A. Appendix

A.1.

A.2. Dataset and Training Details

The CIFAR-100 dataset [20] consists of 100 categories with 60,000 32 $\times$ 32 color images, where 50,000 images are allocated for training and 10,000 images for testing. The Dermnet dataset [1] includes 23 categories with a total of 19,500 images, where 15,500 images are allocated for training and 4,000 images for testing. Since the images have varying sizes, we cropped them to a size of 64$\times$64 pixels. The Tiny Imagenet datasets [22] has 100000 images of 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. The member dataset we use is the splited training datatset of target client and the non-member datatset is consist of the one-tenth hold-out test dataset and the sum of one-tenth training datset of the other clients. In the IID setting, we uniformly and randomly distribute the samples of each class to each client. In the Non-IID setting, we make the labels of each client's training data follow the Dirichlet distribution[15]. For image generative task with diffusion model, we use 10 classes for training model and attacking membership privacy. We run image classification tasks on AlexNet and ResNet18 with NVIDIA 2060 GPU and run image generation task on laten diffusion model with NVIDIA A100 GPU. The training parameters details and dataset splited method of federated learning are shown in Table 3.

A.3. Defense Methods

A.3.1. Gradient Perturbation

Client-level Differential Privacy. Differential Privacy (DP) [8, 49] hides the membership of individual data by clipping the gradients at the client level and adding Gaussian noise. The magnitude of the noise controls the strength of privacy protection: the larger the noise, the better the privacy protection, but the worse the model's performance. In the experiment, we set the DP noise standard deviation from 0.01 to 0.5 to achieve different levels of defense.

Gradient Quantization. Gradient quantization [12, 30] is a technique used to reduce the precision of gradient updates and mitigate information leakage. This algorithm quantizes the values of gradients into discrete approximations, reducing the precision of the gradients. By reducing the detailed information in the gradients, it lowers the sensitivity to individual data and improves privacy protection. The number of bits used for quantization affects the privacy protection effectiveness, where fewer bits introduce larger gradient errors but provide better privacy protection. In the experiment, we set the number of bits from 1 to 10 to achieve different levels of defense.

Gradient Sparsification. The gradient sparsification algorithm [11, 35, 40] reduces the risk of information leakage by setting smaller absolute value elements in the gradient to zero. The fewer non-zero elements in the gradient, the less privacy leakage occurs. In the experiment, we set the rate of gradient elements sparsified from 0.1 to 0.99 to achieve different levels of defense.

A.3.2. Data Replacement

MixUp. MixUp method [9, 47] trains neural networks on composite images created via linear combination of image pairs. It has been shown to improve the generalization of the neural network and stabilizes the training. The coefficient of the linear combination is sampled from a Beta Distribution. We set the Beta Distribution parameter from 1e-5 to 1e5 to achieve different levels of defense.

Data Augmentation. Data Augmentation [37] includes cropping, shifting, rotating, flipping, shearing, and color jittering. We combine these augmentation schemes in different amounts to achieve different levels of defense.

Data Sampling. In each local training epoch, clients may choose to sample a portion of the training data instead of using the entire dataset [23]. We set the portion from 0.1 to 1.0 to achieve different levels of defense.

A.4. Evaluation Metrics

Utility loss (Test error rate). In this paper, we quantify the utility loss by using the test error as a metric. The test error measures the accuracy of the model on a separate test dataset, where a lower test error indicates better model utility. The worst possible test error rate is 1, which means that the model makes incorrect predictions for all instances in the test dataset.

Privacy Leakage (AUC and attack TPR). We consider attacks as a binary classification task, and the TPR@FPR of the AUC can be used to measure the accuracy of the classification, which represents the effectiveness of the attack. TPR@low FPR is a metric recently proposed for measuring MIA (Membership Inference Attack). It focuses more on the data that is most susceptible to attacks, and researchers believe that using it as a metric can better characterize privacy protection in worst-case scenarios.

TPR (True Positive Rate) and FPR (False Positive Rate) are two important metrics used to evaluate the performance of binary classification models, such as machine learning algorithms or diagnostic tests. They are calculated as follows:

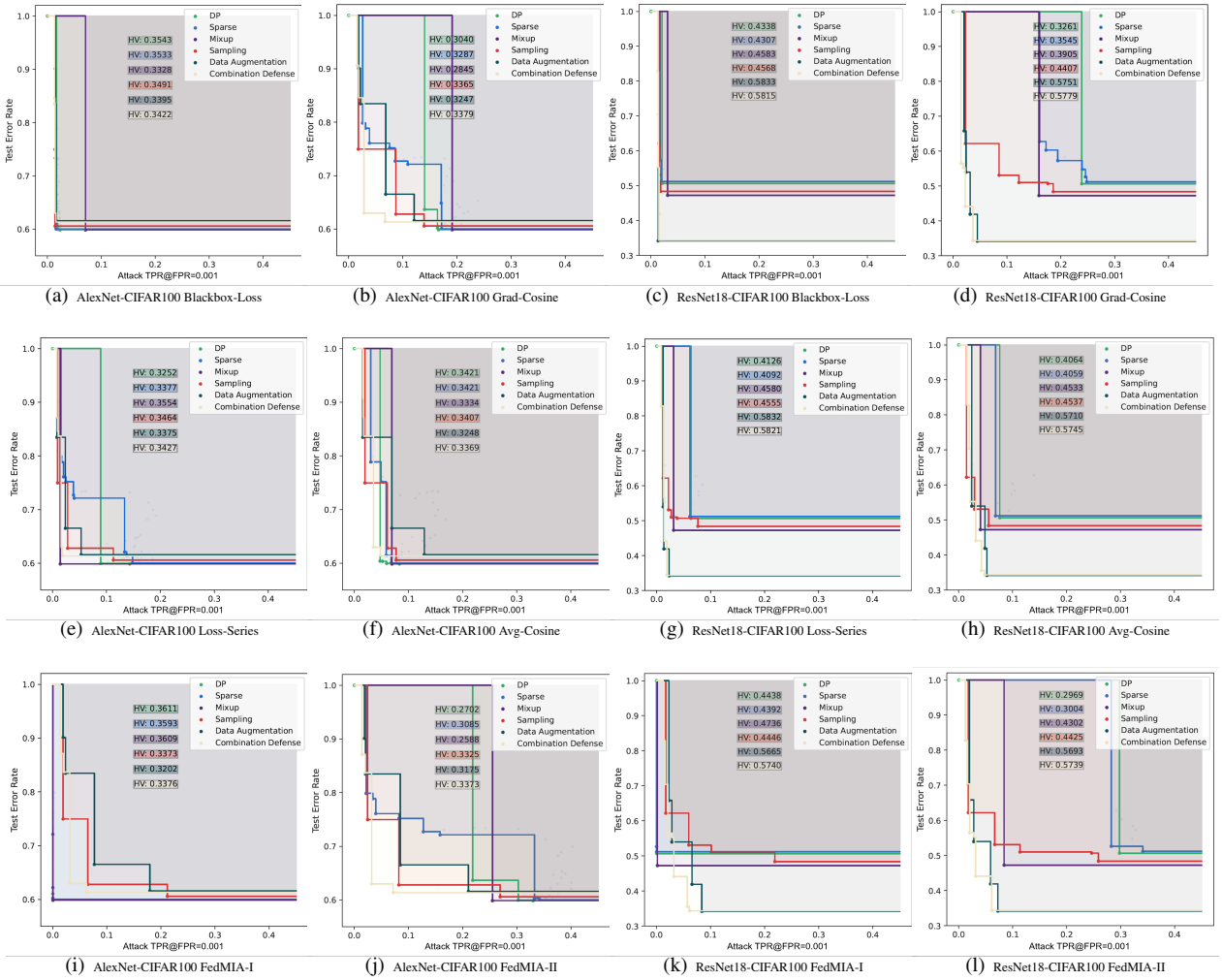
$$\text{TPR} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{FPR} = \text{FP} / (\text{FP} + \text{TN})$$

where: TP (True Positives) represents the number of positive instances correctly classified as positive. FN (False Negatives) represents the number of positive instances incorrectly classified as negative. FP (False Positives) represents the number of negative instances incorrectly classified

Table 3: Training parameters for federated learning in this paper

Dataset	CIFAR100	Dermnet	Tiny ImageNet
Models	AlexNet, ResNet18	AlexNet, ResNet18	Laten Diffusion Model
Communication epoch	300	300	20
Optimizer	SGD	SGD	Adam
Initial learning rate	0.1	0.1	0.001
Learning rate decay	0.99 at each epoch	0.99 at each epoch	Adaptive
Number of clients	10	10	10
Training set size for one client	5000	1500	1000
Testing set size	10000	4500	1000



(a) AlexNet-CIFAR100 Blackbox-Loss (b) AlexNet-CIFAR100 Grad-Cosine (c) ResNet18-CIFAR100 Blackbox-Loss (d) ResNet18-CIFAR100 Grad-Cosine

(e) AlexNet-CIFAR100 Loss-Series (f) AlexNet-CIFAR100 Avg-Cosine (g) ResNet18-CIFAR100 Loss-Series (h) ResNet18-CIFAR100 Avg-Cosine

(i) AlexNet-CIFAR100 FedMIA-I (j) AlexNet-CIFAR100 FedMIA-II (k) ResNet18-CIFAR100 FedMIA-I (l) ResNet18-CIFAR100 FedMIA-II

Figure 6: Figure (a)-(f) demonstrate the TPR@FPR=0.001 of various defence (including client-level differential privacy (green line) [8], sparsification (blue line) [35], mixup (purple line) [47], data sampling (red line) [23], data augmentation (deep blue line) and gradient, combination of data augmentation and sampling (yellow line)) under three attacks (Blackbox-Loss [44], Loss-Series [10], FedMIA-I, Grad-Cosine, Avg-Cosine [24] and FedMIA-II are first, second and third row respectively). A larger hypervolume (HV) [51] indicates a better Pareto front of privacy and utility.

as positive. TN (True Negatives) represents the number of negative instances correctly classified as negative.

Hypervolume $HV()$. In order to compare Pareto fronts achieved by different defense algorithms, we need to quan-

tify the quality of a Pareto front. To this end, we adopt the hypervolume (HV) indicator [51] as the metric to evaluate Pareto fronts. Definition 1 formally defines the hypervolume.

Definition 1 (Hypervolume Indicator)**.** *Let* $z = \{z_1, \cdots, z_m\}$ *be a reference point that is an upper bound of the objectives* $Y = \{y_1, \ldots, y_m\}$*, such that* $y_i \leq z_i$*,* $\forall i \in [m]$*. the hypervolume indicator* $HV_z(Y)$ *measures the region between* Y *and* z *and is formulated as:*

$$HV_z(Y) = \Lambda\left(\left\{q \in \mathbb{R}^m \middle| q \in \prod_{i=1}^{m}[y_i, z_i]\right\}\right) \tag{13}$$

where $\Lambda(\cdot)$ *refers to the Lebesgue measure.*

We set the reference point z of privacy leakage and utility loss to be 1 and 100% respectively.

B. More experiment

Figure 6 demonstrates the privacy-utility tradeoff against different attacks under various defense methods.

C. Proof of Theorem 1

Theorem 2. *Given the threshold* δ*, let* $\mathbb{V}_t$ *be the member sets estimated by* $\hat{\Lambda}^t$ *and* δ *in communication round* t*. Let* $\tilde{\mathbb{V}}$ *be the member sets estimated by* $\tilde{\Lambda}$ *and* δ*. Then we have*

$$\tilde{\mathbb{V}} \subset (\mathbb{V}_1 \cup \cdots \cup \mathbb{V}_T). \tag{14}$$

Proof. We employ proof by contradiction to establish the theorem. Assume there exists an element $v \in \mathcal{V}$ such that $v \notin (\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_T)$.

By definition, the set $\mathcal{V}_t$ is defined as

$$\mathcal{V}_t = \{v \mid v \in \mathcal{V},\ \hat{\Lambda}_t(v) \geq \delta\},$$

which represents the set of members determined by Eq. (4) in the main text. Additionally, let $\mathcal{V}_t^n$ denote the non-member set determined by Eq. (4).

Now, consider an element $v \notin (\mathcal{V}_1 \cup \cdots \cup \mathcal{V}_T)$. This implies that $\hat{\Lambda}_t(v) < \delta$ for all t. Consequently, we have:

$$\tilde{\Lambda}(v) = \frac{1}{T}\sum_{t=1}^{T}\hat{\Lambda}_t(v) < \delta.$$

This result indicates that $v \notin \mathcal{V}$, which contradicts the assumption that $v \in \mathcal{V}$.

Thus, the assumption leads to a contradiction, and the proof is complete.

□