Deceive-to-Defend: Synthesized Data Interpolation Against Gradient Inversion in Federated Learning

Yue Lin[†‖], Shih-Jui Liang[†‖], Liang-Hsuan Liu[†], Yu-Wen Chen[‡], Jian-Jhih Kuo[†*], and Ren-Hung Hwang[§†]
[†]Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan
[‡]Computer Systems Technology, New York City College of Technology, Brooklyn, New York, USA
[§]College of Artificial Intelligence, National Yang-Ming Chiao Tung University, Tainan, Taiwan

***Abstract*—Federated Learning (FL) enables multiple client devices to train a global model collaboratively by sharing only local gradient updates, thus preserving data privacy. Despite its advantages, FL remains vulnerable to Gradient Inversion Attack, where attackers exploit gradient updates to reconstruct private data. To counter these advanced threats, we propose Latent Interpolation Data Synthesis (LIDS), a client-side privacy-preserving training scheme that obfuscates per-sample gradient signals via structured batch construction and latent oversampling. LIDS groups semantically similar samples and generates synthesized variants, reducing the uniqueness of each gradient contribution while preserving model convergence. Experimental results show that LIDS reduces attacker reconstruction quality, with Peak Signal-to-Noise Ratio dropping from 27 to 17. On CIFAR-10, the attacker's reconstruction label accuracy and fraction of successful reconstructions are reduced by up to 64% and 92%; on CIFAR-100, by 68% and 96%. These privacy gains come at a modest cost of only 11% test accuracy degradation in CIFAR-100.**

I. Introduction

Traditional Machine Learning (ML) relies on centralized data collection and computation, which introduces significant risks of data leakage and privacy breaches. Federated Learning (FL) addresses these concerns by enabling collaborative model training directly on client devices. Instead of uploading raw data, each client computes local model updates, which are then aggregated by a central server to update the global model. By keeping data decentralized, FL significantly reduces the risk of data exposure and enhances client privacy.

Despite its advantages, the decentralized nature of FL introduces various security risks. Numerous studies have demonstrated that even when clients only share model gradients, there remains a high risk of data leakage [1]–[5]. Attackers can exploit mathematical analysis and ML techniques to infer or even reconstruct the original client data from the uploaded gradients. Such attacks are commonly referred to as *Gradient Inversion Attack (GradInv)* [6], where training samples are recovered directly from gradients.

While early studies focused on honest-but-curious servers that passively infer client data from received gradients, recent work highlights a more severe threat: malicious servers that actively manipulate the global model. By introducing subtle perturbations or employing gradient disaggregation techniques, a malicious server can significantly amplify privacy leakage beyond what is achievable in passive settings [7]. These attacks are typically categorized into two types: *boosted analytical* and *example disaggregation*, depending on the underlying mechanism used to extract sensitive information [8].

Boosted analytical attacks typically manipulate the model architecture or global parameters to preserve input data as it flows to the dense layers [7], [9], [10]. For instance, attackers may prepend fully connected layers with trap weights, which serve as re-scaling components to enable precise data recovery [9]. Alternatively, malicious weights may be crafted to bypass defenses such as gradient obfuscation [10]. Although these methods can achieve near-perfect reconstruction, they often involve conspicuous architectural modifications and can be detected by inspecting the model weights on the client side [8]. In contrast, example disaggregation attacks aim to isolate individual data points by manipulating the training process so that the aggregated gradient closely approximates that of a single example [11], [12]. This undermines the protection offered by gradient aggregation and enables data reconstruction. While effective, such attacks often involve adversarial control over training conditions, and can be detected by analyzing anomalous gradient patterns or update behavior [8].

Building on this line of research, recent work has explored more subtle forms of disaggregation by modifying model parameters in less detectable ways. For example, a malicious server can train a secret decoder that projects client gradients into a hidden space, suppressing all gradient components that do not satisfy a predefined property $\mathcal{P}$ [8]. The goal is to ensure that only one image per batch satisfies $\mathcal{P}$, effectively isolating it from the others. The server-side decoder can then reconstruct this isolated input from its projected gradient, enabling data theft without altering the model architecture. This attack updates the global model using standard Stochastic Gradient Descent (SGD), avoiding manual weight adjustments and making it difficult to detect through model inspection. Moreover, the leakage is embedded in a learned latent space, evading gradient-based anomaly detection. We refer to this class of attacks as Gradient Filtering-based Example Disaggregation Attack (GF-EDA), and focus on the Secret Embedding and Reconstruction (SEER) framework [8] as a representative example. To the best of our knowledge, no existing defense can effectively counter this class of subtle, decoder-driven attacks.

In this paper, we propose a client-side privacy-preserving training scheme to defend against SEER. Our goal is to prevent

‖: equal contributions;*: corresponding author (lajacky@cs.ccu.edu.tw)
This work was supported in part by the National Science and Technology Council, Taiwan, under Grants 111-2628-E-194-001-MY3, 113-2221-E-194-040-MY3, 114-2221-E-A49-121-MY3, 114-2628-E-194-002-MY3, and 114-2813-C-A49-143-E.

effective data reconstruction and ensure that any recovered images lack meaningful or identifiable content. Since these attacks rely on isolating a single image in the batch that satisfies a specific property $\mathcal{P}$, our key idea is to ensure that multiple images in the same batch share similar properties. This confuses the attacker's decoder, making the extracted gradient a blend of several examples rather than a disaggregated target. As a result, the reconstructed output becomes ambiguous and uninformative. However, the attacker's targeted property $\mathcal{P}$ is unknown and diverse. To address this, we group semantically similar data points into the same batch, increasing the chance that multiple examples satisfy $\mathcal{P}$. Furthermore, when similar samples are insufficient, we leverage a deep-learning-based variant of Synthetic Minority Over-sampling Technique (SMOTE) [13] to synthesize new samples resembling the client's data, further disrupting the attacker's ability to isolate and reconstruct individual examples.

To counter this novel and hard-to-detect threat, we develop a client-side privacy protection framework that proactively disrupts the attacker's disaggregation process. Our design faces *three key challenges*. 1) *Autonomous sample synthesis*: enabling each client to generate protective samples without external coordination independently; 2) *Property-agnostic consistency*: ensuring that synthesized samples share sufficient properties with real data, despite the attacker's targeted property $\mathcal{P}$ being unknown; and 3) *Robust training integration*: maintaining training randomness and model generalization while incorporating synthesized samples. We address these challenges through *latent space interpolation*, *dynamic similarity regulation*, and *structured batch construction*, as described in Section III. Experimental results in Section IV show that our framework significantly reduces the Peak Signal-to-Noise Ratio (PSNR) of reconstructed images from 27 to 17, while preserving test accuracy within 11% degradation margin. These results confirm the effectiveness of our approach in mitigating SEER without sacrificing model utility.

II. Attack Model

In a SEER scenario, the FL system consists of multiple clients and a malicious server that aims to steal private data from uploaded gradients. As in standard FL, each client computes and uploads the aggregated gradient $\boldsymbol{g}$ over a local batch. The server uses $\boldsymbol{g}$ to update the global model. However, the malicious server secretly trains a decoder d to project $\boldsymbol{g}$ into a latent space, filtering out gradient components that do not satisfy a specific property $\mathcal{P}$. This process effectively treats the local model as an encoder that transforms a data batch into the aggregated gradient $\boldsymbol{g}$. The attacker carefully chooses $\mathcal{P}$ so that only one data point in most batches satisfies it. As a result, the decoder filters out all other contributions, isolating the gradient projection of the targeted data point. A separate reconstructor r is then applied to recover the original input from this filtered gradient, completing the data reconstruction.

This encoder-decoder attack framework is trained end-to-end using auxiliary data and gradient updates, without requiring explicit architectural modifications. Consequently, it bypasses both weight-space and gradient-space detection methods. As gradient-filtering attacks become increasingly sophisticated, traditional detection-based defenses are proving insufficient. Motivated by this challenge, we propose shifting the defense paradigm: rather than detecting attacks after the fact, we aim to proactively strengthen privacy at the data source, making disaggregation and reconstruction fundamentally more difficult.

III. The Proposed Framework – LIDS

Early studies have explored using generative models, such as Generative Adversarial Networks (GANs) or diffusion models, to defend against traditional model inversion attacks in standard FL settings, which typically exploit access to the model's outputs (e.g., softmax probabilities or logits) to infer sensitive attributes or reconstruct training data. For instance, [14] and [15] demonstrate how clients can train GANs to generate samples that retain statistical properties of real data, thereby masking individual gradients. However, such defenses are not designed to counter more advanced threats like SEER, where attackers manipulate model updates and train secret decoders to isolate and reconstruct individual samples from aggregated gradients. Moreover, these methods often rely on shared generative infrastructure or inter-client coordination, incurring high computation and communication overhead.

To proactively reduce the success rate of such attacks, we propose a lightweight client-side defense framework, termed Latent Interpolation Data Synthesis (LIDS), to disrupt SEER's disaggregation process without relying on shared infrastructure or inter-client coordination. As illustrated in Fig. 1, LIDS has two phases, where each client first trains an autoencoder solely on its local data and synthesizes new samples via a latent space interpolation strategy to generate visually distinct yet feature-similar images that preserve gradient-relevant characteristics. Then, the real and synthesized inputs in a batch become indistinguishable to the attacker, effectively obfuscating per-sample gradient signals and reducing attack success rate.

The first phase of LIDS, Dataset Construction Phase (DCP), ensures that each client constructs a local dataset with sufficient intra-batch similarity to resist targeted isolation. Without requiring external communication or shared generative models, each client independently projects its data into a latent space using a lightweight *autoencoder* [16], where the encoder $E(\cdot)$ maps inputs to latent representations and the decoder $D(\cdot)$ reconstructs them. Note that this client-side reconstruction module is entirely local and unrelated to the attacker's secret decoder. Then, synthesized samples are generated by a SMOTE-like latent space interpolation, preserving the original data's core structure while introducing controlled variation, which will be detailed in Section III-A. This design directly addresses the first challenge, autonomous sample synthesis, by avoiding any need for external communication or third-party components, and enhances privacy by increasing sample overlap in semantic properties, effectively weakening the disaggregation effectiveness of SEER.

Despite the interpolation strategy's improvements, a key challenge remains: the attacker can arbitrarily select a property $\mathcal{P}$ before distributing the global model and consistently target it in training rounds. Since clients do not know the chosen

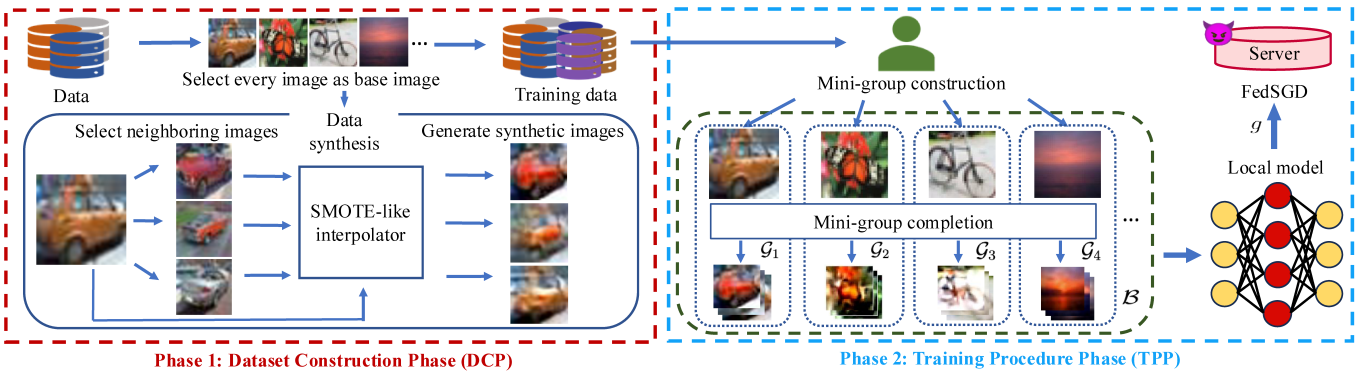


Fig. 1: Architecture of Latent Interpolation Data Synthesis (LIDS).

property, defenses based on a fixed set of attributes are insufficient. To further address this within DCP, LIDS explicitly tackles the second challenge, property-agnostic consistency, by incorporating a dynamic similarity regulation. By setting a threshold τ during latent space interpolation, each synthesized image is constrained to retain sufficient visual similarity to its source. This adaptive control increases the likelihood that multiple samples within a batch share the unknown target property specified by the attacker, thereby reducing the effectiveness of selective isolation under SEER.

However, introducing a large number of synthesized samples into training batches may reduce data diversity and make the model overfit to synthetic artifacts, exacerbating the difficulty of addressing the third challenge, robust training integration. To mitigate this, LIDS further introduces a mini-group construction strategy in the second phase, Training Procedure Phase (TPP). Each mini-group $\mathcal{G}$ contains one original sample x_{base} and a fixed number $S-1$ of synthesized samples. Including a real sample in every mini-group ensures the model remains anchored to authentic data, maintaining learning stability and preventing drift toward synthesized distributions. Training batches $\mathcal{B}$ are then formed by combining k such mini-groups, preserving both diversity and stochasticity throughout training. This structured design enables LIDS to defend against gradient leakage without significantly degrading test accuracy.

The details of the DCP and TPP are provided in Algorithm 1, Algorithm 2, and the following subsections.

A. Dataset Construction Phase (DCP)

DCP introduces a SMOTE-like latent space interpolation to generate structured variation while preserving semantic consistency, improving the resilience of the dataset against SEER's disaggregation attack. Specifically, the original dataset $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$ is augmented by generating synthesized samples through latent space interpolation, where x_i and y_i are an input image and its label, and the semantic similarity within each batch is increased while preserving label consistency, as listed in the following four steps.

1) Class-wise Grouping (Lines 2–3, Algorithm 1): For each class c, all samples $\mathcal{D}_c \subset \mathcal{D}$ are grouped to ensure that the generated synthesized samples maintain consistent class labels. This preserves the semantic structure of the training set.

Algorithm 1: LIDS: Dataset Construction

Input: Original dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$; autoencoder model; number of samples per mini-group S; PSNR threshold τ; interpolation parameters (α_0, $\alpha_{\max}$, $\Delta\alpha$)
Output: Augmented dataset $\mathcal{D}'$

1 Initialize augmented dataset $\mathcal{D}' \leftarrow \emptyset$;
2 **for** *each class* c **do**
3 Form subset $\mathcal{D}_c \subset \mathcal{D}$ with label c;
4 **for** *each image* $x_i \in \mathcal{D}_c$ **do**
5 Encode x_i: $z_i \leftarrow E(x_i)$;
6 Find $S-1$ nearest neighbors $\mathcal{N}_i$ based on MSE;
7 **for** *each neighbor* $x_j \in \mathcal{N}_i$ **do**
8 Encode x_j: $z_j \leftarrow E(x_j)$;
9 Initialize $\alpha \leftarrow \alpha_0$;
10 **repeat**
11 $z_{\text{syn}} \leftarrow \alpha z_i + (1-\alpha) z_j$;
12 $\tilde{x}_{\text{syn}} \leftarrow D(z_{\text{syn}})$;
13 Compute PSNR between $\tilde{x}_{\text{syn}}$ and x_i;
14 **if** PSNR $\geq \tau$ **then**
15 Apply augmentation to $\tilde{x}_{\text{syn}}$ to obtain x_{syn};
16 Add (x_{syn}, c) to $\mathcal{D}'$;
17 **else**
18 $\alpha \leftarrow \alpha + \Delta\alpha$;
19 **until** PSNR $\geq \tau$ *or* $\alpha > \alpha_{\max}$;

2) Image Pairing and Encoding (Lines 4–6, Algorithm 1): Each image $x_i \in \mathcal{D}_c$ is paired with $S-1$ nearest neighbors based on pixel-level Mean Square Error (MSE) to form a neighbor set $\mathcal{N}_i$, and is encoded into a latent representation using a client-side autoencoder:

$$z_i = E(x_i) \text{ and } z_j = E(x_j), \quad \forall x_j \in \mathcal{N}_i. \tag{1}$$

3) Latent Space Interpolation and Filtering (Lines 7–19, Algorithm 1): Latent vectors are interpolated to generate synthesized features:

$$z_{\text{syn}} = \alpha \cdot z_i + (1-\alpha) \cdot z_j, \tag{2}$$

where the interpolation coefficient α is adjusted to ensure that the peak PSNR between the reconstructed sample and x_i exceeds a threshold τ.

4) Image Reconstruction and Enhancement (Line 15, Algorithm 1): Each z_{syn} is decoded to form a preliminary image:

$$\tilde{x}_{\text{syn}} = D(z_{\text{syn}}) \tag{3}$$

Algorithm 2: LIDS: Training Procedure

Input: Augmented dataset $\mathcal{D}'$; original dataset $\mathcal{D}$; number of groups per batch k; number of samples per mini-group S

1 Split $\mathcal{D}$ into batches $\{\mathcal{B}_1, \ldots, \mathcal{B}_M\}$ of k samples each;
2 $\mathcal{B}_{\text{all}} \leftarrow \emptyset$;
3 **for** *each batch* $\mathcal{B}_m$ **do**
4 **for** *each base sample* $x_{base} \in \mathcal{B}_m$ **do**
5 Fetch $S-1$ synthetic samples x_{syn} from $\mathcal{D}'$;
6 Form mini-group $\mathcal{G} \leftarrow \{x_{\text{base}}\} \cup \{x_{\text{syn}}\}$;
7 Add $\mathcal{G}$ to $\mathcal{B}_m$;
8 Add batch $\mathcal{B}_m$ to $\mathcal{B}_{\text{all}}$;
9 **for** *each round* t **do**
10 Receive global model parameters θ^t from server;
11 Randomly select one batch $\mathcal{B}_{m^*}$ from $\mathcal{B}_{\text{all}}$;
12 Compute gradient: $g \leftarrow \nabla \mathcal{L}(\theta^t; \mathcal{B}_{m^*})$;
13 Submit gradient g to server;

To enhance visual quality, a lightweight stochastic augmentation is applied:

$$x_{\text{syn}} = \text{Augment}(\tilde{x}_{\text{syn}}) \tag{4}$$

The image x_{syn} is then added to the extended dataset $\mathcal{D}'$.

B. Training Procedure Phase (TPP)

TPPs follow the mini-group design to construct training batches by combining real samples from the original dataset with synthesized samples generated in DCPs, preserving training diversity while obfuscating gradient signals associated with individual data points. TPPs consist of the following steps.

1) Mini-Group Formation: Given a batch size B and mini-group size S, the number of mini-groups is $k = B/S$. Each mini-group $\mathcal{G}_i$ consists of S images, and the complete batch $\mathcal{B}$ is defined as: $\mathcal{B} = \mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$.

2) Base Image Selection (Line 1, Algorithm 2): For each mini-group $\mathcal{G}_i$, one real image x_{base} is uniformly sampled from the original dataset: $x_{\text{base}} \sim \mathcal{D}$.

3) Mini-Group Completion (Lines 5–6, Algorithm 2): The remaining $S-1$ images in each group are selected from the synthesized images generated based on x_{base} in $\mathcal{D}'$. This maintains semantic closeness within the group. If the synthesized samples are insufficient, previously generated ones are reused.

4) Batch Assembly (Line 7, Algorithm 2): All mini-groups are aggregated to form the final batch $\mathcal{B}$, preserving intra-group similarity and inter-group diversity.

This batch construction strategy conceals each real image among visually similar synthesized companions, significantly increasing the difficulty of isolating and reconstructing private data. Empirical results show a substantial reduction in reconstruction quality, with the average PSNR dropping from 27 to 17. While this approach introduces slight accuracy degradation, it yields a strong privacy-utility trade-off. During TPP, client periodically returns the update gradient g to the central server for standard federated aggregation.

IV. Performance Evaluation

We evaluate LIDS with standard FL via Federated Stochastic Gradient Descent (FedSGD). The code is available online at https://git.new/LIDS. Our experiments aim to assess both the effectiveness of privacy protection and the impact on test accuracy under SEER. The experimental settings are as below.

A. Dataset and Experiment setup

1) Dataset: We use CIFAR-10 and CIFAR-100 as benchmarks, both with $60{,}000$ color images of size 32×32 pixels, split into $50{,}000$ training and $10{,}000$ test samples [17]. CIFAR-10 includes 10 object categories such as airplanes, automobiles, birds, and cats, and is commonly used in image classification tasks. CIFAR-100 shares the same image format and sample count but contains 100 fine-grained categories grouped into 20 superclasses, making the classification task more challenging.

2) Model setting: We follow the setup used in SEER to adopt ResNet-18 [18] as the global model for fair comparison with prior work. ResNet-18 is a compact convolutional neural network widely used in image classification, balancing computational efficiency and predictive performance.

3) Evaluation Metrics: To evaluate image similarity between client data and attacker reconstructions, we employ three commonly used metrics: PSNR, Structural Similarity Index (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). In addition, we adopt two PSNR-based metrics from [8] for evaluating SEER: (i) fraction of successful reconstructions (REC), defined as the proportion of batches where the reconstructed image achieves a PSNR above 19 [19], and (ii) the average PSNR across all attacked batches (PSNR-All).

Since PSNR is sensitive to color distortions and may not accurately reflect whether the reconstructed content remains meaningful, we additionally adopt a semantic-level evaluation using a DINOv2 model finetuned on the CIFAR-10 and CIFAR-100 [20]. Specifically, we measure the classification accuracy of reconstructed images and compare it to that of the original inputs, providing a more reliable assessment of whether the recovered images preserve high-level content rather than merely low-level pixel similarity. We refer to this metric as reconstruction label accuracy (RLA).

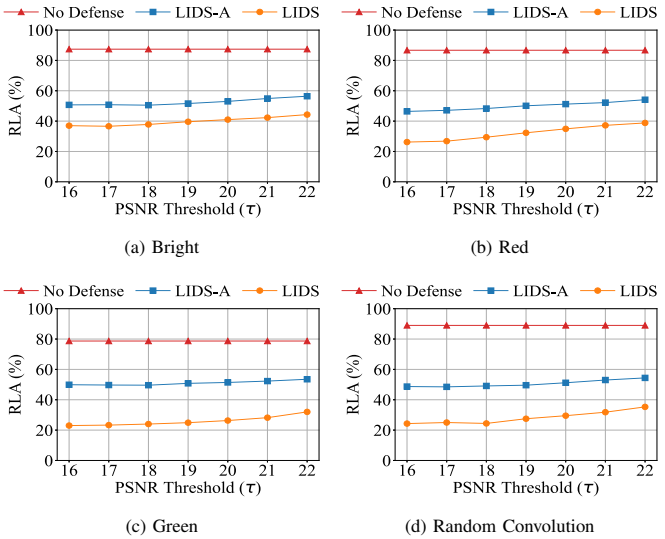
B. Implementation Details

1) Client device setting: Each client trains a convolutional autoencoder to generate synthesized samples via latent space interpolation, following [16]. The encoder $E(\cdot)$ consists of four convolutional layers with filter sizes $[64, 128, 256, 64]$, each using 4×4 kernels, stride of 2, and ReLU activations. This produces a latent representation of size $64 \times 2 \times 2$. The decoder $D(\cdot)$ mirrors this architecture using transposed convolutions for image reconstruction. The autoencoder is optimized using the Adam optimizer with a learning rate of $\eta_a = 0.001$, batch size 128, and trained for 3000 epochs. The generated synthetic samples preserve the visual structure of the base images while incorporating features from neighboring examples, which increases similarity among batch samples and makes gradient disaggregation more difficult for the attacker.

Other parameters used in the full client training pipeline are listed in Table I, where $\mathcal{E}$ is the number of local training epochs, B is the batch size for local model updates, S is the number of samples per mini-group, and τ is the PSNR threshold. α_0, $\alpha_{\max}$, and $\Delta\alpha$ specify the initial value, maximum value, and step interval for the latent interpolation strength, respectively.

TABLE I: Experimental Setting

Parameters	$\mathcal{E}$	B	S	τ	α_0	$\alpha_{\max}$	$\Delta\alpha$
Value	1000	64	4	18	0.5	0.9	0.025



(a) Bright (b) Red

(c) Green (d) Random Convolution

Fig. 2: RLA vs. PSNR threshold for four attack properties (Bright, Red, Green, Random Convolution) before and after defense on CIFAR-100.

2) Attacker setting: We adopt the SEER attack framework [8] to evaluate the defense effectiveness of LIDS. Specifically, the attacker acts as a malicious server in the FL system and aims to extract private client data from the uploaded gradients. Following [8], the attacker selects a target property $\mathcal{P}$ from a predefined set, which includes brightness (Bright, Dark), color channels (Red, Green, Blue), edge features (Horizontal, Vertical), or a random convolution-based feature (Random Convolution (Rand_Conv)). The Rand_Conv property corresponds to the response of a normalized random 3×3 convolution filter. Given the selected $\mathcal{P}$, the attacker uses an auxiliary dataset to jointly train the global model, the gradient disaggregator, and the reconstructor. This process enables the attacker to isolate gradient contributions from samples satisfying $\mathcal{P}$ and reconstruct private data from the extracted signals.

C. Performance Evaluation

To evaluate the effectiveness of LIDS, the performances are examined with multiple attacker-chosen properties using both visual and quantitative metrics under three approaches: the standard FL (No Defense), the full proposed LIDS, and the LIDS without Augmentation (LIDS-A), an ablation variant omitting the final stochastic augmentation described in Eq. (4).

1) PSNR Threshold Selection and Augmentation Impact: To show the PSNR threshold selection and the augmentation impact, we demonstrate how RLA varies under different PSNR thresholds τ in Fig. (2a)–(2d), which show the attack properties in Bright, Red, Green, and Random Convolution, respectively. Compared to the No Defense approach, both LIDS-A and LIDS successfully reduce the attack success rate with the lower RLA. Especially, with the augmentation, LIDS outperforms the LIDS-A, showing a much stronger defense. The results in Fig. (2a)–(2d) also show that the trade-off exists between the PSNR threshold (τ) and RLA. Along with the results in Fig. 3, setting

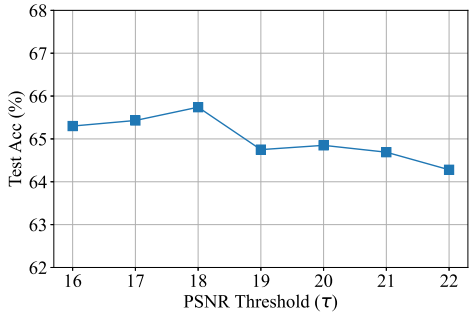


Fig. 3: CIFAR-100 test accuracy under different PSNR threshold τ.

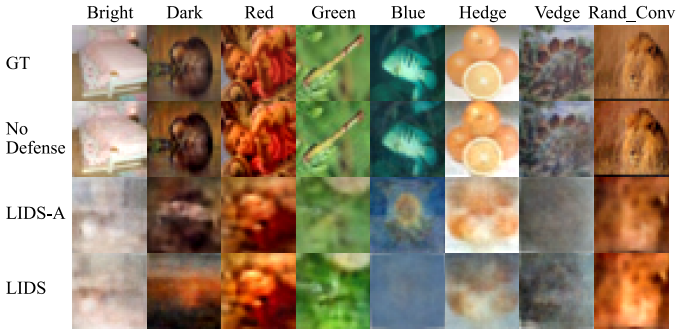


Fig. 4: Reconstructed results under different attack properties.

τ as 18 is the best option as it achieves the highest test accuracy among other threshold values. Importantly, this experiment shows that our method can synthesize effective defense samples locally, directly addressing autonomous sample synthesis.

2) Visual Comparison of Reconstruction Quality: To illustrate how LIDS disrupts visual reconstruction, Fig. 4 shows qualitative comparisons under eight different attack properties (i.e., Bright, Dark, Red, etc.) in each column, and each row displays images under one of four conditions: the original ground truth (GT), reconstructions from SEER without defense, reconstructions from LIDS-A, and reconstructions from LIDS. A clear progression in defense effectiveness is demonstrated in the results, where the reconstructed images are nearly indistinguishable from the originals, preserving both fine-grained details and semantic content under the SEER attack. With LIDS-A, the reconstructed images appear less faithful, showing reduced clarity while retaining some recognizable features, and the full LIDS leads to severe degradation, resulting in reconstructions that are dominated by noise and artifacts, lacking discernible structure and often unrecognizable even at the class level. This further validates LIDS's strength in greatly hindering the attacker's ability to extract meaningful visual information from the gradients.

3) Effectiveness Across Multiple Attack Properties: To show LIDS successfully reduces the effectiveness of the attack and upholds property-agnostic consistency, Table II lists the results with two datasets, all evaluation metrics, and attack properties with approaches of No Defense and LIDS. The preferred directions for each metric are indicated with arrows (↑ for higher is better; ↓ for lower is better). For RLA and REC, we compute the degradation as the difference between the No Defense and LIDS results, divided by the No Defense value.

TABLE II: Comparison of reconstruction results under SEER attacks with and without LIDS on CIFAR-10 and CIFAR-100 across different properties.

	No Defense Against SEER Attacks					Defense with LIDS Against SEER Attacks				
	RLA (%) ↓	REC (%) ↓	PSNR-All ↓	SSIM ↓	LPIPS ↑	RLA (%) ↓	REC (%) ↓	PSNR-All ↓	SSIM ↓	LPIPS ↑
Property	**CIFAR-10**									
Bright	93.2	94.8	27.6 ± 4.5	0.7813 ± 0.1005	0.0757 ± 0.0874	45.9	8.8	15.8 ± 2.4	0.5249 ± 0.0609	0.3418 ± 0.0877
Dark	77.8	83.4	26.7 ± 6.6	0.7650 ± 0.1918	0.1438 ± 0.1946	39.2	48.4	19.4 ± 4.9	0.5563 ± 0.1244	0.3488 ± 0.1443
Red	82.8	86.5	27.2 ± 5.8	0.7791 ± 0.1615	0.1473 ± 0.2033	29.4	14.5	17.2 ± 1.8	0.5205 ± 0.0664	0.4457 ± 0.1021
Green	91.6	92.2	25.7 ± 4.0	0.7881 ± 0.0850	0.1033 ± 0.0824	53.3	21.6	17.7 ± 1.8	0.5174 ± 0.0471	0.3807 ± 0.0613
Blue	93.9	95.4	28.9 ± 4.1	0.6986 ± 0.0892	0.0902 ± 0.0612	62.7	24.3	17.9 ± 2.3	0.5111 ± 0.0704	0.3123 ± 0.0662
Hedge	73.9	77.1	23.6 ± 5.5	0.7371 ± 0.1561	0.1901 ± 0.173	31.2	6.1	16.2 ± 2.0	0.5327 ± 0.0660	0.3914 ± 0.0807
Vedge	75.4	78.6	23.2 ± 4.9	0.7418 ± 0.1175	0.1891 ± 0.1246	33.7	16.5	17.0 ± 2.4	0.5398 ± 0.0801	0.3814 ± 0.0869
Rand_Conv	90.5	94.7	27.0 ± 4.1	0.7733 ± 0.0809	0.1181 ± 0.0766	45.6	19.3	17.4 ± 2.0	0.5177 ± 0.0432	0.3985 ± 0.0596
Property	**CIFAR-100**									
Bright	85.2	97.1	29.9 ± 4.2	0.7727 ± 0.1000	0.0740 ± 0.0598	37.8	28.4	17.8 ± 3.2	0.5198 ± 0.0747	0.3215 ± 0.1015
Dark	65.2	98.1	32.1 ± 4.1	0.8248 ± 0.0801	0.0639 ± 0.0545	31.1	55.1	20.8 ± 4.9	0.5778 ± 0.1235	0.3066 ± 0.1350
Red	80.2	97.0	29.3 ± 3.8	0.8143 ± 0.0722	0.0628 ± 0.0594	29.4	3.0	16.1 ± 1.5	0.5085 ± 0.0599	0.4022 ± 0.0767
Green	61.9	93.5	28.0 ± 4.7	0.7869 ± 0.0980	0.0879 ± 0.1017	24.0	13.8	17.1 ± 2.1	0.5132 ± 0.0548	0.3618 ± 0.0776
Blue	63.5	95.1	29.0 ± 4.3	0.7627 ± 0.0800	0.0703 ± 0.0714	32.2	15.5	16.8 ± 2.8	0.5190 ± 0.0745	0.3210 ± 0.0886
Hedge	54.0	90.2	25.9 ± 4.2	0.7794 ± 0.0854	0.1399 ± 0.0880	19.2	14.0	16.8 ± 2.4	0.5463 ± 0.0715	0.3808 ± 0.0857
Vedge	47.3	85.8	24.5 ± 4.3	0.7534 ± 0.0900	0.1649 ± 0.0954	19.8	17.6	17.3 ± 2.7	0.5552 ± 0.0814	0.3673 ± 0.0913
Rand_Conv	78.5	95.4	28.7 ± 4.0	0.8085 ± 0.0757	0.0691 ± 0.0752	24.4	5.1	16.2 ± 1.7	0.5023 ± 0.0550	0.3978 ± 0.0786

TABLE III: Test Accuracy of Trained Global Model (%)

Defense case / Dataset	CIFAR10	CIFAR100
LIDS with $\tau = 18$	88.72	65.93
No Defense	91.95	73.66

For instance, in CIFAR-10 under the Bright property, RLA drops from 93.2% to 45.9%, resulting in a 50.7% degradation. Comparing the approaches of No Defense and LIDS, LIDS significantly reduces RLA across all the attack properties by 33%–64% in CIFAR-10, and 49%–68% in CIFAR-100. LIDS also remarkably lower the REC by 41%–92% in CIFAR-10, and 43%–96% in CIFAR-100. The effectiveness of LIDS is further evidenced by its impact on visual quality metrics. We observe a consistent drop in PSNR-All on both datasets, ranging from approximately 6.2 to 13.2 dB, indicating significantly poorer reconstruction quality. Structural information is effectively obfuscated, as shown by substantial decreases in SSIM (around 0.19 to 0.31 reduction). Furthermore, reconstructions appear perceptually distant, with LPIPS values increasing by a factor of 2.0 to 6.4×. These uniform degradations across various attack properties and datasets highlight the robust nature of LIDS against SEER.

D. Performance of Test Accuracy

To evaluate whether LIDS affects model utility, we analyze its impact on the test accuracy of CIFAR-10 and CIFAR-100. As shown in Table III, the final accuracy remains comparable to standard training, with only a marginal difference of a few percentage points (i.e., around 11% test accuracy degradation in CIFAR-100). These results show that LIDS effectively balances privacy preservation and model performance, providing robust defense without significantly compromising training effectiveness, in line with the goal of robust training integration.

V. Conclusion

In this paper, we present LIDS, a client-side defense mechanism to mitigate GradInv in FL. Through extensive experiments, LIDS is shown to effectively obscure gradient signals, significantly reducing the attacker's ability to reconstruct sensitive training data. By systematically analyzing the impact of different PSNR thresholds τ, we identify an optimal configuration ($\tau = 18$) that balances privacy protection with model utility. Our results demonstrate that LIDS consistently degrades the attacker's reconstruction quality under SEER, reducing RLA by up to 33%–68% and REC by up to 41%–96%, across various datasets and attack properties with only 11% test accuracy degradation.

References

[1] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Proc. NeurIPS*, 2019.

[2] H. Yin *et al.*, "See through gradients: Image batch recovery via GradInversion," in *Proc. IEEE/CVF CVPR*, 2021.

[3] A. Hatamizadeh *et al.*, "GradViT: Gradient inversion of vision transformers," in *Proc. IEEE/CVF CVPR*, 2022.

[4] X. Xu *et al.*, "CGIR: Conditional generative instance reconstruction attacks against federated learning," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, pp. 4551–4563, 2022.

[5] B. Zhao *et al.*, "iDLG: Improved deep leakage from gradients," 2020. [Online]. Available: https://arxiv.org/abs/2001.02610

[6] R. Zhang, S. Guo, J. Wang, X. Xie, and D. Tao, "A survey on gradient inversion: Attacks, defenses and future directions," in *Proc. IJCAI*, 2022.

[7] L. Fowl, J. Geiping, W. Czaja, M. Goldblum, and T. Goldstein, "Robbing the fed: Directly obtaining private data in federated learning with modified models," in *Proc. ICLR*, 2022.

[8] K. Garov, D. I. Dimitrov, N. Jovanović, and M. Vechev, "Hiding in plain sight: Disguising data stealing attacks in federated learning," in *Proc. ICLR*, 2024.

[9] F. Boenisch *et al.*, "When the curious abandon honesty: Federated learning is not private," in *Proc. IEEE EuroS&P*, 2023.

[10] S. Zhang, J. Huang, Z. Zhang, and C. Qi, "Compromise privacy in large-batch federated learning via malicious model parameters," in *Proc. ICA3PP*, 2022.

[11] Y. Wen *et al.*, "Fishing for user data in large-batch federated learning via gradient magnification," in *Proc. ICML*, 2022.

[12] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *Proc. ACM CCS*, 2022.

[13] N. V. Chawla *et al.*, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[14] A. Triastcyn and B. Faltings, "Federated generative privacy," *IEEE Intell. Syst.*, vol. 35, pp. 50–57, 2020.

[15] Y. Lu *et al.*, "Machine learning for synthetic data generation: A review," 2025. [Online]. Available: https://arxiv.org/abs/2302.04062

[16] A. Oring, Z. Yakhini, and Y. Hel-Or, "Autoencoder image interpolation by shaping the latent space," in *Proc. ICLR*, 2021.

[17] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF CVPR*, 2016.

[19] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. ICPR*, 2010.

[20] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *Trans. Mach. Learn. Res.*, 2024.