Since we calculate the empirical mean of the output confidence scores for each class, we don't need to sort the output confidence scores and can learn more information from the relationship of each label in the output distribution. Table 3 shows that the class-wise variance minimization can provide best utility-defense trade-off.

Table 3: Testing/MI accuracy on unsorted NSH and sorted NN model with variance minimization training on CIFAR10.

β = 300		Batch-wise	Single-sort	Class-wise
Training set accuracy		74.09	51.77	71.29
MI accuracy	Sorted NN	71.41	60.18	50.4
	Unsorted NSH	62.39	55.74	57.41

While class-wise variance addresses the output distribution difference between training and testing set to an extent, weakness of existing defenses against a sorted NN attack suggests that a finer control of model training is required for which we propose the layer-wise output control, described in the subsequent section.

4.3 Neu. net. layer-wise balanced output control

In this section, we propose to perform a neuron regularization through layer-wise balanced output control in the training process to further constrain the output distribution. The neuron regularization is first introduced to perform training while formulating the value of the weights that are in favor of pruning goals [54, 60]. Unlike their design to manipulate the parameters to 0 or some specific values, our design concentrates on balancing the parameters [34] in each layer so that there will be no particular parameters to dominate the prediction results. To perform the layer-wise balanced output control, we separate the output of each layer into two groups, calculate the summation within each group and finally add the mean square error for the group output difference at each layer. The final summation L_{boc} is used as the regularization term to add into the loss function for the training.

$$L_{boc} = \sum_{l=1}^{M} \frac{1}{S_l} \left\| \sum_{i=1}^{\lfloor S_l/2 \rfloor} h_i^l - \sum_{i=\lfloor S_l/2 \rfloor+1}^{S_l} h_i^l \right\|_F^2 \tag{7}$$

In the equation, M is the number of layers, and S_l is the number of layer l's outputs. h_i^l donates the i's output on layer l. The balanced output control on each layer can help to regularize the effects of individual outputs in a general sense as we try to minimize the difference of the output groups. Thus there will be no extremely large values and the intermediate results will be more balanced. Combining this layer-wise balanced output control with variance minimization method for the model training, we can train the target model with high utility and defense effects to the best trade-off.

4.4 Neuron regularization-based training flow

With two regularization terms used for the training, our overall loss function is as follows:

$$Loss = L(F) + \alpha \cdot L_{boc} + \beta \cdot L_{var}, \tag{8}$$

where we use parameters α and β to control the balance between the optimizing classification task and the effects to constrain the output distribution to perform the defense. Here we present the loss calculation algorithm that contains both class-wise variance minimization and layer-wise balanced output control. During each batch of training, the training algorithm will first calculate the difference of two evenly split groups in each layer l using the intermediate results h^l to compute L_{boc}. Then it will update the class-wise mean list μ_y with all data records. With the updated μ_y, it can calculate the variance L_{var} for the data. Finally, the total loss can be added and used for the weight update. The detailed pseudo code is shown in Algorithm 1 in Appendix.

Using our proposed regularization method, we can further improve the performance by amplifying the layer-wise intermediate features. In some cases such as convolutional layers, given the complex feature extraction from coarse to fine through a layer-wise manner, the original feature maps may not be salient enough for our proposed method to learn during the training. Thus in practice, we can perform a layer-wise amplification to intentionally enlarge the top η% of the feature maps' values in the training process, so that the proposed solution in Eqn. 8 can better learn and regularize the most important features, making the model output scores converge to desired distributions easily. Furthermore, we could also leverage such layer-wise amplification during the inference to alleviate the effect of amplified features by training for better maintaining the model utility. We demonstrate its effectiveness in Appendix A.4.

5 EXPERIMENTAL SETUP

Datasets and parameter setting. We use three benchmark datasets Texas100 [40], CIFAR10 and CIFAR100 [20] to demonstrate the effectiveness of our *NeuGuard* defense against membership inference attack for different application scenarios. We follow the data splitting strategy in [16, 31] (See Table 9) and have the detailed description in the Appendix A.1.

We compared our method with three state-of-the-art MI defense methods, i.e., training time based adversarial regularization (AdvReg) [31], Early stopping [41], and inference time based MemGuard [16]. We also select the regular model without defense for comparison. We use the publicly available source code of the three methods for evaluation. All methods are implemented in Pytorch [35]. All experiments are run on a linux PC with AMD Ryzen Threadripper 2990WX 32-Core Processor, 128 GB memory and NVIDIA GeForce RTX 2080 Ti GPU with 11 GB graphic memory.

We follow the training method and hyperparameter setup on each defense methods proposed by the authors and use the published code to evaluate the corresponding defense mechanism. We choose the best results to compare in the evaluation. The detailed setting can be found in Appendix A.2.

As for our proposed method, we apply the *NeuGuard* methods with different settings. For the fully connected model on the Texas100 dataset, we apply the variance minimization method with $\beta = 3000$ and layer-wise balanced output control with $\alpha = 100$. For CIFAR100 task, we set the $\beta = 1000$ and $\alpha = 5$. For the proposed methods to better learn and control the output, we amplify the top 10% of the convolution layer's feature map 2 times during the training, and amplify 2 times of the top 25% feature map in the inference stage. For CIFAR10 task, we set the $\beta = 200$, $\alpha = 30$, and amplify 1.5 times the top 10% feature map in training. For the inference stage, we amplify 1.5 times of the top 35% feature map to obtain the best utility-privacy trade-off.

Table 4: Results of compared defenses against NN based MI attacks. AdvReg and MemGuard are originally designed to defend against sorted NN and unsorted NSH attacks, respectively. Baseline is the normal training without defense.

Texas100		Baseline	Early stopping	AdvReg	MemGuard	***NeuGuard***
Testing accuracy		58.5	50.9	51.2	58.5	55.8
MI accuracy	Unsorted NSH	65.75	57.42	64.18	50.83	50.58
	Sorted NN	60.98	53.32	53.48	60.52	54.54
Training time(s)		0.006	0.006	0.328	0.006	0.045
Training overhead		1×	1×	54.7×	1×	7.5×
Inference time(s)		0.002	0.002	0.002	1.8	0.002
Inference overhead		1×	1×	1×	900×	1×

CIFAR100		Baseline	Early stopping	AdvReg	MemGuard	***NeuGuard***
Testing accuracy		43.8	41.0	39.6	42.9	43.0
MI accuracy	Unsorted NSH	80.95	60.70	62.67	50.41	51.42
	Sorted NN	81.42	59.62	58.64	59.63	57.82
Training time(s)		0.017	0.017	0.050	0.017	0.045
Training overhead		1×	1×	2.96×	1×	2.62×
Inference time(s)		0.017	0.017	0.017	1.7	0.025
Inference overhead		1×	1×	1×	100×	1.47×

CIFAR10		Baseline	Early stopping	AdvReg	MemGuard	***NeuGuard***
Testing accuracy		76.6	71.1	71.1	76.6	74.6
MI accuracy	Unsorted NSH	71.70	60.07	61.20	51.43	51.57
	Sorted NN	70.59	57.47	56.18	62.73	55.60
Training time(s)		0.017	0.017	0.050	0.017	0.046
Training overhead		1×	1×	2.94×	1×	2.71×
Inference time(s)		0.017	0.017	0.017	1.7	0.027
Inference overhead		1×	1×	1×	100×	1.59×

Attack setup. We consider the existing two types of attack models: neural network (NN) based attack and metric based attack. We follow the model structure and standard setup in [16, 31] that use different fully connected neural network as attack classifier. The sorted NN attack classifier contains three hidden layers with 512,256 and 128 neurons, respectively. The unsorted NSH attack classifier consist of three neural networks as introduces in Section 2.1.1. Please also see the detailed description in Appendix A.3.

The metric-based MIAs use the output confidence score vector to calculate metrics and compare the results with preset thresholds to determine the prediction. We evaluate our work with four state-of-the-art metrics for MIAs following the approach proposed in [41], which includes prediction correctness, prediction confidence, prediction entropy and modified prediction entropy (see Section 2.1.2). Moreover, we follow the strongest label attack from [5] that use C&W attack to generate adversarial examples to further demonstrate the effectiveness of our defense method.

In our evaluation, we follow [31, 32], which implies a strong adversary that knows a substantial part of the training set and will use it to train the inference attack models. In particular, we sample the input data from training set and testing set with an equal 0.5 probability following prior works [40, 42, 57] to maximize the uncertainty of membership inference attacks.

Evaluation metrics. We use three metrics to evaluate defenses.

- **Membership inference (MI) accuracy:** It is the accuracy of an attack to infer the membership of data in the training set. A good defense should lead the membership inference attack accuracy to a random guess (e.g. ~ 50%).

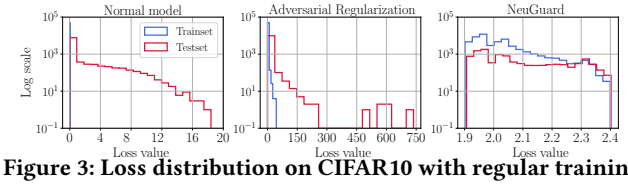


Figure 3: Loss distribution on CIFAR10 with regular training, adversarial regularization training, and our *NeuGuard* .

- **Testing accuracy:** It is the accuracy of a defense method on the testing set. A good defense should obtain the testing accuracy close to the trained model without defense.

  Running time: We measure the model training time and inference time of the defense methods and set the time used for a regular model as the baseline. A good defense should not cause large overhead compare to the baseline.

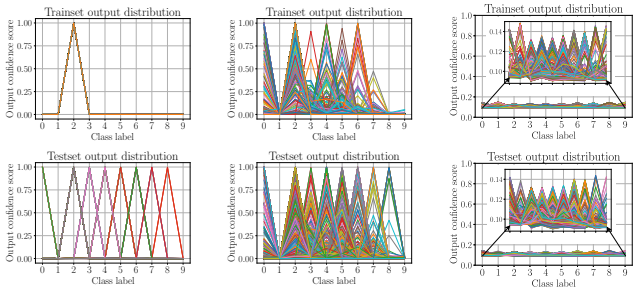
6 EVALUATION ON NN BASED ATTACKS

In this section, we evaluate our proposed defense against neural network based MIAs (see Section 2.1.1). For comprehensive evaluations, we consider the existing two well-known attack models: *sorted attack* and *unsorted attack*.

6.1 Experimental results comparison

Table 4 shows the testing accuracy, MI accuracy, and running time of the compared defenses on the three benchmark datasets. Baseline indicates the regular training method without any defense. We have the following observations:

Our *NeuGuard* achieves the best utility-privacy trade-off against both the sorted and unsorted attacks among evaluated solutions, indicating by far the best defense effectiveness and generality while maximizing the achievable utility. As Table 4 shows, *NeuGuard* effectively mitigates both the unsorted NSH attack and sorted NN attack. Specifically, with *NeuGuard* , the unsorted attack only achieves an MI accuracy close to random guessing on all the three datasets. For the sorted NN attack, *NeuGuard* can significantly decrease the MI accuracy from 60.98%, 81.42%, and 70.59% to 54.54%, 57.82% and 55.6% on the three datasets, respectively. Moreover, the testing accuracy obtained by *NeuGuard* is close to those obtained by the Baseline method without defense. We also observe that while **MemGuard** can maintain the same testing accuracy, it cannot defend against both kinds of NN based attacks simultaneously. To comprehensively compare MemGuard with our defense approach, **1)** we obtain its defense results using an unsorted NSH attack classifier as the target defense model to generate the noised output confidence scores. As expected, MemGuard achieves comparable results as our *NeuGuard* in defending against the unsorted NSH attack. However, it is not effective enough against the sorted NN attack, e.g., 60% MIA accuracy on all the three datasets. **2)** For the original design using a sorted NN attack classifier as the defense model, MemGuard exhibits a similar behavior: defending against sorted NN attack to $\sim$ 50% but $>$ 60% for unsorted NSH attack. We further show the output confidence scores distribution in this case in Fig. 4(a) and analyze the reasons why this is not a general defense in the following section. **AdvReg and Early stopping** show similar defense effects on both attacks in most cases, which is also verified in [41]. Comparing with *NeuGuard* , they achieve the similar level of defense effectiveness against the sorted NN attack, but perform much worse against the unsorted NSH attack on the three datasets. In addition, they also incur a non-negligible (more prominent) utility loss, i.e., large testing accuracy gaps between them and the Baseline method.



(a) MemGuard (b) Adversarial regularization (c) *NeuGuard*

Figure 4: Output distribution of training samples and testing samples with class "2" in CIFAR10 for the compared defenses. Each color line represents one data sample's output confidence score vector.

Our *NeuGuard* has smaller overhead and better performance. For the training and inference overhead, we compare our method with AdvReg and MemGuard separately. In Table 4 we calibrate the training and inference time for processing a single batch of data and use the regular training and inference time as the baselines for overhead comparisons. As the data loader used to load each batch of images dominates the processing time[56], the training and inference time on CIFAR100 and CIFAR10 are the same in the table. Specifically, since we apply layer-wise amplification in both training and inference stages, this incurs some overheads.

In the fully connected neural network case on the Texas100 dataset, our *NeuGuard* training method has $\sim$ 7.5$\times$ overhead while AdvReg training method causes $\sim$ 54.7$\times$ overhead compared to the regular training. As for the inference stage, the MemGuard method causes $\sim$ 900$\times$ overhead as it might need to run several hundred times of inference to search the noisy output confidence vector satisfying their requirements. Our methods, however, leads to no overhead since we have no additional operation in the inference.

For convolutional neural network models, the training overhead incurred by our defense approach and the other approaches are all fairly close to each other. The overhead caused by our method in the inference stage is much less than the MemGuard method (1.47$\times$ and 1.59$\times$ vs. 100$\times$ and 100$\times$ in CIFAR100 and CIFAR10 cases). This is mainly because our layer-wise amplification is a simple one-time process added in the inference stage. In conclusion, our defense can achieve the best defense effectiveness with a marginal model accuracy drop. The overheads caused in training and inference are relatively low compared to other defense mechanisms.

6.2 Why does *NeuGuard* perform better?

One key reason that our defense obtains the best utility-privacy trade-off is because it generates the output confidence scores of the training set and testing set with the smallest variance. Table 5 shows the variance of the output confidence scores calculated on the training set and testing set for CIFAR100 of different methods. We observe the similar results for Texas100 and CIFAR10.

***NeuGuard* obtains the smallest variance of the output confidence scores.** Our defense reduces the variance of the output confidence score to three orders of magnitude compared to the baseline model, while all other defense methods have the same

Table 5: Variance of the output confidence scores on the training set and testing set for CIFAR100.

Model	Baseline	Early stopping	AdvReg	MemGuard	*NeuGuard*
Training set	5.37E-03	4.57E-03	6.99E-03	4.30E-03	**4.44E-06**
Testing set	4.08E-03	3.48E-03	5.89E-03	3.52E-03	**3.88E-06**

level, and we narrow the gap between training set and testing set as shown in Table 5. The results show the positive effects of our *NeuGuard* to reduce the variance of the final outputs.

***NeuGuard* delivers the most consistent loss distribution between the training set and testing set.** Fig. 3 illustrates the loss distributions of the training set and testing set for CIFAR10 under different training methods. Here we use CIFAR10 as an example, since we observe the similar trend on Texas100 and CIFAR100. With the normal training in the baseline method, the training set accuracy will reach ~ 100% when model is overfitted and the loss distribution ranges from 0 to 0.07 while that of the testing set ranges from 0 to 18. AdvReg attempts to reduce the gap between training set and testing set by reducing the training set accuracy and leading the loss distribution range to [0, 50] for the training set. However, the noise introduced by AdvReg also causes a test accuracy drop and significantly increases the range of the testing set loss to [0, 750]. In contrast, our *NeuGuard* constructs two similar loss distributions for both the training set and the testing set, and restricts them to a very small range–[1.9, 2.5].

Visualizing data samples' output confidence scores. To better understand the different defense results among the compared defenses, we further visualize the output confidence score vectors of the training and testing samples generated by these defenses. Specifically, we randomly choose a class (class "2" in our experiment) from CIFAR10 and show the output confidence score vectors of 1000 training and testing samples each from this class in Fig. 4. Note that each color line corresponds to one sample's output confidence score vector (each has ten scores for ten classes). We have the following observations:

First, MemGuard with the sorted NN attack classifier as defense model can construct two similar output distributions for the training set and testing set, i.e., the output confidence score vector has one extremely large score for one class and almost equally small scores for all the other classes and most of the outputs overlap as Fig. 4 (a) shows. In this case, by sorting the confidence score vectors for training and testing samples, the output distributions of training set and testing set are almost the same. This observation explains why MemGuard in this case can defend against the sorted attacks. However, without sorting, the two output distributions are unlike, i.e, the classes associated with the largest output confidence scores are different (see Fig. 4 (a)) and the unsorted attacks can capture this information to perform the attack. Thus, MemGuard cannot effectively defend against unsorted attacks.

Second, AdvReg perturbs the training samples during training and attempts to make the training set output distribution similar to the testing set output distribution. However, as we observe from Fig. 4 (b), the two distributions are not close enough. Therefore, besides suffering from testing accuracy loss, AdvReg neither exhibits strong defense effectiveness.

Third, unlike MemGuard and AdvReg, with the class-wise variance minimization and layer-wise balanced output control, *NeuGuard* controls and orchestrates the final output neurons' results

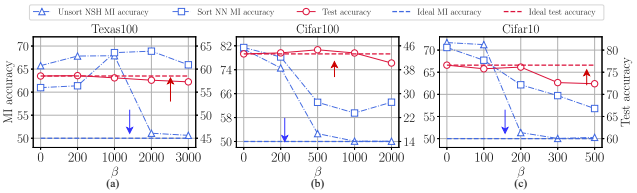


Figure 5: Test accuracy and NN based MI accuracy vs. β on (a)Texas100, (b)CIFAR10, and (c)CIFAR100. *NeuGuard* only uses class-wise variance minimization.

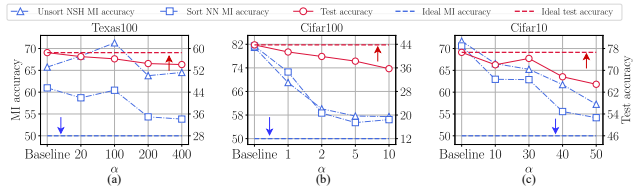


Figure 6: Testing accuracy and NN-based MI accuracy with different α for layer-wise balanced output control term. (a) Texas100, (b)CIFAR10, and (c)CIFAR100.

and intermediate neurons' results for a destined output (confidence score) distribution. As a result, *NeuGuard* can generate the targeted output confidence score vectors for all training and testing samples, where all values in the score vector are close to the mean value, i.e., one over the number of total classes, as shown in Fig. 4(c). Such results have two important implications: i) The output confidence scores of member and non-member training samples are similar, making *NeuGuard* effective to defend against (both sorted and unsorted) membership inference attacks; ii) The output distributions of training samples and testing samples are very close, meaning that *NeuGuard* has good generalization ability. Therefore, *NeuGuard* obtains the best utility-privacy trade-off.

6.3 Hyperparameters setting and ablation study

6.3.1 Parameter setting rules. Hyperparameter setting is essential to achieve effective defensive training by *NeuGuard* . We provide a general guideline to tune the key parameters–α, β for layer-wise balanced output control and class-wise variance minimization. 1) for β which minimizes the variance, its initial value can be set as 10 times #classes, e.g. $\beta = 100$ (1000) for CIFAR10 (CIFAR100/Texas100). Then the maximum output confidence score $F(x)_{max}$ at each training batch can be obtained to guide the tuning: increase β if $F(x)_{max}$ diverges rapidly from mean, and decrease β if accuracy remains unchanged after first several epochs. 2) The initial values of α can be set to $\alpha = 1, 10, 100$ for different models. We adopt the similar parameter tuning approach that is used for β. Through above steps, hyperparameters can easily enter into a suitable range offering reasonable defense results and less sensitive to their actual values. In the following, we investigate the individual contribution to defense of these two regularization terms through an ablation study.

6.3.2 Effectiveness of class-wise variance minimization. Fig. 5 shows the impact of β on the testing accuracy and (both sorted and unsorted) MI accuracy obtained by our *NeuGuard* , where only the class-wise variance minimization is used. We select β roughly based on the number of classes in each dataset, and β trades off the utility (i.e., testing accuracy) and privacy (i.e., MI accuracy). Specifically, when β becomes larger, the class-wise variance should be smaller. This can enforce that all values in output confidence score vectors

Table 6: Evaluation on metric based MI attacks.

Texas100		Baseline	MemGuard	Early stopping	AdvReg	***NeuGuard***
Testing accuracy		**56.0**	**56.0**	54.5	54.0	54.4
Accuracy gap		30.4	30.4	26.1	29.0	**19.6**
MI accuracy	Correctness	65.2	65.2	63.1	64.5	**59.8**
	Confidence	69.1	68.1	65.8	66.6	**62.6**
	Entropy	62.2	60.6	60.5	60.6	**54.4**
	Modified entropy	69.1	68.4	65.7	66.8	**62.7**

CIFAR100		Baseline	MemGuard	Early stopping	AdvReg	***NeuGuard***
Testing accuracy		**43.8**	42.9	41.8	39.7	43.0
Accuracy gap		56.2	30.3	24.2	25	**17.4**
MI accuracy	Correctness	78.1	65.1	61.9	62.5	**58.7**
	Confidence	84.8	65.0	62.6	63.1	**59.7**
	Entropy	80.8	56.1	59.6	**55.5**	55.7
	Modified entropy	84.8	64.9	62.9	63.1	**59.6**

CIFAR10		Baseline	MemGuard	Early stopping	AdvReg	***NeuGuard***
Testing accuracy		**76.6**	**76.6**	74.1	71.1	74.6
Accuracy gap		23.4	23.4	22.2	21.0	**17.8**
MI accuracy	Correctness	61.7	61.7	61.1	60.5	**58.9**
	Confidence	72.3	64.8	63.3	61.0	**60.4**
	Entropy	70.5	58.6	60.4	58.0	**55.9**
	Modified entropy	72.1	65.0	63.2	61.1	**60.4**

are closer, and thus reveal less information to the attacker, lowering the attack accuracy in general. On the other hand, similar values make output confidence score vectors not discriminative enough, which could negatively affect the testing accuracy. In summary, while the class-wise variance minimization provides an effective defense under certain conditions, that alone is insufficient for good generalizability against a range of attacks, e.g. far better defense against Unsorted NSH attack than Sorted NN attack.

6.3.3 Effectiveness of the layer-wise balanced output control. Fig. 6 (a), (b) and (c) show the test accuracy and NN-based MI accuracy when we only deploy the layer-wise balanced output control with different αs for Texas100 dataset (using the fully connected neural network), CIFAR-100 and CIFAR-10 datasets (using CNNs), respectively. Here the convolution layers' output balancing also involves feature amplification, as discussed in Sec. 4.4. Its impact is discussed in detail in Appendix A.4. We can observe that, unlike the results of only applying the variance minimization in Fig. 5, applying layer-wise balanced output control generally can have more impact on the defense against the sorted NN attacks. Particularly, it has stronger constraint on producing balanced intermediate results in each layer as α grows. This restricts the impact of some model parameters that may originally dominate the output prediction, in a layer-wise manner. Then the difference between the output predicted by the training data and testing data can be further reduced.

6.3.4 Summary. The above analysis shows that our class-wise variance minimization contributes more on defending against the unsort NSH attack and layer-wise balanced output control can further reduce the sort NN attack. Therefore, *NeuGuard* consisting of these two terms can better defend against both kinds of NN based MIAs as demonstrated in Table 4.

7 EVALUATION ON METRICS ATTACKS

In this section, we further evaluate our defense against the latest metrics based MI attacks [5, 41], as introduced in Sec. 2.1.

Experimental results comparison Table 6 reports the testing accuracy, accuracy gap between training and testing set, attack accuracy based on the four metrics (i.e., prediction correctness, prediction confidence, prediction entropy and modified prediction entropy) on CIFAR100, CIFAR10, and Texas100, respectively. For these attacks, we have the following observations:

Table 7: Euclidean Distance, KL Divergence, and TV Distance of the empirical output distributions between the training set and testing set after the modified entropy attack is applied.

	Texas100			**CIFAR100**			**CIFAR10**		
Metrics	Euc	KL	TV	Euc	KL	TV	Euc	KL	TV
Baseline	92.88	0.2971	0.3265	132.44	0.8487	0.5829	324.04	0.3147	0.2700
AdvReg	100.41	0.2441	0.2964	1127.68	0.3013	0.3092	76.97	0.1467	0.1864
MemGuard	654.16	0.2645	0.2854	649.26	0.2896	0.3148	323.57	0.3229	0.2625
Early stopping	72.21	0.2142	0.2717	104.19	0.1462	0.2212	182.77	0.2198	0.2435
NeuGuard	**7.39**	**0.0943**	**0.1636**	**11.93**	**0.0681**	**0.1578**	**6.68**	**0.1151**	**0.1840**

Our *NeuGuard* always achieves the best results. This conclusion is consistent with the results on NN based attacks, demonstrating the superior generality and scalability of our *NeuGuard*.

Our *NeuGuard* has the smallest gap between training accuracy and testing accuracy, which further implies that the output distribution of training set and that of testing set are the most similar. Here, we further propose to verify this claim quantitatively. Specifically, we adopt three metrics, i.e., Euclidean distance (*Euc*)[46], Kullback–Leibler divergence (*KL*)[26], Total Variation Distance (*TV*)[23], which are used to measure the similarity between two distributions—a smaller value of these metrics indicates a larger similarity of two distributions. We denote the two probability distributions as P and Q, respectively. For P and Q in the same probability space Ω, we have $KL(P\|Q) = \sum_{\omega\in\Omega} P(\omega)\log\left(\frac{P(\omega)}{Q(\omega)}\right)$ and $TV(P,Q) = \frac{1}{2}\sum_{\omega\in\Omega}|P(\omega) - Q(\omega)|$.

In our evaluation, we use histograms to calculate the probability distribution of the training set P and that of the testing set Q after the strongest modified entropy attack is applied (One can see that all defenses achieved the worst performance against the modified entropy attack in Table 6). Table 7 displays the results of the three metrics on the three datasets. All the results are obtained by selecting the best-performing model for each defense. We can observe that our *NeuGuard* has the lowest values evaluated by all the three metrics and significantly outperforms the compared defenses. These results indicate that our *NeuGuard* can produce models with smallest difference between the training set and testing set outputs. Thus, our defense has the best effect on metric based MIAs.

***NeuGuard* delivers the best defense effectiveness against the strong C&W label-only attack.** Table 8 compares the testing accuracy (utility), the prediction correctness attack accuracy (MI correction as a baseline), and C&W label attack accuracy among different defenses. Again, *NeuGuard* outperforms all other defenses while offering great utility (e.g. closer to baseline) against the C&W label attack. The C&W attack accuracy is even 3.6% (CIFAR10) and 3.4% (CIFAR100) lower than the correctness attack baseline. This is because *NeuGuard* refines model parameters during the training and assures more evenly distributed output confidence scores. As a result, the needed adversarial perturbations to generate adversarial examples that alter prediction results to untargeted incorrect labels become similar for the training and testing data.

Defenses against metrics based attacks are not as effective as against NN based attacks. When comparing the results in Table 6 with those in Table 4, we can see that metrics based attacks (except entropy) have larger MI accuracies than NN based attacks. This is because the performance of defenses against metrics based attacks, except entropy, are bounded by the accuracy gap between the training set and the testing set. When the accuracy gap is larger, all the defenses would achieve worse performance.

Table 8: The results of C&W label-only attack on CIFAR10 and CIFAR100 dataset with different defense methods.

Dataset	Accuracy	Baseline	MemGuard	Early stopping	AdvReg	*NeuGuard*
CIFAR 10	Testing dataset	**76.6**	**76.6**	74	71.9	74.6
	MI correctness	61.7	61.7	59.1	**58.1**	58.9
	C&W label attack	69.2	69.2	59.7	59.2	**55.3**
CIFAR 100	Testing dataset	**44.8**	**44.8**	41.6	39.7	43
	MI correctness	77.5	77.5	61.6	64.6	**57.8**
	C&W label attack	80.9	80.9	61.7	63.3	**54.4**

The accuracy gap fully determine the prediction correctness attack, more detailed description and mathematical analysis can be found in Appendix A.5. In other words, **as long as the accuracy gap exists, no defense can reduce the MI accuracy obtained by the prediction correctness attack to 0.5, i.e., random guessing.** The best defense performance we can achieve is to bring down the MI accuracy based on confidence prediction and modified prediction entropy attacks close to the correctness attack accuracy. We analyze two corner cases to further illustrate these metric-based attacks are bounded by the prediction correctness in Appendix A.5.

8 RELATED WORK

Many methods have been proposed to defend against membership inference attacks (MIAs). We category existing defenses as training time based defense (e.g., dropout [38], L_2 norm regularization [40], model stacking [38], label smoothing [45], min-max adversary regularization (AdvReg) [31], differential privacy [1, 29], early stopping [41], knowledge distillation [39]) and inference time based defense (e.g., output perturbation (MemGuard) [16]).

Label Smoothing (LS) [45] is a hard label augmentation approach to reduce model overfitting by assigning uniform probabilities to classes. However, studies [18, 30] show that LS pushes the model to output smoother and more uniform probabilities for the training data, and erases less information on testing data than training data. As a result, LS leads to greater discrepancies between the training data and testing data predictions, and thus increases model's MIA vulnerabilities. However, *NeuGuard* exploits privacy-dedicated regularizations to constrain inner and output neurons with the goal of restricting the value range of output prediction during training. This explicitly reduces the prediction difference between training set and test set, and thus can defend against MIAs.

Differential privacy (DP) [7] is a probabilistic privacy mechanism that provides an information-theoretical privacy guarantee. Many works aim to integrate DP into machine learning as a general approach to provide theoretical privacy guarantee for the models[1, 36]. The basic idea is to add noise to the gradient used for the stochastic gradient descent [1, 43, 58] or the objective function for the model learning [14] to achieve the differential privacy. The main problem of the differential privacy mechanism is that it cannot provide a satisfactory privacy-utility trade-off. As evaluated in [15, 36, 39] and discussed in [16], while a well founded privatization technique like DP has shown "middling" performance against MIAs, the model utility is fairly low when it is able to effectively defend against MIAs. For example, [39] showed that DP-SGD [1] only delivers a $\sim$ 52.2% testing accuracy for the Alexnet-CIFAR10 setting when the MIA accuracy is $\sim$ 51.7%, while *NeuGuard* gives a $\sim$ 74.6% testing accuracy at a similar level of defense efficiency.

Knowledge distillation [13, 19] uses teacher-student models for model training. Distillation For Membership Privacy (DMP)[39] firstly trains a teacher model with unprotected data and then trains the target model using an unlabeled reference dataset that aims to obtain a similar prediction entropy with the private training data. With limited access to the private data, the trained student model should not leak information. Compared to our *NeuGuard* , this design requires more complex training procedures and additional reference datasets to facilitate the knowledge distillation between the teacher model and student model. However, public reference datasets from the same domain may not be available in real-world applications, especially in fields like medical and financial analysis, where the data are private and confidential. Furthermore, one alternative solution to this is to generate synthetic data using the generative adversarial networks (GANs). However, [39] shows that it would result in significant model utility drop. [6], [47] improve the defense without requiring public data, but they require more complex training strategies, different training phases for teacher and student models, and require sufficient training data since they need to split data into several parts for training.

AdvReg [31] regularizes the training process by solving a min-max problem. However, its has non-negligible utility loss and the computational overhead is significant. MemGuard[16] is the only inference time based defense. It aims to add crafted noise in the final outputs to mislead the membership inference attacks (MIAs). *NeuGuard* controls both the final output and the inner neurons, and shows better generalizability and much less overhead.

In addition to MIAs for the traditional NN models, many studies aim at attacks and defenses on supervised classification tasks in federated learning (FL) setting[12, 28, 51] and centralized unsupervised learning on Generative Adversarial Networks (GANs)[4, 11, 25]. Besides, there are many other attacks that aim to leak the privacy information from the ML models. For instance, attribute inference attacks [10, 17] and hyperparameter stealing attacks [53] leverage the ML classifier to infer users' private attributes. Model inversion attack[8, 37, 55] tries to recover recognizable training data from the target model. Model stealing attack [49] aims to extract the learned parameters of ML models by solving a series of linear equations.

9 CONCLUSION

We explored state-of-the-art defenses and demonstrated that they are unable to effectively defend against existing MIAs, especially sorted and unsorted NN attacks. We advocate that a more effective defense is to orchestrate the output of the training set and testing set for the same explicitly designed distribution that is more evenly distributed in a restricted small range. To achieve this goal, we propose a simple yet effective defense mechanism–*NeuGuard* built upon the technique of fine-grained neuron-level regularization, to simultaneously control and guide the final output neurons and hidden neurons towards constructing a defensive model. *NeuGuard* consists of a class-wise variance minimization method and layer-wise balanced output control method to regularize output and inner neurons in a layer-wise manner. We demonstrate the effectiveness of *NeuGuard* on three different datasets against not only two NN based MIAs, but also five (strongest) metrics based MIAs including the label-only attack. We further discuss the defense upper bound of the metrics based MIAs through theoretical and experimental analysis. With a flexible parameter control, *NeuGuard* always offers the best utility-privacy trade-off with much lower overhead, comparing with all evaluated defenses.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*. PMLR, 274–283.

[3] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*.

[4] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*.

[5] Christopher A Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-only membership inference attacks. In *International Conference on Machine Learning*. PMLR, 1964–1974.

[6] Rishav Chourasia, Batnyam Enkhtaivan, Kunihiro Ito, Junki Mori, Isamu Teranishi, and Hikaru Tsuchida. 2022. Knowledge Cross-Distillation for Membership Privacy. *Proceedings on Privacy Enhancing Technologies* 2022, 2 (2022), 362–377.

[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[8] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

[9] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. 2014. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium*.

[10] Neil Zhenqiang Gong and Bin Liu. 2016. You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 979–995.

[11] J Hayes, L Melis, G Danezis, and E De Cristofaro. 2019. LOGAN: Membership Inference Attacks Against Generative Models. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, Vol. 2019. De Gruyter, 133–152.

[12] Qijian He, Wei Yang, Bingren Chen, Yangyang Geng, and Liusheng Huang. 2020. Transnet: Training privacy-preserving neural network over transformed layer. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1849–1862.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[14] Roger Iyengar, Joseph P Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. 2019. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 299–316.

[15] Bargav Jayaraman and David Evans. 2019. Evaluating differentially private machine learning in practice. In *28th {USENIX} Security Symposium*.

[16] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. Memguard: Defending against black-box membership inference attacks via adversarial examples. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 259–274.

[17] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. Attriinfer: Inferring user attributes in online social networks using markov random fields. In *Proceedings of the 26th International Conference on World Wide Web*. 1561–1569.

[18] Yigitcan Kaya and Tudor Dumitras. 2021. When Does Data Augmentation Help With Membership Inference Attacks?. In *International Conference on Machine Learning*. PMLR, 5345–5355.

[19] Nikos Komodakis and Sergey Zagoruyko. 2017. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *ICLR*.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012), 1097–1105.

[22] Klas Leino and Matt Fredrikson. 2020. Stolen memories: Leveraging model memorization for calibrated white-box membership inference. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1605–1622.

[23] David A Levin and Yuval Peres. 2017. *Markov chains and mixing times*. Vol. 107. 2nd. rev. ed. (AMS, 2017), Proposition 4.2, p. 48. American Mathematical Soc.

[24] Zheng Li and Yang Zhang. 2021. Membership leakage in label-only exposures. In *Proceedings of the 2021 ACM SIGSAC CCS*. 880–895.

[25] Kin Sum Liu, Chaowei Xiao, Bo Li, and Jie Gao. 2019. Performing co-membership attacks against deep generative models. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 459–467.

[26] David JC MacKay and David JC Mac Kay. 2003. *Information theory, inference and learning algorithms*. Cambridge university press.

[27] National Defense Magazine. [n.d.]. . https://www.nationaldefensemagazine.org/.

[28] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 691–706.

[29] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 19–38.

[30] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in Neural Information Processing Systems* (2019).

[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 634–646.

[32] Nasr, Milad and Shokri, Reza and Houmansadr, Amir. 2019. 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, 739–753.

[33] Utku Ozbulak, Arnout Van Messem, and Wesley De Neve. 2019. Impact of Adversarial Examples on Deep Learning Models for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 300–308.

[34] Elbruz Ozen and Alex Orailoglu. 2020. Concurrent monitoring of operational health in neural networks through balanced output partitions. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 169–174.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[36] Md Atiqur Rahman, Tanzila Rahman, Robert Laganière, Noman Mohammed, and Yang Wang. 2018. Membership Inference Attack against Differentially Private Deep Learning Model. *Trans. Data Priv.* 11, 1 (2018), 61–79.

[37] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. 2020. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th {USENIX} Security Symposium*.

[38] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246* (2018).

[39] Virat Shejwalkar and Amir Houmansadr. 2021. Membership Privacy for Machine Learning Models Through Knowledge Transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 9549–9557.

[40] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18.

[41] Liwei Song and Prateek Mittal. 2021. Systematic evaluation of privacy risks of machine learning models. In *30th {USENIX} Security Symposium*.

[42] Liwei Song, Reza Shokri, and Prateek Mittal. 2019. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 241–257.

[43] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 245–248.

[44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[45] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[46] John Tabak. 2014. *Geometry: the language of space and form*. Infobase Publishing.

[47] Xinyu Tang, Saeed Mahloujifar, Liwei Song, Virat Shejwalkar, Milad Nasr, Amir Houmansadr, and Prateek Mittal. 2021. Mitigating Membership Inference Attacks by Self-Distillation Through a Novel Ensemble Architecture. *arXiv preprint arXiv:2110.08324* (2021).

[48] Texas. 2017. *Texas hospital stays dataset*. https://www.dshs.texas.gov/THCIC/Hospitals/Download.shtm.

[49] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 601–618.

[50] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2018. Towards demystifying membership inference attacks. *arXiv preprint arXiv:1807.09173* (2018).

[51] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2019. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing* (2019).

[52] Allyson Versprille. 2015. Researchers hack into driverless car system, take control of vehicle. *National Defense Magazine* (2015).

[53] Binghui Wang and Neil Zhenqiang Gong. 2018. Stealing hyperparameters in machine learning. In *2018 IEEE Symposium on Security and Privacy (SP)*.

[54] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. 2020. Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243* (2020).

[55] Nuo Xu, Qi Liu, Tao Liu, Zihao Liu, Xiaochen Guo, and Wujie Wen. 2020. Stealing your data from compressed machine learning models. In *2020 57th ACM/IEEE*

Algorithm 1 Loss calculation using proposed method

Input: ML model F, a batch of data (x_B, y_B) with N records, class-wise mean list vector μ_y, model layer number M
Output: Loss value $Loss$ calculated for this batch
$\{outputs, h^1, h^2, ..., h^{M-1}\} = F(x_B)$
$softout = softmax(outputs)$
for l in $M-1$ **do**
    $L_{boc} = L_{boc} + \frac{1}{S_l} \left\| \sum_{i=1}^{\lfloor S_l/2 \rfloor} h_i^l - \sum_{i=\lfloor S_l/2 \rfloor+1}^{S_l} h_i^l \right\|_F^2$
end for
for i in N **do**
    $count_{y_i} += 1$
    $\mu_{y_i} = \mu_{y_i} \frac{count_{y_i}-1}{count_{y_i}} + \frac{softout_i}{count_{y_i}}$
end for
$L_{var} = \frac{1}{n} \sum_{i=0}^{n} (F(x_i) - \mu_y)^2$
$Loss = criterion(x_B, y_B) + \alpha \times L_{boc} + \beta \times L_{var}$

Design Automation Conference (DAC). IEEE, 1–6.

[56] Chih-Chieh Yang and Guojing Cong. 2019. Accelerating data loading in deep neural network training. In *2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 235–245.

[57] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE, 268–282.

[58] Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially private model publishing for deep learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 332–349.

[59] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).

[60] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. 2020. Neuron-level Structured Pruning using Polarization Regularizer.. In *NeurIPS*.

A APPENDICES

A.1 Datasets

We use the following three benchmark datasets to demonstrate the effectiveness of our *NeuGuard* against membership inference attack for different application scenarios.

Texas100 [48] is a dataset generated from Hospital Discharge Data Public Use Data File that contains information about inpatients stays in several health facilities. The data is published by the Texas Department of State Health Services (DSHS) and we grant the preprocessed dataset from [40]. This dataset contains 67,330 data records with 6,170 binary features. These features indicate the external causes of injury (e.g., drug misuse and suicide), the diagnosis (e.g., schizophrenia, illegal abortion), the procedures the patient underwent (e.g., surgery), and generic information such as gender, age, race, hospital ID, and length of stay. The records are clustered into 100 classes representing the 100 most frequent medical procedures. Following the existing methods [16, 31, 32], we use 10,000 data records for training and 57,330 data records for testing.

CIFAR10 and **CIFAR100** are benchmark datasets widely used in image classification tasks [20]. Specifically, CIFAR-10 consists of 32×32 color images from 10 classes and each class contains 6,000 images. It includes 50,000 training images and 10,000 testing images. CIFAR100 contains the same size color images from 100 non-overlapping classes, each with 500 training images and 100 testing images. Table 9 summarizes the statistics of these datasets.

Table 9: Dataset split configurations.

Dataset	Training set	Testing set	Training members	Training non-members
CIFAR10	50,000	10,000	25,000	5,000
CIFAR100	50,000	10,000	25,000	5,000
Texas100	10,000	57,330	5,000	10,000

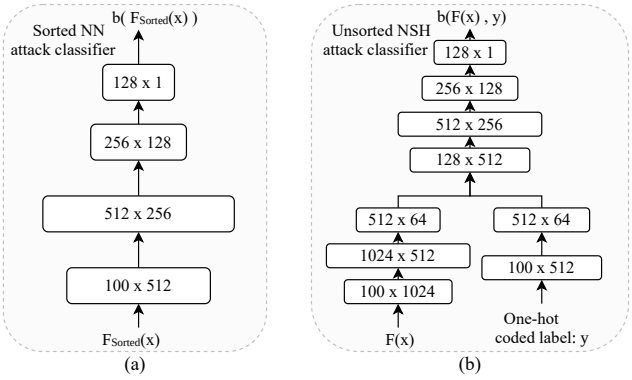


Figure 7: The attack classifier architecture of Sorted NN and Unsorted NSH attack used in the evaluation

A.2 Parameter setting

For the Texas100 classification task, we use a fully connected neural network with four hidden layers, which have layer sizes 1024, 512, 256, 128, respectively. We use the Tanh activation function for the hidden layers and use the softmax function for the final layer. We use the cross-entropy loss function and Adam optimizer to train the model. We train the model with an initial learning rate 0.001 and a decay by 0.1 in every 20 epochs.

For CIFAR10 and CIFAR100 image classification tasks, we use Alexnet[21], which is a convolutional neural network with parameters trained with a cross-entropy loss function and stochastic gradient descent (SGD) optimizer. We set the initial learning rate as 0.01 and decay by 0.1 in every 20 epochs.

We follow the training method and hyperparameter setup proposed by the authors and use the published code to evaluate their defense mechanism. The adversarial regularization parameter is set as $\lambda = 3$ for Texas100 model and $\lambda = 6$ for CIFAR10 and CIFAR100. The Early stopping model is introduced following the [41] to show the defense efficiency of the regular trained model with the similar test accuracy of the AdvReg trained model. We evaluate the MemGuard defense by using Sorted NN model [16] and Unsorted NSH model [31] as its defense classifiers to generate the noisy output, respectively. Since in some cases, the defense classifier with Sorted NN attack model under the constraint of MemGuard, is unable to produce effective noisy output for defense, we select the Unsorted NSH attack model which offers better defense effectiveness as the classifier, for fairly comparing the defense efficiency.

A.3 Neural network based attacks setup

We summarize neural network based attacks as *sorted attack* and *unsorted* attack. The major difference is whether the output confidence score used by the the attack model is sorted or not.

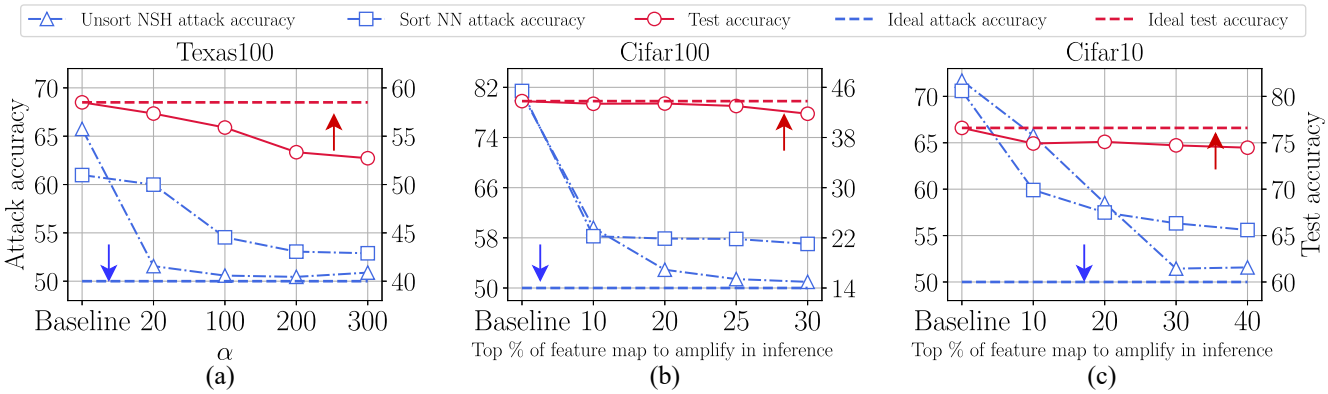


Figure 8: Testing accuracy and NN-based MI accuracy with different hyperparameters for layer-wise operations. For the fully connect model for Texas100 classification, we set $\beta = 3,000$ in the class-wise variance minimization regularization, and we change the α value in (a) to show the effects of the layer-wise balanced output control. For convolutional models, we set $\beta = 200$ on CIFAR10 and $\beta = 1,000$ on CIFAR100. We amplify the top 10% feature map values for 1.5 and 2 times, respectively. (b) and (c) show the results with different selected top feature map values during the inference on CIFAR100 and CIFAR10.

Sorted attack: For the Sorted NN attack, we adopt the standard setting in [16] and use a three-layer fully connected neural network as the attack classifier. See Fig. 7(a).

Unsorted attack: To evaluate the Unsorted NSH attack, we follow the model structure and setup in [31] to construct and train the attack classifier. The Unsorted NSH attack classifier takes two pieces of information as input. One is the unsorted confidence score vector, and the other one is the label that is one-hot coded (all elements except the one that corresponds to the label index are 0). The classifier consists of three fully connected sub-networks. See Fig. 7(b).

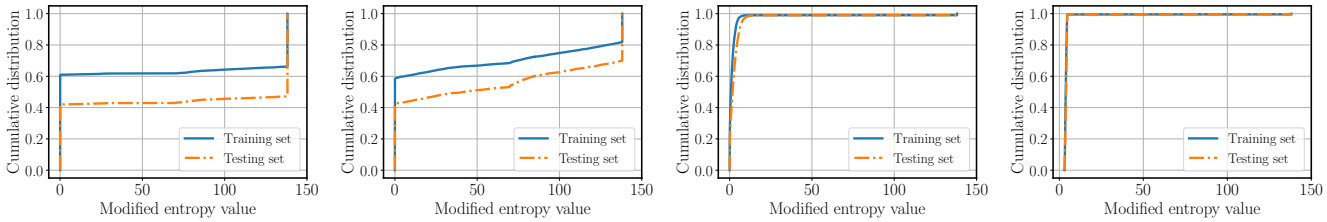
We adopt the Relu activation function for the hidden layers and the sigmoid activation function for the output layer for both attack classifiers. The attack classifier predicts the input as a member if and only if the final prediction probability $b(\cdot) \geq 0.5$; otherwise, it predicts as a non-member. To train the attack classifier, we use the mean squared error (MSE) criterion and Adam optimizer with a learning rate of 0.001. For better convergence, we decay the learning rate by 0.1 in the 40th and 90th epoch for 100 epoch training for the Sorted NN attack, and decay the learning rate by 0.1 in the 30th epoch for 200 training epochs for Unsorted NSH attack.

A.4 Effectiveness of the layer-wise feature map operations

In this section, we show some supplementary results with combined *NeuGuard* training methods and tune one of the component's hyperparameter to further demonstrate the effectiveness of the proposed methods.

Fig. 8 (a) shows the test accuracy and NN-based MI accuracy when we vary α for layer-wise balanced output control on the Texas100 dataset. Here we set $\beta = 3000$ for the class-wise variance minimization. Unlike the results obtained by only applying the variance minimization in Fig. 5 (a), with α growing, the attack accuracies of both Unsorted NSH attack and Sorted NN attack decrease prominently at the cost of limited utility reduction. A larger α indicates a better defense effectiveness but a slightly degraded testing accuracy.

As discussed in Sec. 4.4 and Sec. 6.3.3, we adopt the layer-wise feature map amplification for the convolutional layers in CNNs at both training and inference stages to 1) better assist the proposed learning regularization for converging model output scores towards desired distributions (for defense effectiveness); 2) better maintain model inference accuracy (for model utility). For the hyperparameters setting of feature amplification, we can empirically amplify top $0-10\%$ feature maps to $1-1.5$ times during training guided by the accuracy growth trend. Then at inference we amplify top $0-50\%$ feature maps using the same amplification rate to reduce the accuracy gap with an acceptable utility loss. To validate its effectiveness, we conduct experiments based on the CNN-based CIFAR100 and CIFAR10 image classification tasks and tune the amplified percentage of feature maps during the inference time. As Fig. 8 (b) and (c) show, the test accuracy on CIFAR100 and CIFAR10 drops slightly as the percentage of amplified features increases, while the attack accuracy of both Sorted NN and Unsorted NSH attack decreases more significantly, demonstrating a much improved trade-off between defense effectiveness and model utility using our method. This is because: 1) The impact of the most significant portions of a feature map that are amplified during training decreases as other parts of the feature map is also enlarged due to the increased amplified percentage during inference; 2) The operation is analogous to introducing some noise to model inference, making the output distribution of training set and test set closer and resulting in reduced MI accuracy for both attacks. However, amplifying too many intermediate features will cause more obvious utility drop despite the minor attack accuracy reduction. Therefore, the optimal parameter of our proposed defense can be identified by observing MI accuracy reduction and utility loss, e.g. 25% and 30% for CIFAR100 and CIFAR10 as shown in Fig. 8 (b) and (c).



(a) Normal model (b) *NeuGuard* (c) Normal model (d) *NeuGuard*

Figure 9: Empirical CDF of modified entropy on CIFAR100 with (a) and (b) fully connected layer amplification in inference. Amplify top 50% for 20×; and with (c) and (d) convolution layer amplification in inference. Amplify the top 1% for 1000×.

A.5 Metric based attack defense effectiveness analysis

We first show the close relationship between prediction correctness and accuracy gap between the training set and testing set. Then, we discuss why the defenses cannot reduce the MI accuracy of metrics based attacks to random guessing.

Let d_{tr} and d_{te} denote the number of training samples and testing samples, respectively. Note that in our experiments, we have the same value of d_{tr} and d_{te} and we let $d_{tr} = d_{te} = d$. Moreover, we denote the training accuracy and testing accuracy as Acc_{tr} and Acc_{te}, respectively. Then, we have:

$$\begin{aligned} \mathcal{M}_{\text{corr}}(F; z) = I(\text{argmax}\, F(x) = y) &= \frac{Acc_{tr} \times d_{tr} + (d_{te} - Acc_{te} \times d_{te})}{d_{tr} + d_{te}} \\ &= \frac{d \times (Acc_{tr} - Acc_{te}) + d}{2d} = \frac{1}{2} \times Acc_{gap} + \frac{1}{2} \end{aligned} \tag{9}$$

Eqn 9 implies that the prediction correctness can be completely determined by the accuracy gap. In other words, **as long as the accuracy gap exists, no defense can reduce the MI accuracy obtained by the prediction correctness attack to 0.5, i.e., random guessing.** Furthermore, from Eqn 2 and Eqn 4, we note that **prediction confidence and modified prediction entropy are also highly related to prediction correctness, as they both use the predicted label information.** In an extreme case where the model outputs the same distribution of confidence score vectors, e.g. a single large score with all others being equally small values, regardless of member or non-member data, the calculation of both prediction confidence and modified prediction entropy can converge to two values–one value for all correctly predicted data and another for all incorrectly predicted data, e.g. CDF with only two values in the x-axis as we shall show in Fig. 9(a). As a result, their attack accuracies are the same as that of prediction correctness for any preset threshold chosen between these two values.

The best defense performance we can achieve is to bring down the MI accuracy based on confidence prediction and modified prediction entropy attacks close to the correctness attack accuracy. We explore the two cases on CIFAR100 through the proposed layer-wise intermediate results amplification to demonstrate that the defense performance is upper bounded by applying the prediction correctness attack. We need to point out that this kind of defense is easy to achieve through our layer-wise amplification. In this case, these metric based attacks are equivalent to the prediction correctness attack.

Table 10 shows the testing accuracy and metric based attack results on two cases that lead the attack accuracy of prediction

Table 10: Metric based eval. on two corner cases on CIFAR100.

Model	Baseline FC layer amp	*NeuGuard* FC layer amp	Baseline Conv layer amp	*NeuGuard* Conv layer amp
Test accuracy	43.6	43.4	43	43
Accuracy gap	26.6	17.1	25.6	17.3
Correctness	63.2	58.5	62.8	58.7
Confidence	63.7	58.9	63.5	59.7
Entropy	51.7	55.6	55.5	55.4
Modified entropy	63.2	58.9	63.8	59.7

confidence and modified prediction entropy close to the prediction correctness attack. We apply two amplification strategies to the model and both of them can maintain the test accuracy at the same level as the baseline model. The first strategy is to amplifies top 50% of the intermediate results of the last three FC layers for 20×. Fig. 9(a) and Fig. 9(b) show the absolute modified entropy distribution of this strategy applied on the regular trained model and *NeuGuard* . The output confidence score is amplified to the case that only one label has large value (e.g., close to 1) and all the others have equally small values (e.g., close to 0). In this case, the modified prediction entropy for the correctly classified data is 0 and for the misclassified data is about 138. As the figure shows, the only difference in the modified entropy value is made by the correctness of the model classification. The attack accuracies of prediction correctness, confidence and modified entropy are similar as shown in Table 10 for both the regular trained model and *NeuGuard* .

The second case is explored by amplifying the most significant part of the feature map in the convolution layers while guarantee the model utility. In the experiments, we amplify the top 1% feature map values to 1000× for all the convolution layers. The absolute modified entropy distribution of the normal model and *NeuGuard* are shown in Fig. 9(c) and Fig. 9(d) and the results are in Table 10. The modified prediction entropy has almost no difference for training set and testing set as all of them are close to zero. Nevertheless, the attack accuracy of the modified prediction entropy is still similar to that of the prediction correctness in both the regular trained model and our *NeuGuard* . **This aligns well with our observation that the metric-based attacks are bounded by the prediction correctness that is further determined by the accuracy gap in Eqn 9, despite the indistinguishable prediction entropy.** Minimizing the training and testing accuracy gap is the most viable approach to further improve the defense effectiveness against these attacks, which further explains why our *NeuGuard* always performs the best among existing defense solutions (see Table 6).