Defending Against Membership Inference Attacks on Iteratively Pruned Deep Neural Networks

Jing Shang[§], Jian Wang[§*], Kailun Wang[§], Jiqiang Liu[§], Nan Jiang[†], Md Armanuzzaman[‡], Ziming Zhao[‡*]
[§]Beijing Jiaotong University, Email: {shangjing, wangjian, wangkailun, jqliu}@bjtu.edu.cn
[†]Beijing University of Technology, Email: jiangnan@bjut.edu.cn
[‡]CactiLab, Northeastern University, Email: {m.armanuzzaman, z.zhao}@northeastern.edu

***Abstract*—Model pruning is a technique for compressing deep learning models, and using an iterative way to prune the model can achieve better compression effects with lower utility loss. However, our analysis reveals that iterative pruning significantly increases model memorization, making the pruned models more vulnerable to membership inference attacks (MIAs). Unfortunately, the vast majority of existing defenses against MIAs are designed for original and unpruned models. In this paper, we propose a new framework WeMem to weaken memorization in the iterative pruning process. Specifically, our analysis identifies two important factors that increase memorization in iterative pruning, namely data reuse and inherent memorability. We consider the individual and combined impacts of both factors, forming three scenarios that lead to increased memorization in iteratively pruned models. We design three defense primitives based on these factors' characteristics. By combining these primitives, we propose methods tailored to each scenario to weaken memorization effectively. Comprehensive experiments under ten adaptive MIAs demonstrate the effectiveness of the proposed defenses. Moreover, our defenses outperform five existing defenses in terms of privacy-utility tradeoff and efficiency. Additionally, we enhance the proposed defenses to automatically adjust settings for optimal defense, improving their practicability.**

I. Introduction

Deep neural networks (DNNs) involve millions or even billions of parameters [1]–[4], necessitating significant storage and computing resources. This poses challenges for deploying them on resource-constrained devices [5]–[7]. Model pruning as a compression technique can reduce the model size with little impact on the accuracy by removing redundant model parameters [4], [8]–[10]. Traditional one-shot model pruning [4], [7] consists of training, pruning, and fine-tuning operations. Recent research [8] has shown that an iterative process of repeated pruning and fine-tuning can achieve a better tradeoff between model utility and sparsity.

∗ Corresponding authors

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.240090
www.ndss-symposium.org

Membership inference attacks (MIAs) pose a significant privacy threat to DNNs. Adversaries can exploit these attacks to deduce whether a particular data sample was used in training a model by analyzing the model's output. Such attacks are particularly concerning in sensitive domains like healthcare and finance, where they compromise individual privacy [11], [12]. The underlying issue is the tendency of DNNs to memorize training data details excessively [13], [14], thereby making it simpler for these models to differentiate between member and non-member samples. Furthermore, recent research indicates that one-shot pruned models exhibit a higher vulnerability to MIAs, showing increased attack accuracy compared to the original models [15]. While it is suspected that the increased memorization in DNNs is due to the repetitive use of training data during model fine-tuning, this phenomenon has not yet been quantified.

In this paper, we first quantitatively assess how reusing training data increases memorization in DNNs. We also explore whether iteratively pruned models, which involve multiple reuses of training data, pose a higher privacy risk than one-shot pruned models. As a result, we empirically establish a positive correlation between the occurrences of data reuse and increased memorization. Additionally, we demonstrate that MIA accuracy is notably higher in iteratively pruned models compared to one-shot pruned models. These findings suggest that the repeated data reuse in iterative pruning substantially boosts a model's tendency to memorize training samples, thereby increasing vulnerability to MIAs.

Furthermore, recent work by Feldman et al. [16], [17] argues that certain data are inherently easier to memorize, suggesting that the intrinsic memorability of some data could be a significant factor contributing to increased memorization in iteratively pruned models. We empirically demonstrate that inherently easy-to-memorize training data is more vulnerable to serious privacy threats. The situation worsens during iterative pruning, as data reuse leads to a significant increase in the model's memorization of such data.

Consequently, we present a new framework, namely WeMem, to defend against MIAs in iterative pruning by weakening memorization. We consider three scenarios in iterative fine-tuning that may increase model memorization: (1) the impact of data reuse; (2) the impact of easy-to-memorize characteristics of some data; (3) the combined impacts of the aforementioned two factors. We develop three defense primitives based on the characteristics of the two factors. Then, we combine these primitives in different ways to propose defense solutions for the above scenarios, aiming to mitigate memorization and effectively defend against MIAs.

The existing MIA defense mechanisms can be classified into two types. One aims to provide a provable privacy guarantee through differential privacy mechanisms [18]–[20]. While this method offers strict membership privacy guarantees, it often significantly degrades model accuracy. The other aims to provide empirical membership privacy, which can offer higher model accuracy to a certain extent [12], [15], [21]–[23]. However, existing solutions seldom consider how the use and characteristics of training samples contribute to defense. Although our methods also fall into the category of empirical defenses, they dynamically adjust the focus on training samples based on the memorization degree of different models to various data, better meeting the privacy needs of training samples across different models. Additionally, one of our methods achieves effective defense by merely adjusting the use of training samples without modifying the training algorithm. Experimental results show that our defense methods offer a superior privacy-utility tradeoff and defense efficiency compared to existing defenses. The main contributions of this paper are as follows:

- We find that data reuse and inherent memorability increase model memorization during iterative pruning, heightening membership privacy risks for training data. To address this, we propose the WEMEM defense framework, which contains three methods to weaken the memorization of the iteratively pruned models and thus effectively defend against MIAs.
- The first proposed method reduces memorization by limiting training data reuse without modifying the training algorithm. The second imposes stricter constraints on learning easy-to-memorize data. The third combines both approaches for a stronger reduction in memorization.
- We conduct extensive experiments on ten adaptive MIAs, demonstrating that our methods provide effective defense in iterative model pruning. Our defenses offer a better privacy-utility tradeoff and are more time-efficient than the five existing methods. Based on the results, we further enhance the practicality of our defenses by enabling automatic adjustment of settings.

Our code is available at https://github.com/CactiLab/WeMeM.

II. Background Knowledge and Threat Model

In this section, we introduce the workflow of iterative model pruning, the concept of memorization, and our threat model.

A. Workflow of Iterative Model Pruning

The workflow of iterative model pruning [1], [5] includes the following steps:

- *Step 1*: An original and unpruned model $f_{\boldsymbol{\theta}_0}$ is trained, where $\boldsymbol{\theta}_0$ represents the model weights.
- *Step 2*: To achieve a final pruning rate of v, $(1-(1-v)^{\frac{1}{r}})$ of the less important weights are removed using a binary mask $\boldsymbol{M}$ to obtain the sparse weight parameters after each iteration, denoted as $\boldsymbol{\theta_i} \odot \boldsymbol{M}$, where r is the number of iterations, and $\odot$ is the Hadamard product.
- *Step 3*: To recover the model accuracy, fine-tuning is performed on the pruned model $f_{\boldsymbol{\theta_i}\odot \boldsymbol{M}}$ with t epochs and obtain the pruned model $f_{\boldsymbol{\theta_{i+1}}}$.
- *Step 4*: Repeat Steps 2 and 3 for r iterations to obtain the final pruned model $f_{\boldsymbol{\theta_r}}$ with a pruning rate of v.

B. Memorization

In classification datasets, rare and atypical data samples in the "long tail" are difficult to learn accurately [16]. However, if the model memorizes these data and their labels, it can improve prediction accuracy for these classes, making memorization essential for achieving generalization. Feldman et al. [17] introduced the *memorization score* (mem-score) as a metric to quantify memorization. For the training set D and an arbitrary sample $(\boldsymbol{x}, y) \in D$, given the learning algorithm $\mathcal{A}$, the memorization score of this sample is defined as

$$\begin{aligned} &\mathrm{mem}(\mathcal{A}, D, (\boldsymbol{x}, y)) \\ &= \Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D)}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y] - \Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D\setminus(\boldsymbol{x},y))}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y] \end{aligned} \tag{1}$$

where $f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D)$ denotes the model f parameterized with weights $\boldsymbol{\theta}$ is learned by running the algorithm $\mathcal{A}$ on D, and $D\setminus(\boldsymbol{x}, y)$ denotes the sample $(\boldsymbol{x}, y)$ is removed from D.

C. Likelihood Ratio Attack (LiRA)

LiRA [24] is a recent MIA that uses a hypothesis test to determine whether the target model $f_{\boldsymbol{\theta}}$ was trained on a dataset containing a specific sample $(\boldsymbol{x}, y)$. Its attack performance is positively correlated with the model's memorization scores on individual samples [25], and it achieves high true-positive rates (TPR) even at low false-positive rates (FPR). For a sample $(\boldsymbol{x}, y)$, LiRA considers two Gaussian distributions on the target model, namely $\mathbb{Q}_{in}$ and $\mathbb{Q}_{out}$, which represents the confidence distribution on the logit scale when the model is trained or not on the sample, respectively:

$$\begin{aligned} \mathbb{Q}_{in} &= \{\phi(f_{in}(\boldsymbol{x})_y) \mid (\boldsymbol{x}, y) \in D\} \sim \mathcal{N}(\mu_{in}, \sigma_{in}^2) \\ \mathbb{Q}_{out} &= \{\phi(f_{out}(\boldsymbol{x})_y) \mid (\boldsymbol{x}, y) \notin D\} \sim \mathcal{N}(\mu_{out}, \sigma_{out}^2) \end{aligned} \tag{2}$$

where $\phi(m) = \log(\frac{m}{1-m})$ and $f_{in/out}(\boldsymbol{x})_y$ denotes model confidence. Given a sample $(\boldsymbol{x}, y)$, LiRA performs a likelihood-ratio test to determine which distribution the sample more likely belongs to. The likelihood ratio Λ is defined as

$$\Lambda = \frac{p(\phi(f_{\boldsymbol{\theta}}(\boldsymbol{x})_y) \mid \mathcal{N}(\mu_{in}, \sigma_{in}^2))}{p(\phi(f_{\boldsymbol{\theta}}(\boldsymbol{x})_y) \mid \mathcal{N}(\mu_{out}, \sigma_{out}^2))} \tag{3}$$

To estimate $\mathbb{Q}_{in/out}$, LiRA trains several shadow models, with half trained on the target sample $(\boldsymbol{x}, y)$ and the other half not. Then, LiRA fits two Gaussians to the confidences of these shadow models on $(\boldsymbol{x}, y)$. Finally, the confidence of the target model $f_{\boldsymbol{\theta}}$ is queried on the sample $(\boldsymbol{x}, y)$. If the likelihood ratio Λ for this sample exceeds a specified threshold, the sample is determined as a member of the training dataset.

D. *Threat Model and Design Goals*

In this paper, we consider MIAs using the shadow training technique [11]. We assume adversaries know the statistics of the target pruned model's training dataset and can synthesize or collect data with a similar distribution, dividing it into shadow training and test datasets. Adversaries are also assumed to know the training algorithm and basic architecture of the pruned model, with black-box query access to the model's output but without access to model weights or intermediate calculations. They cannot query the original, unpruned model.

For the attack, adversaries are assumed to know the pruning methods, rate v, and iterations r. They train multiple shadow models on shadow training data, prune them to mimic the target pruned model, and use the outputs of these shadow models to launch the attack. Furthermore, we assume adversaries are aware of the defense mechanisms and perform adaptive attacks by applying defenses during the generation of shadow-pruned models. In summary, our attack assumptions align with existing works [15], [22], [23].

Our design goals for defending against MIAs on iteratively pruned models are as follows: First, the defense mechanisms should maintain an acceptable privacy-utility tradeoff, ensuring that model performance is not significantly compromised. Second, the defenses should not introduce additional storage or computational costs when deploying and using pruned models. Lastly, the defenses should not increase the time required for the iterative pruning process.

III. Our Solution

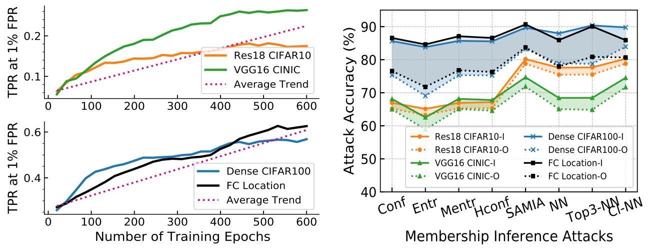
In this section, we first discuss our design rationale, followed by the design overview and approach details.

A. *Motivation and Design Rationale*

It has been shown that one-shot pruned models exhibit higher privacy risks compared to the original models [15], likely due to the fine-tuning process, which reuses training data and increases memorization of training samples. In light of this, we empirically investigate the relationship between data reuse and increased model memorization. Specifically, we use LiRA's results to show increased memorization, where a higher attack TPR reflects greater memorization.

Fig. 1a demonstrates that additional training epochs (i.e., increased data reuse) lead to a higher LiRA TPR, confirming that data reuse amplifies model memorization. Additionally, iterative pruning with multiple rounds of fine-tuning results in increased data reuse, significantly enhancing memorization and presenting greater privacy risks compared to one-shot pruned models. Fig. 1b illustrates that MIA accuracy is higher in iteratively pruned models than in one-shot pruned models.

Furthermore, models memorize training samples to varying degrees [17], with higher memorization score data being more prone to memorization. Multiple rounds of fine-tuning during iterative pruning further increase the use of these easily memorized samples, intensifying the model's memorization of these data.



(a) LiRA TPR (w.r.t number of training epochs)

(b) Privacy risk of iterative (I) / one-shot (O) pruned models ($v = 60\%$).

Fig. 1: (a) The attack accuracy of eight black-box MIAs on iteratively pruned models with five iterations is higher than on one-shot pruned models. (b) As the training epochs increase, the TPR of LiRA on the same data rises, indicating that more data reuses leads to increased memorization.

The core idea of our defense is to reduce model memorization by addressing two key factors: the reuse of training samples and the inherently easy-to-memorize characteristics of certain data. These factors typically work together during fine-tuning in iterative pruning, but to effectively weaken memorization, we address them separately in three scenarios: (1) impact of data reuse: using the entire training set in each epoch increases model memorization; (2) impact of easy-to-memorize data: the model retains stronger memory of easy-to-memorize samples; (3) combined impact of data reuse and easy-to-memorize data: reusing the entire dataset while retaining a deeper memory of easy-to-memorize data amplifies overall memorization and privacy risks.

B. *Overview of the* WeMem *Framework*

Fig. 2 illustrates the workflow of the WeMem framework, which consists of two stages. The first stage involves training an original model and generating memorization scores. The second stage, namely memorization-weakened pruning, iteratively prunes and fine-tunes the model while simultaneously weakening its memorization.

Our fine-tuning process employs three memorization-weakening primitives, which combine to form three defense methods. The first primitive, memorization-score-based data ranking, ranks data within each class based on memorization scores and is essential for weakening memorization across all scenarios. The second, sliding-window-based data sampling, limits data reuse by sampling data for each epoch. The final primitive, adaptive regularization, applies varying L2 regularization intensities based on the data's privacy risk, specifically targeting high memorization-score data.

The three defense methods are (1) Ranking-based Sliding Window (RSW), which combines data ranking with a sliding window to reduce data reuse, weakening memorization in the first scenario; (2) Risky Memory Regularization (RMR),

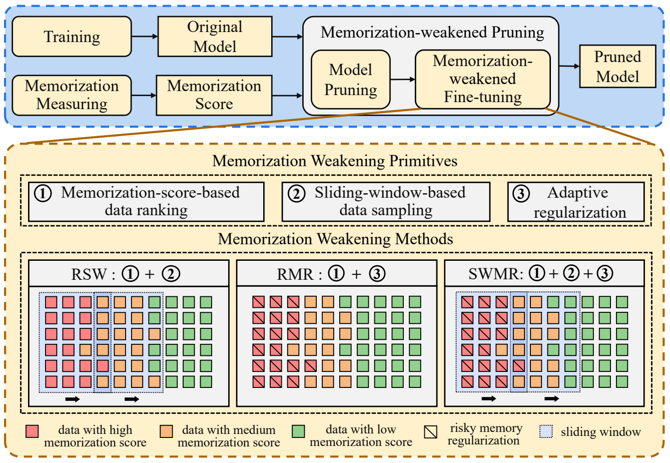

Training
Original Model
Memorization-weakened Pruning
Model Pruning
Memorization-weakened Fine-tuning
Pruned Model
Memorization Measuring
Memorization Score
Memorization Weakening Primitives
① Memorization-score-based data ranking
② Sliding-window-based data sampling
③ Adaptive regularization
Memorization Weakening Methods
RSW : ① + ②
RMR : ① + ③
SWMR: ①+②+③
data with high memorization score
data with medium memorization score
data with low memorization score
risky memory regularization
sliding window

Fig. 2: The WEMEM workflow and methods

which merges data ranking with adaptive regularization to limit training on high memorization-score data, addressing the second scenario; and (3) Sliding Window and Memory Regularization (SWMR), which integrates both RSW and RMR to mitigate memorization in the final scenario.

C. Efficient Memorization Score Calculation

A typical approach to calculate the memorization score for samples in the training dataset is to apply Eq. 1 on each data [17], which is computationally expensive and becomes impractical for large datasets. Van den Burg et al. [26] extended the memorization score calculation for supervised learning in [17] to the unsupervised density estimation problem and adapted it to be more computationally efficient. They applied Bayesian inference [27] to compute the memorization score of probabilistic generative models on data samples. We reintroduce the Bayesian inference approach to the classification problem while preserving computational efficiency.

Specifically, we first divide the dataset D into K disjoint subsets, ensuring each subset's distribution matches D. A subset is denoted as $D_i, i \in \{1, ..., K\}$. Then, we train the corresponding sub-model $f_{\boldsymbol{\theta}^i}$ on each subset D_i. For a sample $(\boldsymbol{x}, y) \in D_i$, we approximate $\Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D)}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y]$ with the prediction probability of $f_{\boldsymbol{\theta}^i}$ on $\boldsymbol{x}$ and its ground truth y, denoted as $\Pr_{f_{\boldsymbol{\theta}^i} \leftarrow \mathcal{A}(D_i)}[f_{\boldsymbol{\theta}^i}(\boldsymbol{x}) = y]$. Additionally, let $D' = D \backslash (\boldsymbol{x}, y)$ and use Bayesian inference, $\Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D \backslash (\boldsymbol{x}, y))}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y]$ can be written as

$$\begin{aligned}\Pr_{\mathcal{A}}[f(\boldsymbol{x}) = y | D'] &= \int p(f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y | D', \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \\ &\approx \frac{1}{K-1} \sum_{j=1, j \neq i}^{K} p(f_{\boldsymbol{\theta}^j}(\boldsymbol{x}) = y | D', \boldsymbol{\theta}^j)\end{aligned} \tag{4}$$

where $p(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$ with uncertainty, and $p(f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y | D', \boldsymbol{\theta})$ represents the conditional probability that model $f_{\boldsymbol{\theta}}$ outputs y for input $\boldsymbol{x}$. And, the probability $\Pr_{\mathcal{A}}[f(\boldsymbol{x}) = y | D']$ can be obtained by calculating the expectation of $p(f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y | D', \boldsymbol{\theta})$. We use the Monte Carlo sampling method[1] [27], [28], taking the average of the prediction probabilities of the remaining $K-1$ sub-models that have not used sample $(\boldsymbol{x}, y)$ as the approximate integration result. Our calculation method only needs to perform K times model training (far less than the total amount of data), which can greatly improve the computational efficiency.

D. Defense Primitives

1) Memorization-score-based Data Ranking: As high mem-score data samples pose greater privacy risks, they are the primary target of WEMEM's defenses. As shown in Fig. 3a, WEMEM ranks data samples within each class based on their mem-scores, a fundamental primitive used across all defense methods in the WEMEM framework.

2) Sliding-window-based Data Sampling: During fine-tuning, WEMEM uses a sliding window to control the amount of training data in each epoch. As shown in Fig. 3b, the window height (h) corresponds to the number of data classes. Let n be the total amount of training data. On average, the amount of data for each class is $l = n/h$. The window width (w) can be set to a value no larger than l. Each sampling by a window with size $h \times w$ provides data for one training epoch, and the window slides forward with a step size (s). The sliding window ensures that each sampling covers all classes of data equally in most cases.

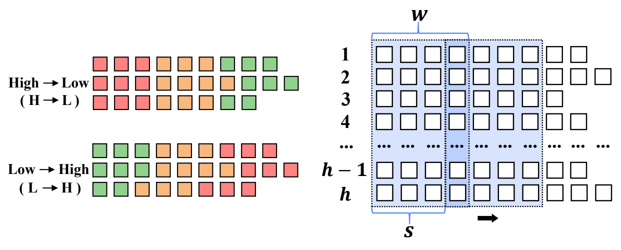
3) Adaptive Regularization: In model training, the L2 parameter regularization method is usually used to limit the model's learning capacity to prevent the model from overfitting the training data [28]. This method adjusts the limit intensity by setting regularization coefficients. Existing MIA defenses also adopt this method [11], [21], [29], but they only use one regularization coefficient to constrain the model's training on the entire dataset with a uniform intensity. WEMEM adopts the L2 parameter regularization method to impose regularization constraints with different intensities on the fine-tuning process of the model on high- and low-risk data. The approach is adaptive to the privacy risk of the training data, therefore achieving effective MIA defense in iterative pruning.

E. Defense Methods

1) Ranking-based Sliding Window (RSW): The RSW method adjusts how training data is utilized without modifying the training algorithm or altering the objective function. A sliding window provides only a subset of training data per epoch during fine-tuning, which reduces data reuse and weakens memorization. Additionally, by ranking data with mem-scores, the sliding window includes high mem-score samples in fewer epochs, thereby reducing the model's memorization of these samples.

2) Risky Memory Regularization (RMR): The core of the RMR method is to identify data samples with risky memory (i.e., high memorization score) by setting a mem-score threshold τ and using L2 parameter regularization to tightly

[1]The rationality for using the Monte Carlo approximation method is shown in Appendix A-F.



(a) (b)

Fig. 3: Diagram of (a) two data ranking orders and (b) sliding window attributes.

constrain the model's learning capacity on these samples. The loss function for this defense is defined as follows:

$$\mathcal{L}_{\text{RMR}} = \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) + \begin{cases} \frac{1}{2}\lambda_r \|\boldsymbol{\theta}\|_2^2, & \text{mem}(\boldsymbol{x}) \geq \tau \\ \frac{1}{2}\lambda_g \|\boldsymbol{\theta}\|_2^2, & \text{mem}(\boldsymbol{x}) < \tau \end{cases} \tag{5}$$

where for a sample $(\boldsymbol{x}, y)$, $\mathcal{L}_{\text{RMR}}$ contains a regular training loss $\mathcal{L}$ and an L2-norm regularization on $\boldsymbol{\theta}$ that is adaptive for the sample's mem-score. λ_r is a large coefficient applied to learn high mem-score data, and λ_g is a general small coefficient used for learning the remaining data. The L2 regularization based on λ_r weakens the model's memorization of high mem-score data, while the general coefficient λ_g for the remaining ensures that model utility is balanced, guaranteeing an effective defense without greatly affecting prediction accuracy.

3) Sliding Window and Memory Regularization (SWMR): This method combines the RSW and RMR methods to weaken memorization further. Specifically, during fine-tuning, each time the sliding window provides training data for an epoch, a threshold τ identifies high mem-score data within the window. L2 regularization is then applied to impose strict constraints on training with these high-risk data. Algorithm 1 details the SWMR method.

IV. Evaluation

In this section, we report the experiment results to evaluate the effectiveness of WeMem.

A. Evaluation Setup

1) Datasets and Models: We considered six datasets in the experiments:

- **CIFAR10** and **CIFAR100** [30]: Color image classification datasets, including 10 and 100 categories. Each contains 60,000 images. We used the default 50,000 training data and 10,000 test data for experiments.
- **CINIC** [31]: This dataset contains ten categories and has the same image size as CIFAR10. It has 270,000 images, and the training, test, and validation datasets contain 90,000 images, respectively. We used the default training and test samples for experiments.
- **Texas** [11]: This tabular dataset is used to predict the main type of surgery for patients based on features. It contains 100 categories, 67,330 data samples, and 6,169 binary features. We randomly selected 80% of them as training data and the remaining 20% as test data.
- **Location** [32], [33]: This tabular dataset is used to predict the social locations visited by users according to geographical history. It contains 30 categories, 5,010 data samples, and 446 binary features. We randomly selected 80% of them as training data and the remaining 20% as test data.
- **Purchase** [11]: This tabular dataset is used to predict which shoppers are repeated buyers according to purchase history. It contains 197,324 data samples, 600 features, and 100 classes. We randomly selected 80% of them as training data and the remaining 20% as test data.

Algorithm 1: Sliding window and memory regularization (SWMR) method

```
1  Input: Original model $f_{\boldsymbol{\theta_0}}$, pruning rate $v$, number of
          iteration $r$, training epochs $t$, mem-score
          calculation method mem(·) from Eq. 1,
          mem-score threshold $\tau$, regularization
          coefficients $\lambda_r$ and $\lambda_g$, learning rate $\eta$, a set of
          sub-datasets $D_s$ sampled by a sliding window
2  Output: Iteratively pruned model $f_{\boldsymbol{\theta_r}}$
3  for $i$ = 0 to $r - 1$ do                          ▷ Iterative pruning
4      $j \leftarrow 0$
5      $f_{\boldsymbol{\theta_i} \odot \boldsymbol{M}} \leftarrow$ Prune $f_{\boldsymbol{\theta_i}}$ with pruning rate $1 - (1 - v)^{\frac{1}{r}}$
6      while $j < t$ do                               ▷ Fine-tuning
7          for $\boldsymbol{x}$ in $D_s[j]$ do
8              if mem($\boldsymbol{x}$) $\geq \tau$ then
9                  $\mathcal{L} = \mathcal{L}(f_{\boldsymbol{\theta_i} \odot \boldsymbol{M}}(\boldsymbol{x}), y) + \frac{1}{2}\lambda_r \|\boldsymbol{\theta_i} \odot \boldsymbol{M}\|_2^2$
10             else
11                 $\mathcal{L} = \mathcal{L}(f_{\boldsymbol{\theta_i} \odot \boldsymbol{M}}(\boldsymbol{x}), y) + \frac{1}{2}\lambda_g \|\boldsymbol{\theta_i} \odot \boldsymbol{M}\|_2^2$
12             $\boldsymbol{\theta_i} \odot \boldsymbol{M} \leftarrow \boldsymbol{\theta_i} \odot \boldsymbol{M} - \eta \nabla_{\boldsymbol{\theta_i} \odot \boldsymbol{M}} \mathcal{L}$
13         $j \leftarrow j + 1$
14     $f_{\boldsymbol{\theta_{i+1}}} \leftarrow f_{\boldsymbol{\theta_i} \odot \boldsymbol{M}}$
15 return $f_{\boldsymbol{\theta_r}}$
```

In the experiments, we first calculated the mem-score for the training data (as described in Section IV-B). We then randomly split the training data into two parts: one for the target model and the other for the shadow models. From each partition, 10% of the data are added to the default test set, ensuring the test data for the target and shadow models are different. Based on this, we further split the incorporated data into an actual testing set (66%) and a validation set (34%).

For CIFAR10, CIFAR100, and CINIC, we used three DNNs: ResNet18 [34], DenseNet121 [35], and VGG16 [36]. For the Texas, Location, and Purchase datasets, we used a two-layer fully connected network (FC) with 256 and 128 neurons, respectively, with ReLU activation applied to all layers except the last one.

2) Existing MIAs and Defenses Used for Evaluation: We considered ten adaptive MIAs, with four being metric-based attacks (i.e., Conf [37], Entr [38], Mentr [23], Hconf [38]) and the remaining six being classifier-based attacks (i.e., SAMIA [15], NN [11], Top3-NN [38], Cl-NN [21], LiRA [24], TRAJECTORY [39]). A metric-based MIA is denoted as $\mathcal{M}(\cdot)$, and $\mathbb{1}\{\cdot\}$ is the indicator function:

- **MIA based on prediction confidence (Conf)**. Yeom et al. [37] pointed out that if a sample's prediction confidence on the ground-truth label exceeds a certain threshold, the adversary infers it as a member. This is defined as $\mathcal{M}_{Conf}(f_{\boldsymbol{\theta}}, (\boldsymbol{x}, y)) = \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x})_y \geq \tau_y\}$, where $f_{\boldsymbol{\theta}}(\boldsymbol{x})_y$ is prediction confidence for ground-truth label y, and τ_y is the confidence threshold produced by shadow training.
- **MIA based on prediction entropy (Entr)**. Salem et al. [38] proposed using prediction entropy for membership inference. If a sample's prediction entropy is below a threshold, the adversary infers it as a member. This is defined as $\mathcal{M}_{Entr}(f_{\boldsymbol{\theta}}, (\boldsymbol{x}, y)) = \mathbb{1}\{-\sum_i f_{\boldsymbol{\theta}}(\boldsymbol{x})_i log(f_{\boldsymbol{\theta}}(\boldsymbol{x})_i) \leq \hat{\tau}_y\}$, where $f_{\boldsymbol{\theta}}(\boldsymbol{x})_i$ is prediction confidence for any label, and $\hat{\tau}_y$ is the entropy threshold for label y.
- **MIA based on modified prediction entropy (Mentr)**. Song et al. [23] used modified prediction entropy performed membership inference. They integrated the information of ground-truth label into entropy computations, which is defined as $\mathcal{M}_{Mentr}(f_{\boldsymbol{\theta}}, (\boldsymbol{x}, y)) = \mathbb{1}\{-(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})_y) log(f_{\boldsymbol{\theta}}(\boldsymbol{x})_y) - \sum_{i \neq y} f_{\boldsymbol{\theta}}(\boldsymbol{x})_i log(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})_i) \leq \check{\tau}_y\}$, where $\check{\tau}_y$ is generated in the same way as Entr.
- **MIA based on highest prediction confidence (Hconf)**. Salem et al. [38] pointed out that whether a sample is a member can be found by comparing the highest prediction confidence with a threshold. It is defined as $\mathcal{M}_{Hconf}(f_{\boldsymbol{\theta}}, (\boldsymbol{x}, y)) = \mathbb{1}\{f_{\boldsymbol{\theta}}(\boldsymbol{x})_{max} \geq \tau'_y\}$, where $f_{\boldsymbol{\theta}}(\boldsymbol{x})_{max}$ is the highest prediction confidence of an input sample, and τ'_y is the threshold.
- **MIA based on prediction confidence (NN)**. Shokri et al. [11] invented a shadow training technique to train a binary classifier-based attack model to distinguish whether an input is member data.
- **MIA based on top-3 confidence (Top3-NN)**. Salem et al. [38] proposed that the top three prediction confidences can be used to train an attack classifier for membership inference.
- **MIA based on prediction confidence and ground truth label (Cl-NN)**. Nasr et al. [21] encoded the ground-truth labels as one-hot vectors and combined them with the prediction confidence as features to train an attack classifier.
- **MIA based on self-attention mechanism (SAMIA)**. Yuan et al. [15] used "prediction sensitivity" to measure changes in prediction behavior from slight input variations and employed a self-attention network to automatically extract attack thresholds and produce attack results.
- **MIA based on loss trajectory (TRAJECTORY)**. Liu et al. [39] employed knowledge distillation to represent membership information through the losses evaluated on a series of intermediate models at various distillation epochs, in addition to the loss from the target model.

We compared five existing defenses in our evaluation:

- **Early Stopping and Regularization (Base)**. Early stopping and L2 regularization are classical and effective methods to defend against MIAs [11], [21], [23]. Early stopping reduces training epochs to balance model utility and privacy, while L2 regularization penalizes over-training, trading slight accuracy loss for lower privacy risk.
- **Pair-based Posterior Balancing (PPB)**. Yuan et al. [15] aligned prediction behaviors on prediction confidence and sensitivity by making the distributions of ranked posterior predictions from two input examples as similar as possible. The similarity is adjusted by a hyper-parameter λ, effectively reducing the differences in prediction confidence and sensitivity between members and non-members.
- **Adversarial Regularization (ADV)**. Nasr et al. [21] modeled a min-max privacy game between the defense mechanism and MIAs. The defense first maximizes the surrogate attack classifier's membership inference gain and then minimizes both the target classifier's prediction loss and the surrogate attack classifier's membership inference gain. The α controls the model privacy-utility tradeoff.
- **Differential Privacy (DP)**. We applied differentially private SGD (DPSGD) as used in prior studies [18]–[20], [40]. DPSGD involves clipping the gradient and adding noise from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The Gaussian mechanism achieves (ϵ, δ)-DP with σ determined by $\Omega(q\sqrt{T\log(1/\delta)}/\epsilon)$, where q is the sampling ratio, T is the total number of iterations, and ϵ is the privacy parameter [41]–[44]. A larger δ provides stronger privacy but potentially reduces accuracy.
- **Training Based on Relaxed Loss (RelaxLoss)**. Chen et al. [22] mitigated MIAs by blurring the distinction between member and non-member loss distribution. In their defense, if the current loss exceeds the target mean, run normal gradient descent; otherwise, apply gradient ascent or flatten the target posterior scores for non-ground-truth classes.

We used the Base method in all experiments to enhance defense performance and expedite experiment progress. All defenses are applied during the iterative fine-tuning process.

3) Attack and Defense Settings: The detailed settings for model training, pruning, attacks, and defenses are as follows.
Training and Attack Settings. We used the Adam optimizer to train both the target and attack models for 100 epochs, with a learning rate of 0.001 and a batch size of 128. For model pruning, we used the same optimizer and batch size, with each fine-tuning lasting 21 epochs. The default L2 regularization coefficient is 0.0005. We applied early stopping, ending training, or fine-tuning if the validation loss didn't improve for five consecutive epochs.
Model Pruning Settings. We used the L1 unstructured pruning method as our primary pruning strategy because it is suitable for any model architecture. We also evaluated the effects of using L1 and L2 structured pruning strategies. In evaluation, we followed the pruning workflow described in Section II-A and performed five iterations for pruning. We used three

TABLE I: Sliding windows and mem-score threshold settings.

Data	Height (h)	Width (w)	Step Size (s)	Model	Threshold
CIFAR10	10	{1500, 1000, 500}	{50, 100}	All three DNNs	τ = 0.5
CIFAR100	100	{150, 100, 50}	{5, 10}	All three DNNs	τ = 0.6
CINIC	10	{2700, 1800, 900}	{100, 200}	ResNet18, VGG16 DenseNet121	τ = 0.7 τ = 0.65
Texas	100	{160, 110, 55}	{5, 10}	FC	τ = 0.6
Location	30	{40, 30, 15}	{1, 3}	FC	τ = 0.6
Purchase	100	{474, 316, 158}	{25, 35}	FC	τ = 0.75

pruning rates: 50%, 60%, and 70%, representing the proportion of weights removed.

Defense Settings. (1) Data Ranking Settings: To evaluate the impact of high mem-score data appearing in the front or back of the dataset on the defense effect, we ranked the data in descending and ascending order of mem-scores.

(2) General Settings for Sliding Windows: We empirically divided the data for each class into ten segments. So according to Section III-D2, the window width can be set between $l/10$ to l. The step size is determined by the number of epochs (t) during fine-tuning, with an upper limit of $\lfloor l/t \rfloor$[2]. It should be noted that a width that is too small results in too little training data, thereby significantly reducing the model utility, while a larger width maintains utility but weakens the sliding window's defense effect. Therefore, we set multiple hyperparameters within the above settings, depending on the size of the different datasets and the number of training epochs (for each dataset, we set three w between $\frac{2}{3}l$ and $\frac{2}{9}l$, two s close to $\frac{1}{2}\lfloor l/t \rfloor$ and $\lfloor l/t \rfloor$), to observe the evaluation results. Our hyperparameter settings are detailed in Table I.

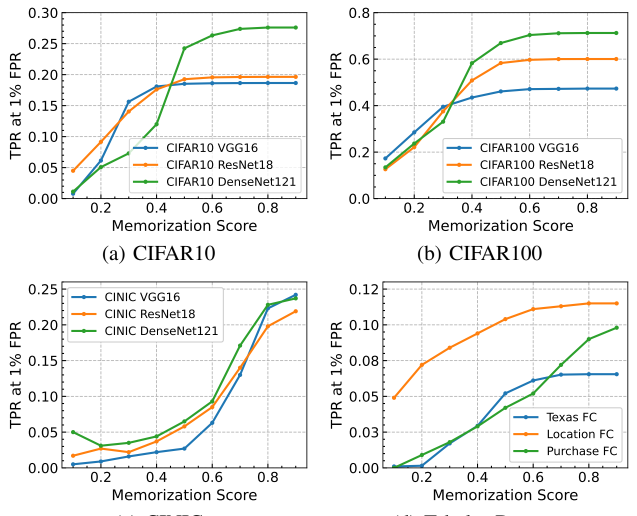
(3) RMR Settings: We used common hyperparameters from L2 regularization techniques. Specifically, we set $\lambda_g = 0.0005$ and $\lambda_r \in \{0.01, 0.1, 1\}$. Given that the SWMR also utilizes these two coefficients, we selected those that optimize the privacy-utility tradeoff based on RMR's results.

(4) Memorization Score Thresholds Settings: To make the mem-scores interpretable, we normalized them to a value between 0 and 1 according to the range of score values in [17]. We found that when the mem-score is 0.5 or higher, the data privacy risk increased significantly (as shown in Fig. 4), and the corresponding amount of training data decreased substantially. Therefore, if the threshold is too high, the amount of high-risk data identified will be insufficient, weakening the defense effectiveness. Conversely, if the threshold is too low, excessive data will be treated as high-risk, reducing model utility. We recommend that the threshold be no less than 0.5 and that a value be selected based on mem-scores and the corresponding data amount. The last two columns of Table I show our settings, where high mem-score data didn't exceed 30% of the total training data in each dataset.

B. *Effectiveness of Mem-score*

We used LiRA to evaluate the effectiveness of mem-scores. In theory, there is a positive correlation between the TPR of

[2]Generally, $l \geq t$, if $l < t$, the step size is set to 1.



(a) CIFAR10 (b) CIFAR100

(c) CINIC (d) Tabular Datasets

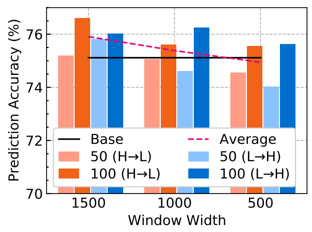
Fig. 4: Privacy risk of data with different memorization scores.

LiRA and the mem-score [25]. We obtained each training sample's mem-score using the method described in Section III-C. Notably, instead of the individual sample's precise mem-score value, our defenses focus more on the model's memorization trend of all training data and the privacy risk for data with varying memorization degrees. Therefore, as long as the mem-score we calculated correctly reflects the privacy risk of the data, our calculation method is effective.

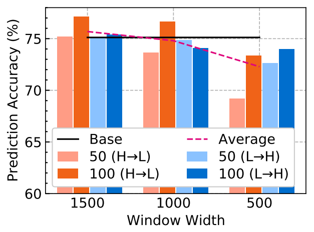
The available computing resources determine the upper limit of the K value described in Eq. 4 (i.e., the number of sub-models). Additionally, we found that when K is set to a larger value (with a maximum of 10 in our experiments), the training data available to each sub-model becomes limited. The limitation reduces each sub-model's prediction capability on the data samples. Since the mem-score relies on each sub-model's performance on the samples, a larger K makes it more difficult to ensure the mem-score's effectiveness. Thus, in our evaluation, we empirically divided CIFAR100 into four subsets and the rest of the datasets into five subsets. For LiRA, we trained a target model and corresponding 16 shadow models for experiments. Fig. 4 illustrates that the TPR of LiRA increases with the rise of samples' mem-scores, proving that the data with higher mem-scores face higher privacy risk and are easier to classify as a member correctly. Our method can effectively calculate the mem-scores.

C. *Effectiveness of* WEMEM

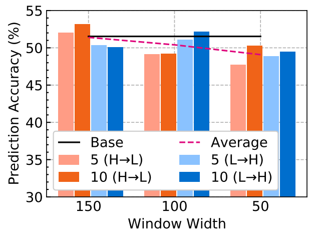
We evaluated the impact of our defenses on model prediction (i.e., test) accuracy and their effectiveness under different settings. We conducted adaptive MIAs where adversaries knew all defense and pruning methods and replicated the process to obtain shadow-pruned models. Thus, we trained five shadow models, pruned them, and used their predictions on shadow training and test data to train the attack models.



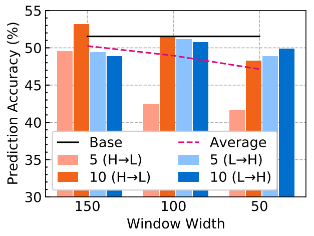
(a) CIFAR10 Res18 (RSW)



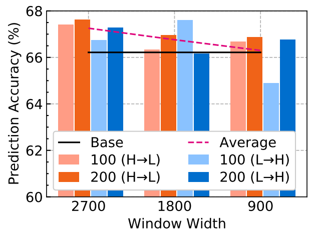
(b) CIFAR10 Res18 (SWMR)



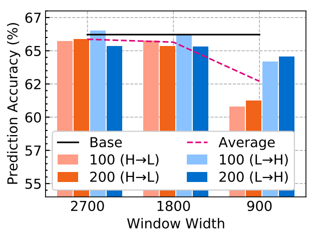
(c) CIFAR100 Dense121 (RSW)



(d) CIFAR100 Dense121 (SWMR)



(e) CINIC VGG16 (RSW)



(f) CINIC VGG16 (SWMR)

Fig. 5: The prediction accuracy of the pruned models using two data rankings (H→L/L→H) and two step sizes (e.g., $s \in \{50, 100\}$) under three window widths in RSW and SWMR. The black line presents the result of the Base method. The *Average* indicates the change in average prediction accuracy under different window widths.

1) Prediction Accuracy (acc): As shown in Fig. 5, for the RSW and SWMR methods using sliding windows, there is no significant decrease in model prediction accuracy compared to the Base method. Generally, as window width decreases (e.g., the dotted line in Fig. 5a), model prediction accuracy declines due to the limited training data available for fine-tuning.

Intuitively, a small step size allows the model to reuse more data from the previous epoch, enhancing memorization and improving prediction capability. However, experimental results show that, in many cases, a larger step size corresponds to higher prediction accuracy than a smaller one (e.g., dark bars are higher than light bars in Fig. 5a), which occurs more frequently when the window width is smaller. We speculate this is because, unlike reusing the entire training set, the specified window width in our defense limits the amount of training data available in each epoch. As a result, reusing a small amount of data each epoch is less effective than extracting features from more "new data"[3] for improving generalization performance.

[3]Refers to the training data that have never participated in the fine-tuning.

TABLE II: Under RMR defense with $\lambda_g = 0.0005$ and $\lambda_r \in \{0.01, 0.1, 1\}$, the test and attack accuracy on different pruned models. The best result of the privacy-utility tradeoff is bold in the column of regularization coefficient, test accuracy, and eight adaptive MIAs' attack accuracy.

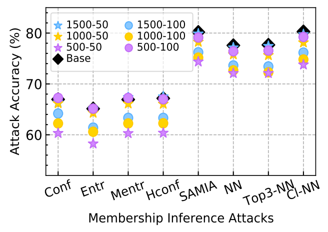
Data&Model	λ_r	Test Acc	Adaptive Attack Accuracy (%)							
		(%)	Conf	Entr	Mentr	Hconf	SAMIA	NN	Top3-NN	Cl-NN
CIFAR10 DenseNet121	Base	80.01	63.91	62.05	63.96	64.33	78.10	75.85	76.08	78.44
	0.01	78.96	60.69	58.43	60.67	60.78	76.19	73.50	73.41	76.22
	0.1	**77.81**	**54.60**	**53.06**	**54.78**	**54.84**	**73.07**	**72.89**	**73.17**	**73.13**
	1	69.83	52.14	50.97	51.99	51.93	72.79	73.27	72.04	73.03
CIFAR100 ResNet18	Base	42.44	91.91	91.02	92.10	92.09	94.39	93.98	94.84	94.36
	0.01	41.03	90.03	88.68	90.18	90.24	93.17	92.78	93.02	93.58
	0.1	**37.46**	**60.12**	**54.69**	**60.07**	**59.93**	**73.30**	**73.29**	**72.45**	**72.91**
	1	10.13	50.88	50.07	50.88	51.21	71.32	72.05	71.67	72.37
CINIC VGG16	Base	66.21	68.11	62.55	68.15	67.78	74.67	68.43	68.46	74.52
	0.01	**64.31**	**53.85**	**53.52**	**56.73**	**56.55**	**63.25**	**63.02**	**63.02**	**63.86**
	0.1	10.181	50.17	50.00	50.17	50.00	63.31	63.02	63.22	63.02
	1	9.63	50.08	50.18	50.08	51.21	62.32	63.05	63.22	63.02

TABLE III: The results of defense against LiRA and TRAJECTORY. The lowest attack TPR obtained by three defenses is in bold.

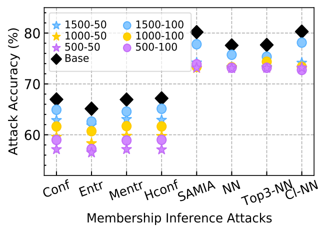
Data & Model	Defense	Prediction Acc (%)	TPR at 1% FPR (LiRA)	TPR at 0.5% FPR (TRAJECTORY)
CIFAR10 ResNet18	Base	90.45	14.0%	5.9%
	RSW (H→L)	87.84	7.1%	3.9%
	RSW (L→H)	**85.76**	**2.3%**	**1.8%**
	RMR	**79.25**	**2.0%**	**1.3%**
	SWMR (H→L)	**80.40**	**1.2%**	**1.3%**
	SWMR (L→H)	83.37	1.8%	1.7%
CIFAR100 DenseNet121	Base	63.85	51.4%	28.9%
	RSW (H→L)	55.96	22.6%	13.3%
	RSW (L→H)	**59.11**	**16.0%**	**8.3%**
	RMR	**50.12**	**2.6%**	**5.7%**
	SWMR (H→L)	**53.07**	**2.7%**	**8.2%**
	SWMR (L→H)	54.48	8.7%	9.1%
Location FC	Base	65.07	48.3%	17.7%
	RSW (H→L)	62.87	26.7%	12.4%
	RSW (L→H)	**64.27**	**20.0%**	**10.6%**
	RMR	**63.07**	**18.0%**	**9.2%**
	SWMR (H→L)	**62.37**	**6.9%**	**10.2%**
	SWMR (L→H)	60.27	19.1%	11.6%

Additionally, we observe that prediction accuracy under SWMR is often lower than under RSW with identical settings (e.g., Fig. 5e vs. Fig. 5f) due to the additional effect of memory regularization in the SWMR defense besides the sliding window. For the RMR defense, as shown in Table II, the prediction accuracy decreases as the value of λ_r increases in most cases.

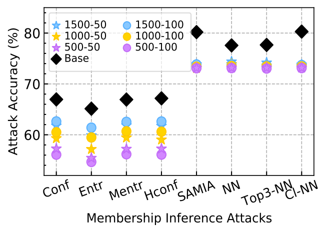
2) Defense Effectiveness: As shown in Table II, the RMR method assigns higher coefficients to high mem-score data, reducing the memorization of high-risk data and effectively defending against adaptive MIAs. The highlight rows that achieve the best privacy-utility tradeoff are mostly when $\lambda_r = 0.1$. In these cases, attack accuracy significantly decreases compared to the Base method, while prediction accuracy is less affected. Although the attack accuracy drops further for higher values like $\lambda_r = 1$, model utility becomes unacceptable in many cases.



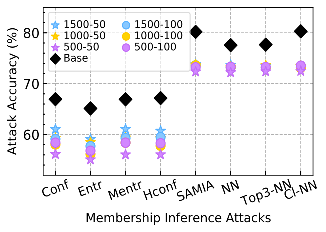
(a) RSW (H→L)



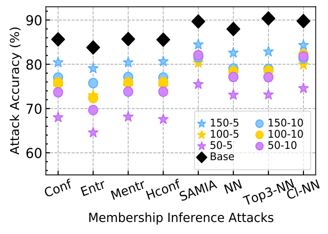
(b) RSW (L→H)



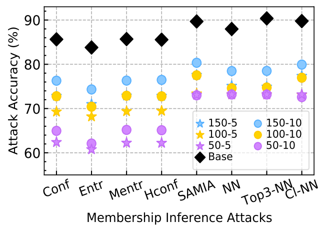
(c) SWMR (H→L)



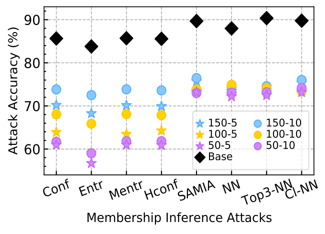
(d) SWMR (L→H)



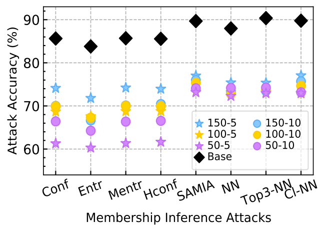
(e) RSW (H→L)



(f) RSW (L→H)



(g) SWMR (H→L)



(h) SWMR (L→H)

Fig. 6: The defense effectiveness of the pruned models using two data rankings (H→L/L→H) and two step sizes (e.g., $s \in \{50, 100\}$) under three window widths (e.g., $w \in \{1500, 1000, 500\}$) in RSW and SWMR. The results for CIFAR10-ResNet18 are presented in figures (a)-(d), and the results for CIFAR100-DenseNet121 are presented in figures (e)-(h).

Fig. 6 shows the effectiveness of RSW and SWMR defense methods. Generally, a sliding window with a small width and small step size significantly weakens memorization, achieving the best defense in our settings. Additionally, due to the combined effects of sliding windows and risky memory regularization, SWMR provides better defense compared to RSW under identical settings (e.g., Fig. 6a vs. Fig. 6c).

For RSW, ranking data from high to low by mem-score is generally less effective than ranking from low to high (e.g., Fig. 6e vs. Fig. 6f). When prioritizing high-risk data early in training, the model captures key features from these high-risk samples in the initial epochs to recover prediction accuracy quickly. These data are more beneficial to the model's performance than low-risk data appearing in later epochs, but they also pose higher privacy risks. While for SWMR, ranking data from high to low by mem-score (e.g., Fig. 6c and Fig. 6g) maximizes the effectiveness of risky memory regularization, further weakening memorization of high-risk data and improving defense.

Table III shows the results of the proposed defense methods on LiRA and TRAJECTORY with optimal defense parameters (e.g., minimum window width and step size, $\lambda_r = 0.1$), indicating that our defenses are also effective against the more advanced adaptive MIAs and can achieve defense without significantly reducing the utility of the pruned model.

D. Performance Comparison with Existing Defenses

We used identical adaptive attack settings for other defenses in the evaluation. We set the hyperparameters as follows: $\lambda \in \{1, 2, 4, 8, 16\}$ for PPB, $\delta \in \{1, 2, 4, 8\}$ for ADV, $\sigma \in \{0.01, 0.1, 1\}$ for DP noise vectors, and $\alpha = 1$ for RelaxLoss. For the proposed defenses, we select the results with the best defense effect for comparison, while for other defense

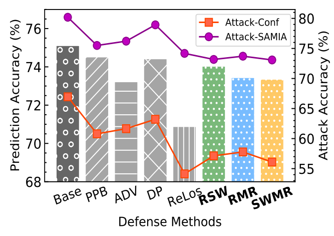
TABLE IV: LiRA's TPR at 1% FPR on high-risk and low-risk data before and after using our defense methods. The difference compared to the Base is shown in brackets.

CIFAR10 ResNet18					**CIFAR100 DenseNet121**				
$\tau = 0.5$	**Base**	**RSW**	**RMR**	**SWMR**	$\tau = 0.6$	**Base**	**RSW**	**RMR**	**SWMR**
Low-risk $[0, 0.5)$	6.4	2.9 (-3.5)	0.8 (-5.6)	2.0 (-4.4)	Low-risk $[0, 0.6)$	15.3	10.7 (-4.6)	2.5 (-12.8)	2.8 (-12.5)
High-risk $[0.5, 1]$	14.3	2.7 **(-11.6)**	1.2 **(-13.1)**	1.1 **(-13.2)**	High-risk $[0.6, 1]$	29.5	12.1 **(-17.4)**	2.8 **(-26.7)**	4.2 **(-25.3)**

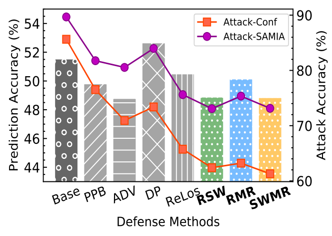
methods, we report experimental results that are comparable to our defenses in terms of prediction accuracy. Fig. 7 shows the prediction accuracy and adaptive attack accuracy of different defenses under two representative MIAs. In most cases, our defenses can achieve high prediction accuracy while reducing attack accuracy more than other defense methods. In addition to our defenses, RelaxLoss (abbreviated as ReLos in Fig. 7) also achieves a good privacy-utility tradeoff in many cases and is similar to our methods in terms of defense effects.

E. Data Privacy Risks Before and After Our Defenses

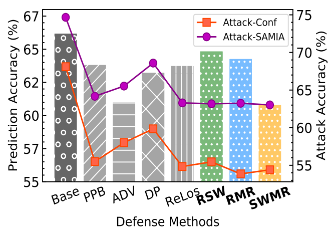
We compared the attack performance on training data with high and low mem-scores before and after using our defenses in iterative pruning. Specifically, we used LiRA, closely linked to memorization for evaluation, used the mem-score threshold τ to divide the high- and low-risk data, and calculated the average attack TPR under Base and proposed defenses. Table IV shows that high-risk data usually present higher attack TPR before using our defenses (i.e., results with the Base method). After our defenses, data privacy risks are effectively reduced, and our defense effects on high-risk data are more significant



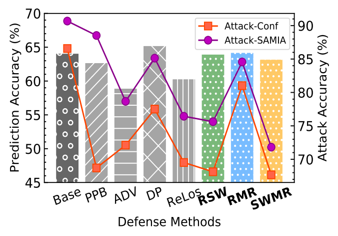
(a) CIFAR10 ResNet18



(b) CIFAR100 DenseNet121

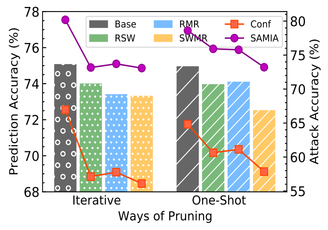


(c) CINIC VGG16

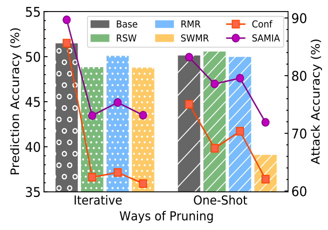


(d) Location FC

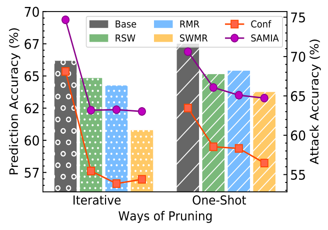
Fig. 7: Comparison of defense performance for iteratively pruned models against Conf and SAMIA under different defense methods. Bars represent prediction acc after defenses, and points on the line indicate attack acc under different defense methods.



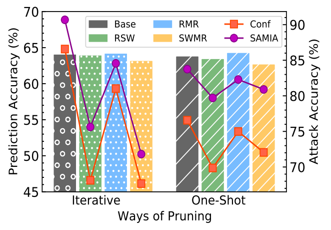
(a) CIFAR10 ResNet18



(b) CIFAR100 DenseNet121



(c) CINIC VGG16



(d) Location FC

Fig. 8: Comparison of results when using the proposed defenses designed for iterative pruning in one-shot pruning. Bars represent prediction acc after defenses, and points on the line indicate attack acc under different defense methods.

(the TPR is reduced more compared with the defense on low-risk data, as shown in bold). Additionally, the defense effects of RMR and SWMR methods are better than those of RSW.

F. Performance Comparison with One-Shot Pruning

We applied the proposed defenses in the one-shot pruning, and the best defense effects under the small window width are shown in Fig. 8. Although using our defenses in one-shot pruning can also reduce the attack accuracy, the reduction is less pronounced than the results in iterative pruning. This is mainly because the relatively small memorization increase during one-shot pruning limits the effectiveness of our defenses.

G. Defense Efficiency Comparison

We compared the time cost of our defenses and others in iterative pruning with five iterations. All experiments are conducted on a single NVIDIA GeForce RTX 4090 GPU. For our three defenses, we present the time cost for the best defense effects; for other defenses, we show the time costs closest to our best defense effects. As shown in Table V, RSW performs better in most cases. For example, the RSW defense time cost for CINIC-DenseNet121 (463s) is nearly 1/4 of the Base method (1616s). This is mainly because the sliding window sampling reduces the amount of training data in each epoch, speeding up the iterative fine-tuning process. In addition to RSW, ADV defense efficiency is also good, while DP has the least ideal time cost.

H. Other Pruning Strategies

To assess the robustness of our defenses across different pruning strategies, we applied them in iterative pruning using

TABLE V: The comparison of time cost in iterative pruning.

Data&Model	Base	RSW	RMR	SWMR	PPB	ADV	RelaxLoss	DP
CIFAR10 VGG16	630s	**269s**	468s	332s	571s	275s	434s	7h
CIFAR100 ResNet18	458s	**174s**	611s	259s	643s	226s	532s	9h
CINIC DenseNet121	1616s	**463s**	2404s	1498s	1759s	495s	1696s	50h
Location FC	93s	**68s**	98s	95s	99s	195s	88s	231s

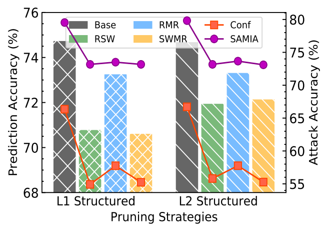
L1 and L2 structured pruning. The best results are shown in Fig. 9. The results show that our defenses effectively reduce attack accuracy, even with structured pruning. For example, on CIFAR10-ResNet18, they reduce the Conf attack accuracy by an average of 10.40% and 10.44% across both pruning strategies. Moreover, the defenses have a smaller impact on model utility with L2 structured pruning compared to L1. On CIFAR100-VGG16, the average prediction accuracy after L2 structured pruning decreased by 2.48% compared with Base, while the average prediction accuracy after L1 structured pruning decreased by 4.47%.

V. Ablation Study and Further Improvement

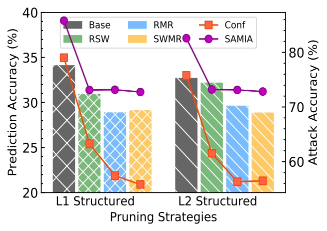
In this section, we conduct an ablation study and improve the proposed defense methods.

A. Ablation Study

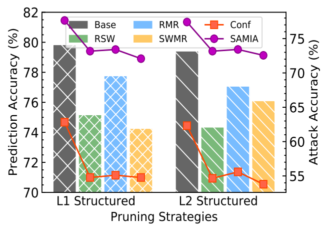
Our ablation study focuses on data ranking, a critical primitive for effective defense. We denote experiments using



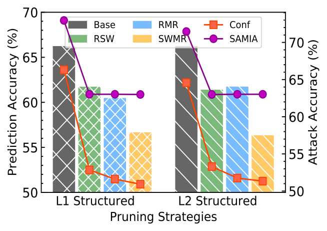
(a) CIFAR10 ResNet18



(b) CIFAR100 VGG16



(c) CIFAR10 DenseNet121



(d) CINIC DenseNet121

Fig. 9: Performance of our defenses when iterative pruning using L1 and L2 structured pruning strategies. Bars represent prediction accuracy after defenses, and points on the line indicate attack accuracy under different defense methods.

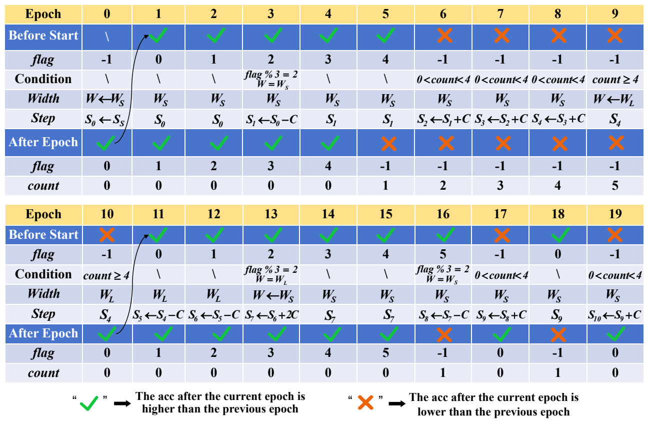
TABLE VI: The results of the ablation study for data ranking. The SW represents only using sliding window without data ranking, and the R1, R2 are two variants of the RMR method with two different settings of coefficients.

Data & Model	Defense	Test Acc (%)	Adaptive Attack Accuracy (%)			
			Conf	Mentr	SAMIA	Cl-NN
CIFAR100 VGG16	Base	35.97	81.52	81.42	87.95	87.71
	RSW	**34.04**	**64.73**	**64.69**	**73.22**	**72.29**
	SW	33.09	67.29	67.31	75.61	73.81
	RMR	**30.36**	**57.40**	**57.25**	**73.33**	**73.25**
	R1	0.98	50.78	50.78	73.17	73.17
	R2	0.98	50.67	50.68	72.69	72.35
Location FC	Base	64.11	86.57	87.09	90.68	85.91
	RSW	**63.98**	**68.13**	**68.35**	**75.62**	**71.54**
	SW	62.85	72.47	72.92	77.55	75.20
	RMR	**64.23**	**84.02**	**83.95**	**88.56**	**85.02**
	R1	18.89	53.04	53.11	69.42	68.36
	R2	16.89	52.63	52.68	69.42	67.99

only the sliding window as SW and the two variants of RMR as R1 (swapping the values of λ_r and λ_g) and R2 (making λ_r equal to λ_g). The results are shown in Table VI, where we highlight the experimental results with the worse performance of SW, R1, and R2 in red. Compared to the Base method, the SW without data ranking is less effective than RSW (L→H). For the SAMIA attack on CIFAR100-VGG16, RSW reduces attack accuracy by 14.73%, while SW reduces it by 12.34% at most, and SW's prediction accuracy is lower than RSW. For R1 and R2, the prediction accuracy is unacceptable in both cases; for example, in CIFAR100 results, the prediction accuracy is only 0.98%.

B. Adaptive Defense Improvements

In Section IV, we demonstrated the effectiveness of our defenses against MIA in iteratively pruned models. However,



Epoch	0	1	2	3	4	5	6	7	8	9
Before Start	\	✓	✓	✓	✓	✓	✗	✗	✗	✗
flag	-1	0	1	2	3	4	-1	-1	-1	-1
Condition	\	\	\	$flag \% 3 = 2$ $W = W_S$	\	\	$0 < count < 4$	$0 < count < 4$	$0 < count < 4$	$count \geq 4$
Width	$W \leftarrow W_S$	W_S	W_S	W_S	W_S	W_S	W_S	W_S	W_S	$W \leftarrow W_L$
Step	$S_0 \leftarrow S_S$	S_0	S_0	$S_1 \leftarrow S_0 - C$	S_1	S_1	$S_2 \leftarrow S_1 + C$	$S_3 \leftarrow S_2 + C$	$S_4 \leftarrow S_3 + C$	S_4
After Epoch	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗
flag	0	1	2	3	4	-1	-1	-1	-1	-1
count	0	0	0	0	0	1	2	3	4	5

Epoch	10	11	12	13	14	15	16	17	18	19
Before Start	✗	✓	✓	✓	✓	✓	✓	✗	✓	✗
flag	-1	0	1	2	3	4	5	-1	0	-1
Condition	$count \geq 4$	\	\	$flag \% 3 = 2$ $W = W_L$	\	\	$flag \% 3 = 2$ $W = W_S$	$0 < count < 4$	\	$0 < count < 4$
Width	W_L	W_L	W_L	$W \leftarrow W_S$	W_S	W_S	W_S	W_S	W_S	W_S
Step	S_4	$S_5 \leftarrow S_4 - C$	$S_6 \leftarrow S_5 - C$	$S_7 \leftarrow S_6 + 2C$	S_7	S_7	$S_8 \leftarrow S_7 - C$	$S_9 \leftarrow S_8 + C$	S_9	$S_{10} \leftarrow S_9 + C$
After Epoch	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓
flag	0	1	2	3	4	5	-1	0	-1	0
count	0	0	0	0	0	0	1	0	1	0

"✓" → The acc after the current epoch is higher than the previous epoch
"✗" → The acc after the current epoch is lower than the previous epoch

Fig. 10: Example of adaptive window adjustment.

RSW and SWMR require manual tuning of window width and step size, which complicates their use across different datasets and settings. To enhance their practicality, we aim to automate the adjustment of defense parameters based on model accuracy for adaptive defense during pruning.

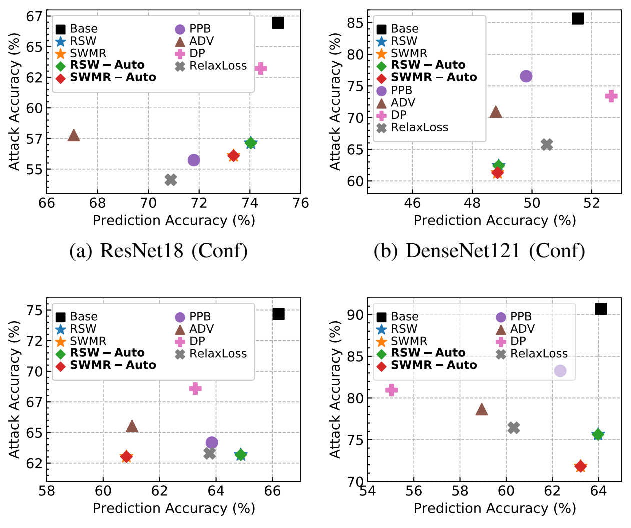
Results from Section IV-C show that smaller window widths improve defense, while step size helps balance accuracy. A smaller step usually limits model utility, while a larger one improves it. Based on these findings, we set initial window widths and step sizes to smaller values and prioritize step size adjustments to balance model accuracy during window sliding. Specifically, we set the window width and step size within the recommended range in Section IV-A3: $\frac{2}{10}l$ for a small width (W_S), $\frac{4}{10}l$ for a large width (W_L), $\frac{1}{2}\lfloor l/t \rfloor$ for a small step (S_S), and $\lfloor l/t \rfloor$ for the largest step (S_L). The constant C, used for gradually changing the step size, is defined as

$$C = \begin{cases} 10^{\lfloor \lg S_S \rfloor}, & \lfloor \lg S_S \rfloor < \lfloor \lg S_L \rfloor \\ 10^{\lfloor \lg S_S \rfloor - 1}, & \lfloor \lg S_S \rfloor = \lfloor \lg S_L \rfloor \end{cases} \tag{6}$$

where $\lfloor \lg S_S \rfloor$ and $\lfloor \lg S_L \rfloor$ denote the orders of magnitude of S_S and S_L.

Fig. 10 illustrates the automatic adjustment process. Assuming a fine-tuning contains 20 epochs, we initialized the window width and step size to W_S and S_S, respectively (epoch 0). We tracked increases in prediction accuracy with $flag$ and decreases with $count$, adjusting the window settings for the next epoch based on these conditions:
(1) If accuracy rises three times in a row with a small window, reduce step size (epoch 3, 16);
(2) If accuracy decreases and $0 < count < 4$, increase step size of the next epoch (epoch 6-8, 17, 19);
(3) If accuracy decreases four or more times in a row, switch to a large window (epoch 9, 10);
(4) If accuracy rises with a large window, reduce step size before the next epoch (epoch 11, 12);
(5) If accuracy rises three times in a row with a large window, switch to a small window and increase step size (epoch 13).

We used the same training settings from Section IV-A and sliding window sampling based on the above conditions to



(a) ResNet18 (Conf) (b) DenseNet121 (Conf)

(c) VGG16 (SAMIA) (d) FC (SAMIA)

Fig. 11: The results of the pruned models under RSW and SWMR with optimal fixed window settings, improved adaptive RSW and SWMR (i.e., RSW-Auto and SWMR-Auto), and other defenses with the best performance. We use CIFAR10-ResNet18, CIFAR100-DenseNet121, CINIC-VGG16, and Location-FC.

evaluate the proposed adaptive methods. Notably, due to the early-stopping setting with a threshold of five (described in Section IV-A3), the critical value for the number of accuracy decreases (i.e., *count*) should be under five. Thus, we set the critical value of *count* to four. When the critical value of $flag$ is three, the balance between the defense effect and model utility is better. Experimental results in Fig. 11 show that the improved defense methods automatically adjusting the window attributes are effective, achieving the same defense effect as the optimal fixed window settings (e.g., RSW-Auto and SWMR-Auto in Fig. 11a almost overlap with RSW and SWMR).

VI. Discussion

A. Computational Complexity

The additional computational cost of WeMem primarily arises from ranking the data by memorization scores, which is conducted once during iterative fine-tuning, resulting in a complexity of $O(n \log n)$. Compared to the total computational cost for a typical training process, which scales as $tn \sum_{i=1}^{d} m_i$ (t: number of training epochs; n: the size of training dataset; m_i: computational cost of the i-th layer, depends on the type of layer, e.g., convolutional layer, fully connected layer; d: number of layers), the additional cost of data ranking is negligible, especially in training large-scale models.

B. Necessity of Pruning

Applying the proposed methods to the unpruned model is inconsistent with the defense goal of WeMem. We conducted defense evaluation and relevant experimental analysis on the pruned model rather than the unpruned model, as WeMem aims to weaken the memorization that significantly increases during iterative model pruning. Therefore, we establish baselines on the pruned model for subsequent evaluations and analyses.

C. Robustness beyond MIAs

In addition to MIAs, model inversion attacks have also emerged as a significant privacy risk, capable of reconstructing training data from a public model and posing a direct threat to privacy. To assess whether our proposed methods are vulnerable to this type of attack, we evaluated it using recent white-box inversion attacks by Chen et al. [45] (KED-MI) and Yuan et al. [46] (PLG-MI). Chen's KED-MI leveraged a generative adversarial network (GAN) to extract knowledge from public data to attack private models, while Yuan's PLG-MI improved attack success rates via conditional GANs. We used 25,000 training samples (labels 0-4), trained a ResNet18 target model (and iteratively pruned it), evaluated it on VGG16, and finally reconstructed 100 images per class. Note that we assessed the performance of the proposed RSW and SWMR under the optimal settings. While KED-MI and PLG-MI achieve 49.22% and 73.87% attack accuracy on the undefended pruned models, the RSW defense drops it to 44.75% and 69.31%, and SWMR further reduces the accuracy to 29.77% and 60.21%. These results demonstrate that WeMem does not make the pruned model vulnerable to more severe privacy threats and even mitigates model inversion attacks.

D. Limitation and Future Works

First off, WeMem performed less stably on tabular datasets than on image datasets, likely due to the less pronounced memorization properties in simpler models. As shown in Fig. 4, the gap in LiRA's TPR between low and high mem-score data is smaller for tabular datasets, suggesting a weaker link between memorization and privacy risk. Nonetheless, WeMem is still effective for tabular datasets.

Second, while WeMem's defenses empirically achieved a good privacy-utility tradeoff, they lack a strict theoretical guarantee. This is a common limitation of empirical defense methods (e.g., [15], [21], [22]), and we plan to address this in future work.

Finally, in RSW and SWMR defenses, we filtered and used high-risk data within a window via data ranking and mem-score thresholds, which are coarse-grained approaches. More fine-grained methods can be considered, such as sampling data by different probabilities based on mem-scores. However, since this involves balancing multiple factors like sampling probability and the final data distribution's impact on the privacy-utility tradeoff, we leave this as future work to explore fine-grained memorization weakening techniques.

VII. Related Work

We provide a brief introduction to model pruning, review existing MIAs and defenses, and present related work on memorization and privacy.

A. Model Pruning

Model pruning is a technique for model compression, performed either in one-shot or iterative ways. One-shot pruning involves training, pruning, and fine-tuning the model. However, learning the correct connections is an iterative process [1]. In iterative pruning, the pruning and fine-tuning are repeated multiple times, yielding better performance compared to one-shot pruning [5]. Regarding pruning strategies, Han et al. [1] proposed unstructured pruning, which removes unimportant connections with weights below a threshold. He et al. [47], Luo et al. [48], and Liu et al. [49] introduced structured pruning methods that remove filters or channels with less contribution. Recently, neural architecture search [50]–[52] has further advanced model pruning techniques.

B. MIAs and Defenses

MIA aims to infer whether the data is used to train the target model. Shokri et al. [11] proposed the first classifier-based MIA. The adversary creates multiple shadow models to simulate the target model, generate data to train the attack model, and indicate whether the input data is a member. Salem et al. [38] used only one shadow model to achieve the same attack effect. Liu et al. [39] used the training losses of different training epochs to form a loss trajectory, which could represent membership information. Carlini et al. [24] recently proposed an advanced MIA that shows more confident results at low false-positive rates. Yuan et al. [15] recently studied the privacy risks of model pruning and proposed a self-attention MIA by exploring the impact of one-shot model pruning on prediction divergence. The metric-based methods calculate the metrics of the prediction vector, such as prediction loss [37], prediction confidence [38], and prediction entropy [23], and then the metrics are compared with the preset thresholds to determine the membership state of the data point.

There have been some defense mechanisms against MIAs. Differential privacy mechanism provides provable privacy guarantee [18]–[20], [41], [44] but can hardly achieve better privacy-utility tradeoff. A series of regularization defenses reduce the generalization gap between the member and non-member samples, including L1 and L2 regularization [11], [21], dropout [13], [53], adversarial regularization [21], [54] and early stopping [23], etc. Chen et al. [22] reduced the distinguishability between the training and testing loss distributions. Recently, Yuan et al. [15] proposed a defense to reduce the divergence of prediction confidence and sensitivity between members and non-members in model pruning.

C. Memorization and Privacy

In general, the machine learning (ML) models are always overconfident in the training data [55], and it is the main reason for the success of MIAs [11], [37], [53], [56]. ML models can learn helpful features in big data, but it also causes them to memorize some sensitive information [14], [16], [57]. In ML, the concept of memorization is an alternative perspective of the training data influence [58]. Pruthi et al. [59] proposed *self-influence* of training samples and showed that high *self-influence* is more likely to have an indicative effect on memorization. This view coincides with the definition of memorization by Feldman et al. [16], [17], that is, before and after removing a data sample, the greater the difference in prediction of the sample, the easier it is to memorize. In this paper, we use this definition to measure the memorization for classification models.

VIII. Conclusion

In this paper, we proposed a new defense framework WeMem to defend MIAs in iterative pruning by weakening memorization. We found that data reuse and the easy-to-memorize characteristics of some data are important factors that increase memorization during iterative pruning, leading to greater privacy risks. We considered these factors' separate and combined impacts, forming three scenarios that make pruned models more vulnerable to MIAs. To address these scenarios, we designed three defense primitives. Combining these primitives, we proposed methods tailored to each scenario that effectively weaken memorization.

Comprehensive evaluations of ten adaptive MIAs show that the three proposed methods successfully weaken memorization and provide effective defenses. The proposed methods outperform five existing defenses in privacy-utility tradeoff and defense time cost and effectively mitigate the privacy risks of high memorization-score data. Additionally, we improved the proposed methods from a practical perspective to make them automatically adjust hyperparameter configurations. Furthermore, the experimental results indicate that reusing easy-to-memorize data is a key factor contributing to the significant increase in memorization during iterative pruning.

Acknowledgment

The research of Beijing Jiaotong University and Beijing University of Technology was supported by the Fundamental Research Funds for the Central Universities (2022JBZY020) and the National Key R&D Program of China under Grant 2023YFB2703700.

References

[1] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems (NIPS)*, 2015.
[2] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, 2020.
[3] M. Gupta and P. Agrawal, "Compression of deep learning models for text: A survey," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2022.
[4] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations," *arXiv preprint arXiv:2308.06767*, 2023.
[5] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *International Conference on Learning Representations (ICLR)*, 2017.
[6] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, "Only train once: A one-shot neural network training and pruning framework," *Advances in Neural Information Processing Systems (NIPS)*, 2021.
[7] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, 2020.

[8] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *International Conference on Learning Representations (ICLR)*, 2016.

[9] S. Srinivas and R. V. Babu, “Data-free parameter pruning for deep neural networks,” *arXiv preprint arXiv:1507.06149*, 2015.

[10] A. Ardakani, C. Condo, and W. J. Gross, “Sparsely-connected neural networks: Towards efficient vlsi implementation of deep neural networks,” in *International Conference on Learning Representations (ICLR)*, 2017.

[11] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *IEEE Symposium on Security and Privacy (SP)*, 2017.

[12] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot, “Label-only membership inference attacks,” in *International Conference on Machine Learning (ICML)*, 2021.

[13] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro, “Logan: Membership inference attacks against generative models,” in *Privacy Enhancing Technologies (PETs)*, 2019.

[14] H. Hu, Z. Salcic, L. Sun, G. Dobbie, P. S. Yu, and X. Zhang, “Membership inference attacks on machine learning: A survey,” *ACM Computing Surveys*, 2022.

[15] X. Yuan and L. Zhang, “Membership inference attacks and defenses in neural network pruning,” in *USENIX Security Symposium*, 2022.

[16] V. Feldman, “Does learning require memorization? a short tale about a long tail,” in *Annual ACM SIGACT Symposium on Theory of Computing*, 2020.

[17] V. Feldman and C. Zhang, “What neural networks memorize and why: Discovering the long tail via influence estimation,” *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[18] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[19] B. Jayaraman and D. Evans, “Evaluating differentially private machine learning in practice,” in *USENIX Security Symposium*, 2019.

[20] S. Rahimian, T. Orekondy, and M. Fritz, “Differential privacy defenses and sampling attacks for membership inference,” in *ACM Workshop on Artificial Intelligence and Security*, 2021.

[21] M. Nasr, R. Shokri, and A. Houmansadr, “Machine learning with membership privacy using adversarial regularization,” in *ACM SIGSAC conference on computer and communications security (CCS)*, 2018.

[22] D. Chen, N. Yu, and M. Fritz, “Relaxloss: Defending membership inference attacks without losing utility,” in *International Conference on Learning Representations (ICLR)*, 2022.

[23] L. Song and P. Mittal, “Systematic evaluation of privacy risks of machine learning models,” in *USENIX Security Symposium*, 2021.

[24] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer, “Membership inference attacks from first principles,” in *IEEE Symposium on Security and Privacy (SP)*, 2022.

[25] X. Li, Q. Li, Z. Hu, and X. Hu, “On the privacy effect of data enhancement via the lens of memorization,” *IEEE Transactions on Information Forensics and Security (TIFS)*, 2024.

[26] G. van den Burg and C. Williams, “On memorization in probabilistic deep generative models,” *Advances in Neural Information Processing Systems (NIPS)*, 2021.

[27] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006.

[28] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[29] Y. Long, L. Wang, D. Bu, V. Bindschaedler, X. Wang, H. Tang, C. A. Gunter, and K. Chen, “A pragmatic approach to membership inferences on machine learning models,” in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[30] A. Krizhevsky, “Learning multiple layers of features from tiny images,” *Master’s thesis, University of Toronto*, 2009.

[31] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, “Cinic-10 is not imagenet or cifar-10,” *arXiv preprint arXiv:1810.03505*, 2018.

[32] D. Yang, D. Zhang, and B. Qu, “Participatory cultural mapping based on collective behavior data in location-based social networks,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2016.

[33] D. Yang, D. Zhang, L. Chen, and B. Qu, “Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns,” *Journal of Network and Computer Applications*, 2015.

[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations (ICLR)*, 2015.

[37] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, “Privacy risk in machine learning: Analyzing the connection to overfitting,” in *IEEE Computer Security Foundations Symposium (CSF)*, 2018.

[38] A. Salem, Y. Zhang, M. Humbert, M. Fritz, and M. Backes, “Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models,” in *Network and Distributed Systems Security Symposium (NDSS)*, 2019.

[39] Y. Liu, Z. Zhao, M. Backes, and Y. Zhang, “Membership inference attacks by exploiting loss trajectory,” in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[40] G. Zhang, B. Liu, T. Zhu, M. Ding, and W. Zhou, “Label-only membership inference attacks and defenses in semantic segmentation models,” *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 2023.

[41] C. Dwork, “Differential privacy,” in *International colloquium on automata, languages, and programming*, 2006.

[42] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu, “Effects of differential privacy and data skewness on membership inference vulnerability,” in *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2019.

[43] J. Chen, W. H. Wang, and X. Shi, “Differential privacy protection against membership inference attack on machine learning for genomic data,” in *Pacific Symposium on Biocomputing*, 2020.

[44] D. Zhong, H. Sun, J. Xu, N. Gong, and W. H. Wang, “Understanding disparate effects of membership inference attacks and their countermeasures,” in *ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2022.

[45] S. Chen, M. Kahla, R. Jia, and G.-J. Qi, “Knowledge-enriched distributional model inversion attacks,” in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2021.

[46] X. Yuan, K. Chen, J. Zhang, W. Zhang, N. Yu, and Y. Zhang, “Pseudo label-guided model inversion attack via conditional generative adversarial network,” in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[47] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[49] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[50] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, “Once-for-all: Train one network and specialize it for efficient deployment,” in *International Conference on Learning Representations (ICLR)*, 2020.

[51] B. Li, B. Wu, J. Su, and G. Wang, “Eagleeye: Fast sub-net evaluation for efficient neural network pruning,” in *European Conference on Computer Vision (ECCV)*, 2020.

[52] X. Lu, W. Dong, X. Li, J. Wu, L. Li, and G. Shi, “Adaptive search-and-training for robust and efficient network pruning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023.

[53] K. Leino and M. Fredrikson, “Stolen memories: Leveraging model memorization for calibrated white-box membership inference,” in *USENIX Security Symposium*, 2020.

[54] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, “Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer,” *arXiv preprint arXiv:1912.11279*, 2019.

[55] D. Hintersdorf, L. Struppek, and K. Kersting, “To trust or not to trust prediction scores for membership inference attacks,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[56] N. Carlini, M. Jagielski, C. Zhang, N. Papernot, A. Terzis, and F. Tramer, “The privacy onion effect: Memorization is relative,” *Advances in Neural Information Processing Systems (NIPS)*, 2022.

[57] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *USENIX Security Symposium*, 2019.

[58] Z. Hammoudeh and D. Lowd, "Training data influence analysis and estimation: A survey," *arXiv preprint arXiv:2212.04612*, 2022.

[59] G. Pruthi, F. Liu, S. Kale, and M. Sundararajan, "Estimating training data influence by tracing gradient descent," *Advances in Neural Information Processing Systems (NIPS)*, 2020.

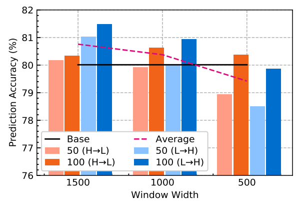
Appendix A
Supplementary Content

In this section, we present contents and experimental results at a 60% pruning rate that are not included in the main text.

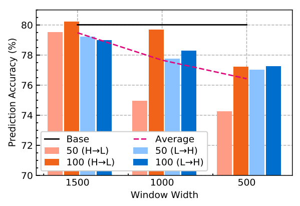
A. Prediction Accuracy

TABLE VII: Under RMR defense with $\lambda_g = 0.0005$ and $\lambda_r \in \{0.01, 0.1, 1\}$, the test and attack accuracy on different pruned models. The best result of the privacy-utility tradeoff is bold in the column of regularization coefficient, test accuracy, and eight adaptive MIAs attack accuracy.

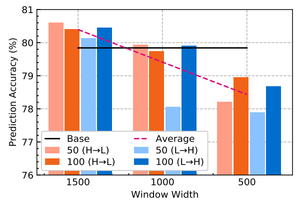
Data&Model	λ_r	**Test Acc (%)**	**Adaptive Attack Accuracy (%)**							
			Conf	**Entr**	**Mentr**	**Hconf**	**SAMIA**	**NN**	**Top3-NN**	**Cl-NN**
CIFAR10 ResNet18	Base	75.11	66.95	65.13	66.93	67.17	80.19	77.58	77.67	80.30
	0.01	74.35	64.06	61.08	64.10	64.06	77.62	74.12	74.04	77.54
	0.1	**73.44**	**57.77**	**54.67**	**57.87**	**57.92**	**73.71**	**73.17**	**73.17**	**73.29**
	1	64.83	51.99	50.71	51.75	51.92	73.17	73.17	73.17	73.17
CIFAR10 VGG16	Base	79.84	62.85	60.62	62.92	63.08	77.93	75.30	75.41	77.94
	0.01	79.79	61.25	59.13	61.35	61.60	77.21	74.41	74.46	77.27
	0.1	**79.21**	**58.71**	**55.78**	**58.57**	**58.50**	**75.24**	**73.19**	**73.17**	**74.03**
	1	75.97	53.55	51.95	53.70	53.57	73.09	73.17	73.22	73.07
CIFAR100 VGG16	Base	35.97	81.52	72.55	81.42	81.34	87.95	79.58	79.85	87.71
	0.01	35.23	77.61	67.49	77.61	77.47	84.69	75.72	75.77	84.15
	0.1	**30.36**	**57.40**	**53.34**	**57.25**	**57.47**	**73.33**	**73.17**	**73.13**	**73.25**
	1	8.88	50.61	50.16	50.69	50.70	72.86	73.14	72.17	72.59
CIFAR100 DenseNet121	Base	51.53	85.64	83.80	85.69	85.56	89.68	87.97	90.36	89.76
	0.01	49.10	74.38	69.06	74.31	74.57	82.97	77.09	77.30	82.84
	0.1	**50.14**	**63.21**	**57.15**	**63.35**	**62.98**	**75.35**	**73.17**	**74.13**	**73.73**
	1	41.40	53.21	51.07	53.22	53.35	72.97	73.17	73.17	72.79
CINIC ResNet18	Base	60.99	71.87	66.86	71.98	72.06	77.46	71.51	71.56	77.40
	0.01	61.76	65.22	60.34	65.24	64.87	70.74	64.71	64.55	70.45
	0.1	**57.92**	**52.59**	**51.28**	**52.55**	**52.41**	**63.02**	**63.02**	**63.22**	**63.14**
	1	35.89	50.48	50.17	50.18	50.20	62.48	62.88	63.02	62.96
CINIC DenseNet121	Base	67.77	66.04	61.24	66.09	66.12	72.76	67.09	67.09	72.76
	0.01	67.54	58.38	55.25	58.47	58.39	65.17	63.02	63.02	64.96
	0.1	**62.88**	**51.68**	**50.62**	**51.53**	**51.72**	**63.12**	**63.02**	**63.41**	**63.06**
	1	35.17	50.21	50.02	50.20	50.25	62.36	62.05	63.02	62.11
Location FC	Base	64.11	86.57	84.62	87.09	86.58	90.68	85.94	90.06	85.91
	0.01	64.61	86.60	85.38	86.98	86.21	90.10	85.90	89.68	85.82
	0.1	**64.23**	**84.02**	**81.96**	**83.95**	**84.23**	**88.56**	**84.67**	**87.52**	**85.02**
	1	62.25	83.15	81.08	83.44	83.42	87.37	84.33	86.98	84.17
Purchase FC	Base	90.17	57.22	56.47	57.25	57.43	69.49	70.44	70.54	71.73
	0.01	88.13	53.35	52.58	53.31	53.36	69.53	69.64	69.58	69.54
	0.1	**82.96**	**51.52**	**50.96**	**51.57**	**51.51**	**69.44**	**69.29**	**69.44**	**69.44**
	1	79.69	51.22	50.85	51.43	51.48	69.12	69.44	69.34	69.20
Texas FC	Base	61.19	74.77	67.75	74.68	72.85	82.58	77.55	76.12	82.85
	0.01	60.77	66.04	58.94	66.25	64.08	74.49	69.77	69.44	74.56
	0.1	**62.21**	**56.79**	**53.31**	**56.56**	**55.10**	**69.42**	**69.44**	**68.94**	**68.71**
	1	61.45	55.47	53.24	55.57	54.09	69.37	68.24	69.64	70.04



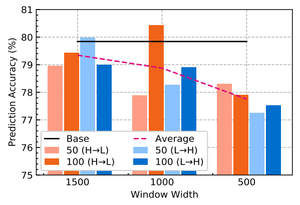
(a) CIFAR10 Dense121 (RSW)



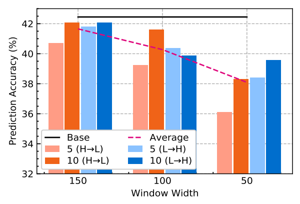
(b) CIFAR10 Dense121 (SWMR)



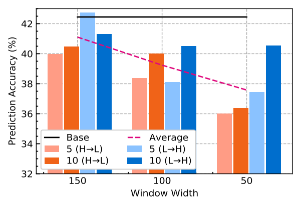
(c) CIFAR10 VGG16 (RSW)



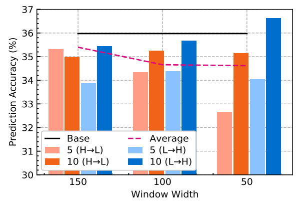
(d) CIFAR10 VGG16 (SWMR)



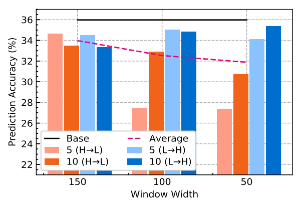
(e) CIFAR100 Res18 (RSW)



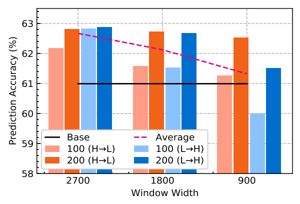
(f) CIFAR100 Res18 (SWMR)



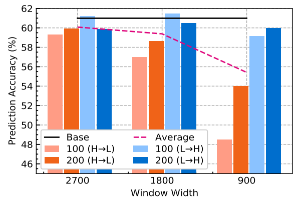
(g) CIFAR100 VGG16 (RSW)



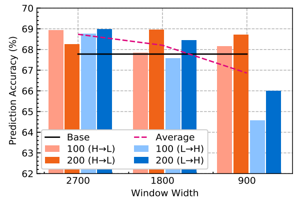
(h) CIFAR100 VGG16 (SWMR)



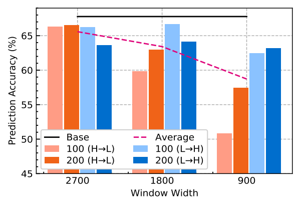
(i) CINIC Res18 (RSW)



(j) CINIC Res18 (SWMR)



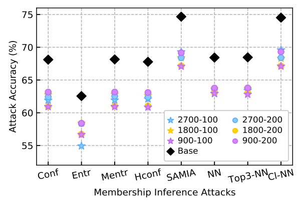
(k) CINIC Dense121 (RSW)



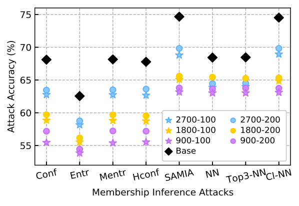
(l) CINIC Dense121 (SWMR)

Fig. 12: The prediction accuracy of the pruned models using two data rankings and two step sizes under three window widths in RSW and SWMR.

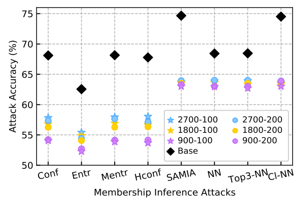
B. Defense Effectiveness



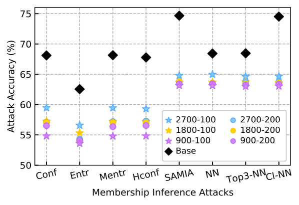
(a) RSW (H→L)



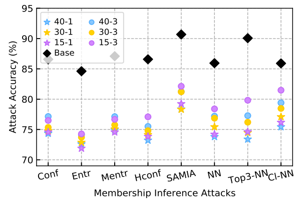
(b) RSW (L→H)



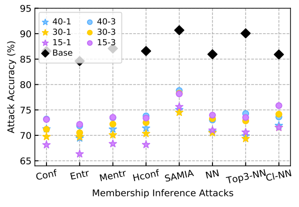
(c) SWMR (H→L)



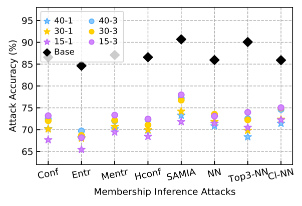
(d) SWMR (L→H)



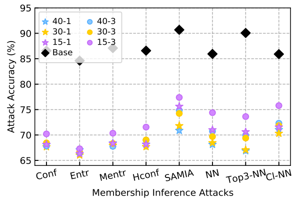
(e) RSW (H→L)



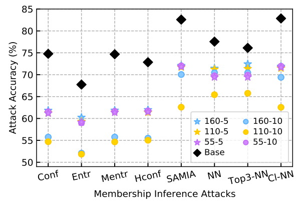
(f) RSW (L→H)



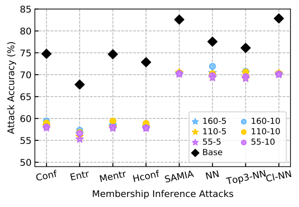
(g) SWMR (H→L)



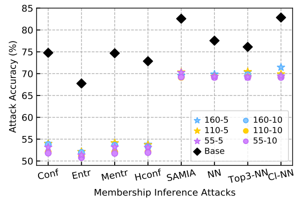
(h) SWMR (L→H)



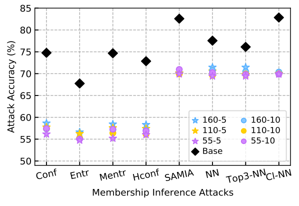
(i) RSW (H→L)



(j) RSW (L→H)



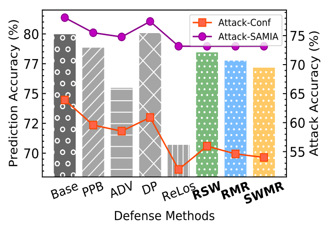
(k) SWMR (H→L)



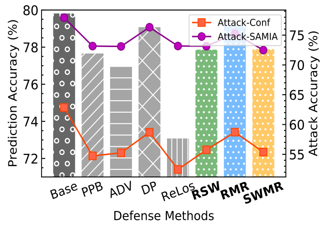
(l) SWMR (L→H)

Fig. 13: The defense effectiveness of the pruned models in RSW and SWMR. The results for CINIC-VGG16 are presented in figures (a)-(d), the results for Location-FC are presented in figures (e)-(h), and the results for Texas-FC are presented in figures (i)-(l).

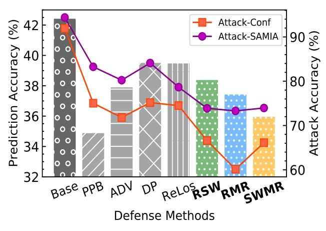
C. Defense Performance Comparison



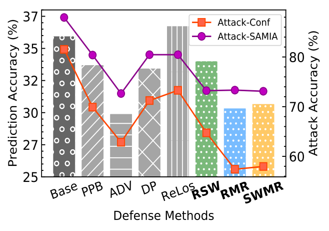
(a) CIFAR10 DenseNet121



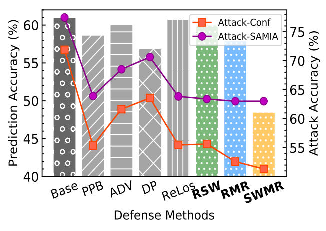
(b) CIFAR10 VGG16



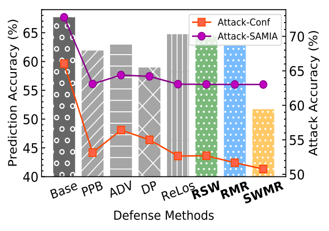
(c) CIFAR100 ResNet18



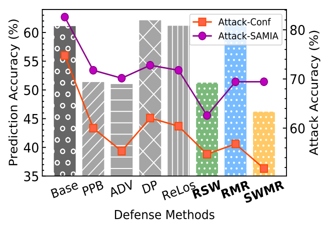
(d) CIFAR100 VGG16



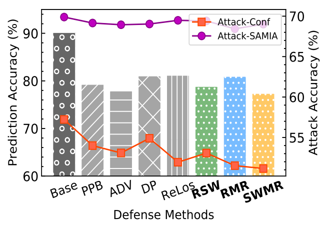
(e) CINIC ResNet18



(f) CINIC DenseNet121



(g) Texas FC



(h) Purchase FC

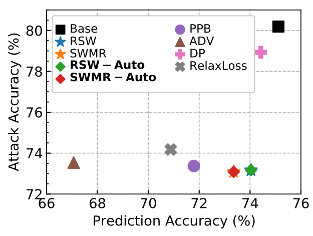
Fig. 14: Comparison of defense performance for pruned models against Conf and SAMIA under different defense methods.

D. Data Privacy Risks Before and After Our Defenses

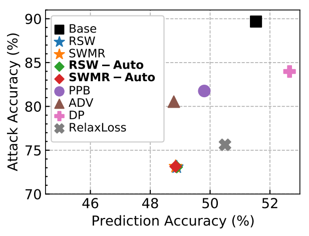
TABLE VIII: LiRA's TPR @ 1% FPR on high-risk and low-risk data before and after using our defense methods. The difference compared to the Base is shown in brackets.

CINIC VGG16					**Location FC**				
$\tau = 0.7$	**Base**	**RSW**	**RMR**	**SWMR**	$\tau = 0.6$	**Base**	**RSW**	**RMR**	**SWMR**
Low-risk $[0, 0.7)$	2.8	2.1 (-0.7)	1.3 (-1.5)	1.1 (-1.7)	Low-risk $[0, 0.6)$	30.3	25.3 (-5.0)	24.4 (-5.9)	6.2 (-24.1)
High-risk $[0.7, 1]$	15.2	5.3 **(-9.9)**	1.3 **(-13.9)**	1.2 **(-14.0)**	High-risk $[0.6, 1]$	46.2	19.1 **(-27.1)**	40.1 **(-6.1)**	8.7 **(-37.5)**

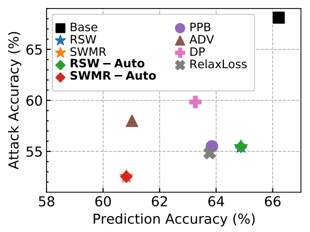
E. Adaptive Defense Improvement



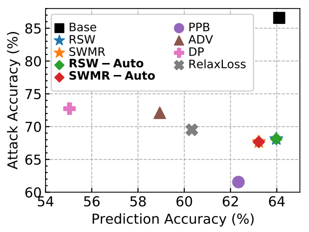
(a) ResNet18 (SAMIA)



(b) DenseNet121 (SAMIA)



(c) VGG16 (Conf)



(d) FC (Conf)

Fig. 15: The results of the pruned models under RSW and SWMR with optimal fixed window settings, improved adaptive RSW and SWMR (i.e., RSW-Auto and SWMR-Auto), and other defenses with the best performance. We use CIFAR10-ResNet18, CIFAR100-DenseNet121, CINIC-VGG16, and Location-FC.

F. Rationality of Adopting Monte Carlo Method

In the process of calculating the memorization score, for Eq. 1, we let $D' = D \backslash (\boldsymbol{x}, y)$ and use Bayesian inference to write $\Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D \backslash (\boldsymbol{x}, y))}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y]$ in the form of integral:

$$\begin{aligned} \Pr_{\mathcal{A}}[f(\boldsymbol{x}) = y | D'] &= \int p(f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y | D', \boldsymbol{\theta}) p(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \\ &= E_{p(\boldsymbol{\theta})}[p(f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y | D', \boldsymbol{\theta})] \end{aligned} \tag{7}$$

where the integral actually calculates the expectation of the probability that the model's prediction of the input $\boldsymbol{x}$ is y, given uncertain parameters. We can approximate the expectation by a corresponding average [28], which is in line with the idea of Monte Carlo sampling. In Section III-C, we divide the dataset into K disjoint subsets to ensure that each data sample has only been used in one sub-model $f_{\boldsymbol{\theta}^i}$, $i \in \{1, ..., K\}$. For a data sample, there are $K - 1$ models that have not used it, so we can use the aforementioned sampling method to approximate $\Pr_{f_{\boldsymbol{\theta}} \leftarrow \mathcal{A}(D \backslash (\boldsymbol{x}, y))}[f_{\boldsymbol{\theta}}(\boldsymbol{x}) = y]$ with the average prediction probability of $(\boldsymbol{x}, y)$ by $K - 1$ models. We denote this average as $\hat{\Pr}$:

$$\hat{\Pr} = \frac{1}{K-1} \sum_{j=1, j \neq i}^{K} p(f_{\boldsymbol{\theta}^j}(\boldsymbol{x}) = y | D', \boldsymbol{\theta}^j) \tag{8}$$

This approximation is justified, since the estimator $\hat{\Pr}$ is unbiased:

$$\begin{aligned} \mathbb{E}[\hat{\Pr}] &= \frac{1}{K-1} \sum_{j=1, j \neq i}^{K} \mathbb{E}[p(f_{\boldsymbol{\theta}^j}(\boldsymbol{x}) = y | D', \boldsymbol{\theta}^j)] \\ &= \frac{1}{K-1} \sum_{j=1, j \neq i}^{K} \Pr_{\mathcal{A}}[f(\boldsymbol{x}) = y | D'] \\ &= \Pr_{\mathcal{A}}[f(\boldsymbol{x}) = y | D'] \end{aligned} \tag{9}$$

Therefore, our method conforms to the definition of memorization score in Eq. 1, and the adopted Monte Carlo sampling method can ensure the overall effectiveness of the memorization score calculation of all samples in the training set.

Appendix B
Artifact Appendix

A. Description & Requirements

1) How to access: The artifact can be accessed at https://github.com/CactiLab/WeMeM. The artifact is also available on Zenodo with DOI: https://doi.org/10.5281/zenodo.14189484.

2) Hardware dependencies: Commodity GPUs (e.g., we used NVIDIA GeForce RTX 4090 GPU in our evaluation).

3) Software dependencies: PyTorch, torchvision, torchaudio, pandas, sklearn, numpy, matplotlib, tensorboard, pynvml, and nni.

4) Benchmarks: (1) Datasets: CIFAR10, CIFAR100, CINIC, Location, Texas, and Purchase; (2) Model architectures: For CIFAR10, CIFAR100, and CINIC, we use three DNNs: ResNet18, DenseNet121, and VGG16. For Texas, Location, and Purchase, we employ a fully connected network (FC) with two layers, containing 256 and 128 neurons, respectively. All FC layers, except the last one, utilize ReLU activation functions.

The details on how to obtain the datasets can be found in the artifact's README file.

B. Artifact Installation & Configuration

Please refer to the *Getting Started* section of the README file in the artifact to: (1) Configure the running environment and install the software dependencies; (2) Prepare the datasets.

C. Experiment Workflow

Please see the Evaluation section.

D. Major Claims

- (C1): The *Ranking-based Sliding Window (RSW)*, *Risky memory regularization (RMR)*, and the *Sliding Window and Memory Regularization (SWMR)* achieve effective defense against MIA in iteratively pruned models. Among them, the RSW effectively reduces the time cost of defense. The three methods can effectively defend against adaptive MIAs with limited model accuracy reduction. They are proven by the experiment (E1), the results of which are reported in Table V and Fig. 7.

E. Evaluation

Reference to the results in Fig. 7: Compared with the Base defense method, the three defense methods (i.e., RSW, RMR, and SWMR) can reduce the adaptive attack accuracy by 6%-20% at most when the model prediction accuracy is reduced by no more than 5%.

NOTE. We provide a detailed evaluation of the Base defense method and the three proposed methods, demonstrating that the latter achieves a better privacy-utility trade-off. We have also provided the description (in the README file) for evaluating other existing defense methods for anyone interested in performing the comprehensive evaluation.

1) Experiment (E1): The experimental steps required for the evaluation are consolidated into a single script, which can be run to reproduce the key results on two representative datasets, one from image data (i.e., CIFAR10) and one from tabular data (i.e., Location). For a detailed description, please refer to *Examples for Evaluation* section in the README file.

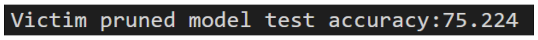
Execution. Please run `chmod +x run_shell.sh` first to make sure the script is executable. Then, run `bash ./run_shell.sh`. This script contains evaluation examples for two datasets and is expected to take approximately 16 hours to complete. Each example contains several steps, and the results on CIFAR10-ResNet18 are as follows:

- **Step 1**: Train one victim model and five shadow models (used for attack).

Results. Generate pre-trained original models and save them in the `./result` path.

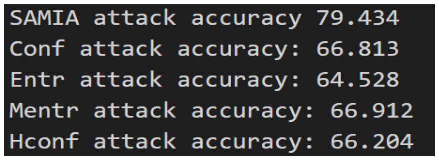
- **Step 2**: Prune with the pruning rate of 0.6 and fine-tune (shadow) models with **Base** defense method, and perform adaptive MIA attacks.

Results. The output file is saved in the `./log/cifar10_resnet18` directory. Read the file `iter_pruning_0.6_.txt`, which shows the iteratively pruned model's prediction accuracy (e.g., Fig. 16), privacy risk (e.g., Fig. 17), and the time required for iterative pruning (e.g., Fig. 18) under **Base** defense method. The privacy risk[4] is evaluated by using metric-based attacks (i.e., Conf, Entr, Mentr, Hconf) and a classifier-based attack[5] (i.e., SAMIA).



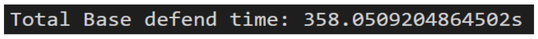
```
Victim pruned model test accuracy:75.224
```

Fig. 16: Example output showing the **prediction accuracy** on the iteratively pruned model with Base defense.



```
SAMIA attack accuracy 79.434
Conf attack accuracy: 66.813
Entr attack accuracy: 64.528
Mentr attack accuracy: 66.912
Hconf attack accuracy: 66.204
```

Fig. 17: Example output showing the **privacy risk** on the iteratively pruned model with Base defense.



```
Total Base defend time: 358.0509204864502s
```

Fig. 18: Example output showing the **time required** for the iterative pruning under Base defense.

- **Step 3**: Prune with the pruning rate of 0.6 and fine-tune (shadow) models with **RSW** defense method, and perform adaptive MIA attacks.

Results. Read the output file `iter_pruning_0.6_slide_re.txt`, which shows the iteratively pruned model's prediction accuracy, privacy risk, and the time required for iterative pruning under **RSW** defense method.

[4]All the attacks we used in the evaluation examples are the same, and the specific attacks we used will not be repeated in the following descriptions.

[5]Since classifier-based attacks take a long time to evaluate, we select the representative SAMIA attack for evaluation.

- **Step 4**: Prune with the pruning rate of 0.6 and fine-tune (shadow) models with **RMR** defense method, and perform adaptive MIA attacks.

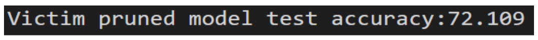
Results. Read the output file `iter_pruning_0.6_ml2.txt`, which shows the iteratively pruned model's prediction accuracy, privacy risk, and the time required for iterative pruning under **RMR** defense method.

- **Step 5**: Prune with the pruning rate of 0.6 and fine-tune (shadow) models with **SWMR** defense method, and perform adaptive MIA attacks.

Results. Read the output file `iter_pruning_0.6_slide_ml2.txt`, which shows the iteratively pruned model's prediction accuracy, privacy risk, and the time required for iterative pruning under **SWMR** defense method.

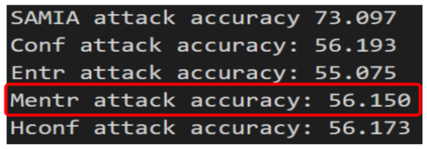
2) Interpreting the results: The prediction accuracy, attack accuracy, and time required under the proposed three defense methods can be obtained from the corresponding output files of Steps 3 to 5. The evaluation conclusion is drawn by comparing the results obtained in these three steps with the Base results in Step 2, respectively. The conclusion can be compared with those in Fig. 7 and Table V.

For example, The output results of the RSW defense method obtained in Step 3 compared with the Base method show that the prediction accuracy of the pruned model is reduced by no more than 5% (e.g., Fig. 16 vs Fig. 19), while the attack accuracy is reduced by 10.762% at most (e.g., Mentr attack accuracy in Fig. 17 and Fig. 20). In addition, the time required for iterative pruning under the RSW method is better than that under RMR and SWMR, which is less than that under the Base method (e.g., Fig. 18 vs Fig. 21), which accords with the conclusion of Table V.



```
Victim pruned model test accuracy:72.109
```

Fig. 19: Example output showing the **prediction accuracy** on the iteratively pruned model with RSW defense.



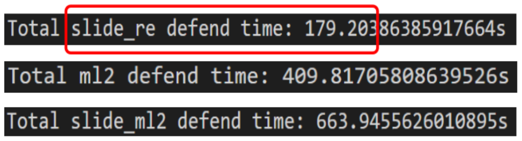
```
SAMIA attack accuracy 73.097
Conf attack accuracy: 56.193
Entr attack accuracy: 55.075
Mentr attack accuracy: 56.150
Hconf attack accuracy: 56.173
```

Fig. 20: Example output showing the **privacy risk** on the iteratively pruned model with RSW defense.

NOTE. Since model training is a statistical learning process, the model training and fine-tuning (i.e., retraining) required for evaluation may cause the final results to differ from those reported in the paper (regarding specific values) but will not affect the final conclusions.

F. Customization

The provided additional running script `run_custom.sh` is used for evaluating other defense methods. Please run the script directly `bash ./run_custom.sh` to conduct experiments. In this script, Step 1 is a pretraining process. If the `run_shell.sh` described in Appendix B-E1 has already been executed, Step 1 in this script can be skipped. Steps 2-5 in this script are the experimental processes of the ppb, adv, relaxloss, and dp defense methods used in this paper. The hyperparameters for these defense methods can be adjusted based on item 9 of the *NOTE* section in the README file.



```
Total slide_re defend time: 179.20386385917664s
Total ml2 defend time: 409.81705808639526s
Total slide_ml2 defend time: 663.9455626010895s
```

Fig. 21: Example output showing the **time required** for the iterative pruning under RSW, RMR, and SWMR defenses.