Fed-AugMix: Balancing Privacy and Utility via Data Augmentation

Haoyang Li, Wei Chen, and Xiaojin Zhang

***Abstract*—Gradient leakage attacks pose a significant threat to the privacy guarantees of federated learning. While distortion-based protection mechanisms are commonly employed to mitigate this issue, they often lead to notable performance degradation. Existing methods struggle to preserve model performance while ensuring privacy. To address this challenge, we propose a novel data augmentation-based framework designed to achieve a favorable privacy-utility trade-off, with the potential to enhance model performance in certain cases. Our framework incorporates the AugMix algorithm at the client level, enabling data augmentation with controllable severity. By integrating the Jensen-Shannon divergence into the loss function, we embed the distortion introduced by AugMix into the model gradients, effectively safeguarding privacy against deep leakage attacks. Moreover, the JS divergence promotes model consistency across different augmentations of the same image, enhancing both robustness and performance. Extensive experiments on benchmark datasets demonstrate the effectiveness and stability of our method in protecting privacy. Furthermore, our approach maintains, and in some cases improves, model performance, showcasing its ability to achieve a robust privacy-utility trade-off.**

***Index Terms*—Federated learning, privacy-utility tradeoffs, deep leakage attack.**

I. Introduction

With the explosive growth of data and the rising concern of privacy protection, the conventional approach of transmitting and aggregating raw data has become increasingly impractical due to its high bandwidth costs and the significant risks of privacy leakage. Federated learning (FL) [1, 2, 3, 4] emerges as a groundbreaking paradigm, enabling collaboratively model training without sharing private data. However, FL systems face significant security challenges, particularly from scenarios involving *semi-honest* adversaries who follow FL protocols yet analyze the exchanged information, such as model updates, to infer sensitive client information. Among these vulnerabilities, "gradient leakage attacks" pose a severe threat, allowing adversaries to reconstruct private training data with pixel-level accuracy. Several studies, including DLG [5], Inverting Gradients [6], Improved DLG [7], and GradInversion [8], have demonstrated the feasibility of such attacks. Mitigating this vulnerability is crucial for protecting individual privacy and ensuring the broader adoption of FL in privacy-sensitive industries such as healthcare, finance, and IoT.

Early attempts aiming to thwart privacy attacks include homomorphic encryption (HE) [9], secure multi-party computation (MPC) [10, 11, 12], differential privacy (DP) [13], and gradient compression (GC) [14]. HE and MPC can protect private data without jeopardizing model performance, but they incur heavy computation and communication overhead, especially for deep neural networks. In addition, HE and MPC do not secure clients' private data after decryption for the server aggregation [15]. DP and GC protect data privacy by *distorting* (i.e., adding noise or compressing) shared model updates, which typically leads to significantly deteriorated model performance. To obtain the best of both worlds (i.e., privacy and performance), [16, 17, 18, 19] leverage fine-grained DP or regularization to mitigate the impact of noise on model performance.

Early approaches to mitigating privacy attacks include homomorphic encryption (HE) [9], secure multi-party computation (MPC) [10, 11, 12], differential privacy (DP) [13], and gradient compression (GC) [14]. While HE and MPC can safeguard private data without compromising model performance, they impose substantial computational and communication overhead, particularly for deep neural networks. Moreover, HE and MPC do not protect private data after decryption during server-side aggregation [15]. In contrast, DP and GC enhance data privacy by *distorting* shared model updates (e.g., adding noise or applying compression), but this often results in significant degradation of model performance. To achieve a better privacy-utility trade-off, recent works [16, 17, 18, 19] leverage fine-grained DP or regularization techniques to minimize the adverse effects of the performance degradation caused by noise. However, existing protection mechanisms struggle to prevent performance degradation, making it challenging to preserve test accuracy while ensuring privacy.

Geiping et al. [6] observed that data augmentation during model training increases the difficulty of localizing objects when performing gradient leakage attacks to recover original images. However, these attacks still successfully recover the original data from model updates. This is because the perturbations from augmentations are applied directly to the data, not the gradients, making them vulnerable to reverse-engineering. To address this issue, our approach incorporates the distortion generated by data augmentation directly into the model updates, achieving effects similar to DP. Additionally, this distortion ensures the stability of model training while protecting data privacy.

In this work, we propose a novel protection framework that integrates a client-level data augmentation algorithm as shown

Haoyang Li, and Xiaojin Zhang are with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Laboratory, Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: u202115372@hust.edu.cn; xiaojinzhang@hust.edu.cn)

Wei Chen is with the the School of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: lemuria_chen@hust.edu.cn)

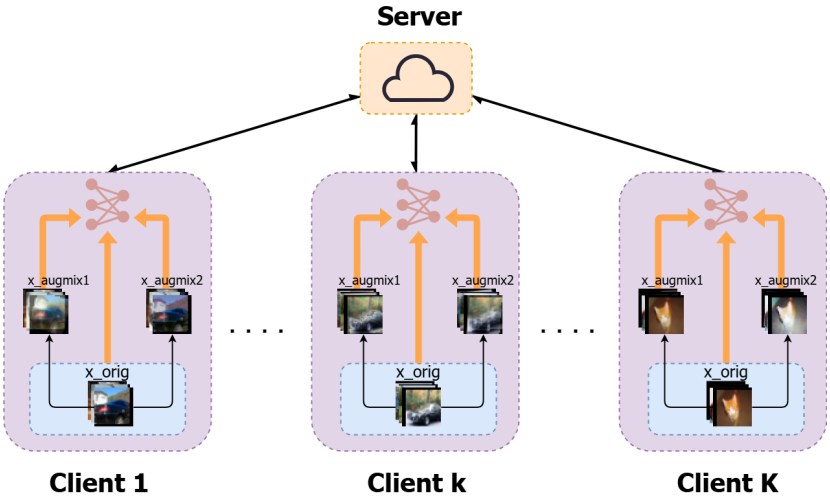

Fig. 1: An illustration of training process of Fed-AugMix. Client Updata consists of two parts: (1) In data augmentation part, we generate two augmented data based on original data; (2) In model updating part, we first compute the JS divergence of two augmented data and the original data, then add the divergence to our loss, based on which we update the parameter.

in Figure 1. Additionally, we leverage the Jensen-Shannon divergence to introduce controlled distortion into model updates, effectively safeguarding client privacy while maintaining or even enhancing model performance. This approach achieves a favorable balance between privacy and utility. Our contribution are three-fold:

- Firstly, we implement the AugMix algorithm (outlined in Algorithm 2) at the client level. This process constructs augmentation chains comprising multiple stochastic operations selected from a predefined set of transformations. The images produced by these chains are then mixed using the MixUp method. To further enhance diversity and consistency, the final output of the augmentation chains is combined with the original image through an additional convex combination. Enable multiple data augmentation methods appear on one image. Employing MixUp ensures that the final mixed image reflects a wide range of transformations.
- Secondly, we modify the client-level loss function to incorporate the Jensen-Shannon (JS) divergence between the original and augmented data into the loss function. This modification not only improve the model's robustness but also enhances privacy protection against gradient leakage attacks. By introducing complex noise through the JS divergence, this noise is embedded into the gradients during back-propagation. Due to the inherent complexity of AugMix, this noise becomes challenging to approximate during gradient leakage attacks, significantly hindering data reconstruction. Consequently, the integration of AugMix and JS divergence provides robust privacy preservation against such attacks.
- Lastly, we evaluate the proposed algorithm framework on diverse datasets and models to validate its effectiveness. Experimental results reveal that the reconstructed images under our framework's protection are unrecognizable. This demonstrates the robustness of our approach in safeguarding data privacy. Furthermore, our framework consistently achieves a favorable privacy-utility trade-off, and in some cases, it even enhances model performance, highlighting its dual benefits in privacy preservation and model optimization.

II. Related Work

Relevant prior works consist of 3 parts: study of various data augmentation strategies, gradient attack models, and privacy-utility trade-off in FL.

Data Augmentation. Data augmentation is a crucial technique for enhancing model generalization performance. Common methods for image data include random flipping and cropping, which are widely used in practice [20]. Mixup offers a distinct approach by combining information from two images through an elementwise convex combination rather than region replacement, which has proven effective for improving model robustness [21, 22]. An adaptive mixing policy can further refine Mixup's effectiveness by mitigating manifold intrusion issues [23]. In addition to these manually designed techniques, learned augmentation methods like AutoAugment [24] optimize a sequence of operations—such as translation, rotation, and shearing—by fine-tuning the probabilities and magnitudes of each operation, ultimately enhancing performance on downstream tasks. In this paper, we implement AugMix [25], a method that enhances model robustness and accuracy on standard benchmark datasets. AugMix achieves this by combining stochastic, diverse augmentations with a Jensen-Shannon Divergence consistency loss and a novel formulation to mix multiple augmented images.

Furthermore, De Luca et al. [26] demonstrate that appropriate data augmentation can mitigate data heterogeneity in FL settings, leading to improved accuracy on unseen clients. Similarly, FedM [27], a data augmentation method based on MixUp-style training, enhances FL performance by enabling data augmentation without requiring the exchange of raw local data among participants.

Gradient Attack Model. Despite the privacy-preserving nature of FL, research has shown that shared model updates can still inadvertently leak sensitive information about participants' private data. Zhu et al. [5] introduced the Deep Leakage from Gradients (DLG) attack, which reconstructs training data by solving an optimization problem on the gradients. Building on this, Geiping et al. [6] developed the Inverting Gradients (InvGrad) attack, which improves reconstruction quality by utilizing a cosine similarity-based approach. Yin et al. [8] later proposed the Recursive Gradient Inversion (RGI) attack, which refines data reconstruction by iteratively inverting gradients across multiple rounds.

In addition to these gradient-based attacks, federated learning systems are vulnerable to other privacy attacks. Notable examples include membership inference attacks, which determine if a specific data point was part of the training set [28]; property inference attacks, which extract general properties of the training data [29]; and model inversion attacks, which aim to approximate sensitive training data directly from model outputs [30]. These findings underscore the importance of developing robust defenses against privacy risks in federated learning.

Privacy-Utility Trade-off. Earlier works [31, 32, 33] explored the rate-distortion-equivocation region, quantifying utility by accuracy and privacy by entropy for large data samples. Wang et al. [34] assessed privacy leakage through identifiability, differential privacy, and mutual-information frameworks within a unified privacy-distortion model, while Wang et al. [35] proposed a χ^2-based information framework for balancing utility and privacy.

In federated learning, privacy-utility trade-offs are often framed as constrained optimization problems, minimizing utility loss under privacy constraints [36]. Pittaluga et al. [37] used adversarial optimization to train a privacy-preserving encoder within a deep neural network. FL-APB [38] combines adversarial training with adaptive privacy protection, dynamically balancing privacy and performance. Zhang et al. [39] propose an algorithmic framework using projected gradient descent to optimize a hyperparameter for near-optimal utility while respecting privacy constraints.

III. Preliminaries

In this section, we provide a notation table, formally define the general FL optimization problem, and present the model update framework underpinning our methodology.

The target of FL is to obtain a global model that is collectively trained by clients. It can be formulated as follows:

$$w^* = \arg\min_w \sum_{k=1}^{K} \frac{n^{(k)}}{n} \mathcal{L}^{(k)}(w) \tag{1}$$

where $n^{(k)}$ denotes the size of the dataset $\mathcal{D}^{(k)}$, $n = \sum_{k=1}^{K} n^{(k)}$, and $\mathcal{L}^{(k)}(w)$ represents the loss of predictions made by the model w on dataset $\mathcal{D}^{(k)}$.

The traditional FL algorithm is FedAvg [1]. The training procedure is described as follows:

- Upon receiving the global model w_t at communication round t, each selected client k locally updates its model parameters over E local iterations, following the rule: $w_{t+1}^{(k)} \leftarrow w_t - \eta \cdot \nabla \mathcal{L}^{(k)}(w_t)$.
- After completing the local training, each client transmits its updated model parameters $w_{t+1}^{(k)}$ back to the central server.
- Upon receiving the model parameters from all sampled clients, the server aggregates them by computing $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n^{(k)}}{n} w_{t+1}^{(k)}$ and subsequently distributes the updated parameters w_{t+1} to all clients.

In this context, w_t represents the aggregated model parameters at communication round t, and η refers to the learning rate. The term E represents the number of local epochs, indicating that each client updates its local model parameters for E iterations before sending the updated parameters back to the server for aggregation.

TABLE I: Notations.

Notation	Description
K	Client number
C	Participation rate
T	Communication rounds
E	Local epochs
w	Model parameter
$\mathcal{D}^k$	Dateset for client k
x	Image
y	Label
$p(y\|x)$	The prediction of image x with model $p(w, x)$
n	Augmentation chain number
l	Augmentation chain length
s	Augmentation severity
$ch_i()$	ith augmentation chain

IV. Method

We introduce the Fed-AugMix framework in Sec. IV-A, detailing the implementation of AugMix and JS loss in Secs. IV-B and IV-C, respectively. Sec. IV-E covers our loss scaling technique, and Sec. IV-D explains how these methods balance the trade-off between performance and privacy.

A. Framework Overview

To tackle the trade-off between performance and privacy protection, as mentioned in Sec. I, we present a data augmentation framework for FL: Fed-AugMix, aimed at enhancing model performance and robustness while providing privacy protection. The learning procedure is illustrated in Figure 1, and the full training process of Fed-AugMix is shown in Algorithm 1. Specifically, we adopt the AugMix algorithm [25] on clients, as detailed in Algorithm 2.

At the server level, we aggregate the weights of selected client models. At the client level, we stochastically augment the training data two times and make some adaptations to the loss function. Initially, stochastic augmentation operations are applied, layered, and combined to generate a diverse set of augmented images. Additionally, we use the MixUp technique [40] to further process both the augmented images and the original image. Besides, we incorporate the Jensen-Shannon (JS) divergence into the loss function.

Algorithm 1 Fed-AugMix.

Input: K: client number; C: the fraction of active client in each round; T: communication rounds; E: local epochs; η: learning rate; w: model parameters; $f(w, x)$: model's output of image x; $\mathcal{L}_c$: classification loss; $\mathcal{D}$: datasets for clients, $\mathcal{D} = \{\mathcal{D}^1, \cdots, \mathcal{D}^{(K)}\}$

Server executes:

Initialize w_0
for each round $t = 1, 2, \ldots, T$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{CLIENTUPDATE}(k, w_t)$
  end for
  $w_{t+1} \leftarrow \text{GLOBALUPDATE}(w_{t+1}^1, \cdots, w_{t+1}^K)$
end for

ClientUpdate(k, w): ▷ *Run on client k*

Initialize $\mathcal{L}^{(k)}(w)$
for each local epoch i from 1 to E **do**
  for (image x_{orig} and label y) $\in \mathcal{D}^{(k)}$ **do**
    $x_{\text{augmix1}} = \text{AugMix}(x_{\text{orig}})$
    $x_{\text{augmix2}} = \text{AugMix}(x_{\text{orig}})$ ▷ $x_{augmix1} \neq x_{augmix2}$
    $p_{\text{orig}} = p(y \mid x_{\text{orig}}) = f(w, x_{\text{orig}})$
    $p_{\text{augmix1}} = p(y \mid x_{\text{augmix1}}) = f(w, x_{\text{augmix1}})$
    $p_{\text{augmix2}} = p(y \mid x_{\text{augmix2}}) = f(w, x_{\text{augmix2}})$
    $\mathcal{L}^{(k)}(w) = \mathcal{L}_c(p_{\text{orig}}, y) + \lambda D_{\text{JS}}(p_{\text{orig}}; p_{\text{augmix1}}; p_{\text{augmix2}})$
    $w \leftarrow w - \eta \nabla \mathcal{L}^{(k)}(w)$
  end for
end for
return w to server

Algorithm 2 AugMix.

Input: x_{orig}: original image; n: augmentation chain number; $\mathcal{O}$: operation set, $\mathcal{O} = \{\text{rotate}, \text{shear}, \ldots, \text{posterize}\}$; s: augmentation severity.

Fill x_{aug} with zeros
Sample mixing weights $(b_1, \ldots, b_n) \sim \text{Dirichlet}(\alpha, \ldots, \alpha)$
for $i = 1, \ldots, n$ **do**
  Randomly choose chain length l from 1 to 3.
  Sample operations $\text{op}_1, \cdots, \text{op}_l \sim \mathcal{O}$
  Compose operation chain ch_i with operations of length l, $ch_i = \text{op}_{1\ldots l} = \text{op}_l \circ \cdots \circ \text{op}_1$
  $x_{\text{aug}} \mathrel{+}= b_i \cdot ch_i(x_{\text{orig}},\ s)$ ▷ *Addition is elementwise*
end for
Sample weight $m \sim \text{Beta}(\alpha, \alpha)$
Interpolate with rule $x_{\text{augmix}} = m x_{\text{orig}} + (1 - m) x_{\text{aug}}$
return x_{augmix}

B. AugMix

In this section, we provide a comprehensive explanation of how the AugMix algorithm utilizes two techniques: data augmentation[41] and MixUp. An example of the procedure of AugMix is illustrated in Figure 2.

First, we employ augmentation chains composed of multiple stochastic operations selected from a predefined set of augmentations. Specifically, we sample operations from AutoAugment [24] to construct an operation chain. We generate n independent operation chains to produce n different augmented images, where $n = 3$ by default. For example, in the first row of Fig. 2, the first augmentation chain consists of $Rotate$, $Translate\_x$ and another $Rotate$, each with varying augmentation level. These two $Rotate$ operations can vary in rotation angle ranging from $2°$ to $-15°$, where the augmentation level $AugLevel$ (rotation angle in this case) is determined based on sample level $SampLevel$ and a predefined max value $MaxVal$ that vary in operation:

$$\text{AugLevel} = \frac{\text{SampLevel}}{10} \cdot \text{MaxVal} \ , \tag{2}$$

where $SampLevel \sim \mathcal{U}(0.1,\ s)$ and hyperparameter s denotes augmentation severity.

After the augmented images are generated through operation chains, they are combined using a convex mixing process. Instead of alpha compositing, we use elementwise convex combinations for simplicity. First, the n augmented images $ch_i(x_{\text{orig}},\ s)$ are mixed to produce a final output x_{aug} based on

$$x_{\text{aug}} = \sum_i^n b_i \cdot ch_i(x_{\text{orig}},\ s) \ , \tag{3}$$

where $(b_1, \ldots, b_n) \sim \text{Dirichlet}(\alpha, \ldots, \alpha)$. Moreover, we combine the final output of the augmentation chains x_{aug} with the original image x_{orig} through a second convex combination, known as "skip connection":

$$x_{\text{augmix}} = m x_{\text{orig}} + (1 - m) x_{\text{aug}} \ , \tag{4}$$

where $m \sim \text{Beta}(\alpha, \alpha)$. Employing MixUp ensures that the final mixed image reflects a wide range of transformations, incorporating several sources of perturbation: the choice of operations, the severity of these operations, the lengths of the augmentation chains and the mixing weights.

C. JS (Jensen-Shannon) Loss

To ensure model consistency across different augmentations of the same image, we modify the client-level loss function. Since the semantic content of an image is preserved under AugMix, the model is expected to learn the core representations while ignoring the perturbations introduced during the AugMix process. Thus, we aim for similar predicted probabilities for the original image x_{orig} and its two stochastically augmented versions x_{augmix1} and x_{augmix2}. To achieve this, we introduce the JS divergence into the loss function. Given the model's predictions of posterior probability distribution $p_{\text{orig}} = p(y \mid x_{\text{orig}})$, $p_{\text{augmix1}} = p(y \mid x_{\text{augmix1}})$ and $p_{\text{augmix2}} = p(y \mid x_{\text{augmix2}})$, the original loss function $\mathcal{L}_c$ is modified to:

$$\mathcal{L}_c(p_{\text{orig}},\ y) + \lambda\, D_{\text{JS}}(p_{\text{orig}};\ p_{\text{augmix1}};\ p_{\text{augmix2}}). \tag{5}$$

To compute the JS divergence D_{JS}, we first calculate the average distribution q as $q = (p_{\text{orig}} + p_{\text{augmix1}} + p_{\text{augmix2}})/3$.

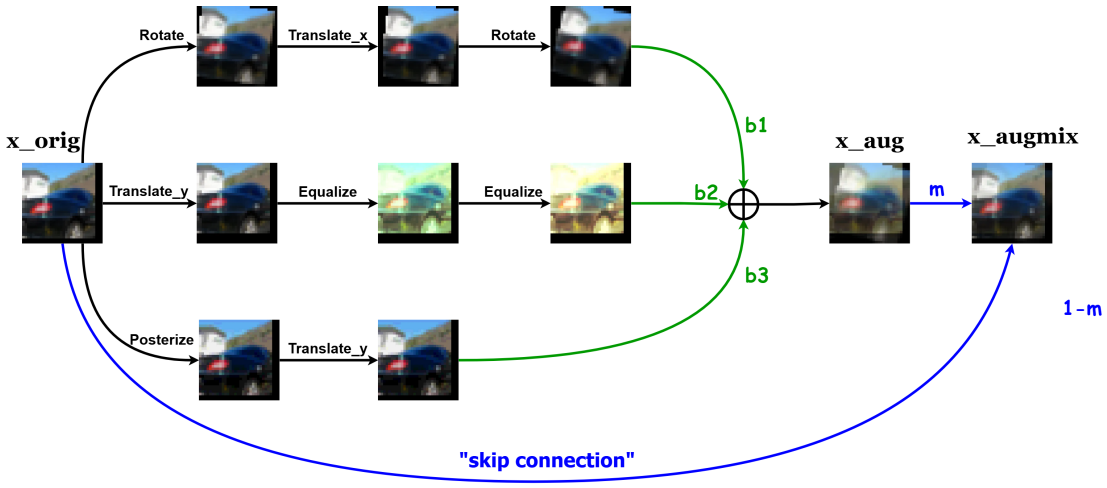

Fig. 2: An example of AugMix. First generate x_aug using three stochastic augmentation chains. Then employ "skip connection" to MixUp the augmented image and the original image.

Next, we compute the KL divergence between q and each of the three distributions: p_{orig}, p_{augmix1} and p_{augmix2}. Finally, we take the average of these KL divergences to obtain the JS divergence:

$$D_{\text{JS}}(p_{\text{orig}};\ p_{\text{augmix1}};\ p_{\text{augmix2}}) = \frac{1}{3}\Big(D_{\text{KL}}(p_{\text{orig}} \parallel q) + D_{\text{KL}}(p_{\text{augmix1}} \parallel q) + D_{\text{KL}}(p_{\text{augmix2}} \parallel q) \Big). \quad (6)$$

Unlike KL divergence, JS divergence is bounded above by the logarithm of the number of classes. We select two augmented images for sampling because computing $D_{\text{JS}}(p_{\text{orig}}; p_{\text{augmix1}})$ alone underperforms compared to our approach. Adding more distributions, such as $D_{\text{JS}}(p_{\text{orig}}; p_{\text{augmix1}}; p_{\text{augmix2}}; p_{\text{augmix3}})$, increases computational cost with only marginal performance gains. The Jensen-Shannon Consistency Loss thus encourages model stability, consistency, and resilience to input variations [42, 43, 44].

D. Balancing Performance and Privacy

The core innovation of our framework is the incorporation of JS divergence in the loss function, which evaluates the local model's ability to maintain consistent probability distributions across stochastic augmentations of the same data. we provide an intuitive explanation of our framework through two key questions: **Q1:** How does Fed-AugMix improve model's performance? **Q2:** How does Fed-AugMix offer privacy protection against gradient leakage attack (GLA), such as InvGrad attack?

To enhance model performance, AugMix introduces stochastic data corruption through random augmentation operations, varying severities, chain lengths, and mixing weights. Rather than training directly on augmented data, we incorporate JS divergence to measure prediction disparities between the original and two augmented versions of the same data. This approach enforces consistency in local model predictions across augmentations, encouraging models to learn effective representations while disregarding perturbations from AugMix. Consequently, this improves model generalization in FL scenarios and enhances robustness against various corruptions.

To explain Fed-AugMix's privacy protection mechanism, it's essential to first understand GLA. GLA works by exploiting the gradient-sharing process in collaborative training. While normal participants compute gradients using their private training data, a malicious attacker initialize "dummy inputs" (random noise) and corresponding labels, iteratively updating them to minimize the gradient distance to the shared gradients. By optimizing these dummy gradients to fit the leaked gradient, the attacker can effectively reconstruct the input data.

A effective method to mitigate privacy leakage risk posed by GLA is Differential Privacy (DP), which adds random noise on gradients. This noise reduces the precision of gradient information, making it difficult for attackers to reconstruct original data, while simultaneously limiting the influence of individual data points on the gradient, thereby preserving privacy. Our framework, Fed-AugMix, shares certain similarities with DP.

Since GLA can reconstruct original data from gradients, simply data augmentation alone remains vulnerable to such recovery. Although AugMix introduces perturbations, an attacker could potentially reverse-engineer these modifications to reconstruct the augmented data [6]. As standard augmentations like AugMix do not inherently render data indistinguishable, augmented data remains at risk of recovery, potentially leading to privacy leakage of the original data.

In addition to AugMix, our framework introduces a JS loss that integrates JS divergence into the loss function, effectively enhancing privacy protection against GLA. Since AugMix generates stochastic augmentations, the JS divergence between the original and augmented data can be viewed as complex noise. Through back-propagation, this noise is incorporated into each gradient. Due to the complexity of AugMix, this noise is difficult to aprroximate during GLA, making data reconstruction challenging. Thus, the combination of AugMix and JS divergence offers robust privacy preservation against attacks.

E. Scaling the Loss

In the InvGrad attack experiment discussed in Sec. V-B, we find that our framework does not fully protect privacy, as a small portion of the image can still be recovered, revealing information about the original data. To identify factors limiting privacy protection, we analyze the algorithm and observe that, when attacking an untrained model, the classification loss $\mathcal{L}_c$ is approximately equal to the JS divergence scaled by 10^5. Given the negligible magnitude of the JS divergence, the noise added to gradients through back-propagation remains minimal, making data recovery feasible.

As we increase λ (the coefficient of JS divergence) to a large value, such as 10^5, the model's performance degrades significantly. We attribute this to the modified loss function, which causes the model to prioritize optimizing the JS divergence over the $\mathcal{L}_c$ during training. Specifically, during model training, while the $\mathcal{L}_c$ decreases, λD_{JS} remains high, leading subsequent training to primarily focus on JS divergence. Consequently, this disrupts convergence and hampers model performance.

To address this dilemma, we propose a phased approach to adjusting λ. Here, $\mathcal{L}_c$ denotes the classification loss, D_{JS} represents the Jensen-Shannon divergence, and $Scale$ is a hyperparameter controlling the sensitivity of the condition. When $\mathcal{L}_c$ surpasses the scaled divergence, λ is set to a large constant value $LargeVal$; otherwise, λ retains its previous value. In the early training stages, when JS divergence is low, we set λ to a higher value $LargeVal$ to ensure sufficient noise is added to the gradients for privacy protection. In later stages, as lower loss provides less gradients information for attackers to exploit, we reduce λ to prioritize optimizing the $\mathcal{L}_c$, thereby maintaining model performance. This approach effectively balances performance with privacy requirements.

V. Experiments

In this section, we evaluate the Fed-AugMix framework's ability to balance performance and privacy. First, we analyze the privacy protection achieved under varying augmentation severities and compare it with the baseline of no protection. Next, we present results on three datasets, demonstrating that Fed-AugMix delivers a superior privacy-utility trade-off, effectively safeguarding privacy while preserving model performance. Finally, we show empirical evidence of improved test accuracy and faster convergence with our framework in certain scenarios.

A. Experimental Setup

Dataset Partitioning. We conducted experiments on three datasets: MNIST [45], CIFAR10 [46] and CIFAR100 [46]. We set the number of clients to $K = 100$, with 10% of clients ($C = 0.1$) participating in each communication round, meaning 10 clients were active per round. To simulate a non-IID data distribution, we used Dirichlet sampling with a parameter of $\alpha = 0.1$ to partition the data across clients. Each client's data was further divided into training and test sets with a test split ratio of 0.25.

TABLE II: The average of defense effect metrics on MNIST, CIFAR10 and CIFAR100 (measured by MSE, SSIM and PSNR). Here, an upward arrow (↑) indicates that higher values correspond to better protection, while a downward arrow (↓) signifies that lower values are preferred for improved defense outcomes.

MNIST

Stage	Protection	MSE (↑)	SSIM (↓)	PSNR (↓)
UNTRAINED	none	1.387	10.23%	9.204
	$s = 2$	2.547	2.54%	6.519
	$s = 4$	2.749	2.64%	6.186
	$s = 6$	2.811	2.08%	6.126
	$s = 8$	2.838	2.71%	6.142
	$s = 10$	**2.909**	**1.94%**	**5.992**
CONVERGENT	none	2.092	6.65%	7.731
	$s = 2$	2.260	4.02%	7.237
	$s = 4$	2.292	3.36%	7.142
	$s = 6$	2.377	**2.56%**	6.919
	$s = 8$	2.382	2.71%	6.965
	$s = 10$	**2.414**	3.36%	**6.883**

CIFAR10

Stage	Protection	MSE (↑)	SSIM (↓)	PSNR (↓)
UNTRAINED	none	3.445	1.08%	8.744
	$s = 2$	3.725	0.73%	8.339
	$s = 4$	3.786	0.66%	8.266
	$s = 6$	3.828	0.68%	8.222
	$s = 8$	3.835	0.62%	**8.208**
	$s = 10$	**3.836**	**0.59%**	8.212
CONVERGENT	none	3.629	1.94%	8.584
	$s = 2$	3.751	1.56%	8.387
	$s = 4$	3.707	1.68%	8.481
	$s = 6$	3.760	1.42%	8.411
	$s = 8$	3.837	1.33%	8.328
	$s = 10$	4.323	1.01%	7.785

CIFAR100

Stage	Protection	MSE (↑)	SSIM (↓)	PSNR (↓)
UNTRAINED	none	3.462	0.95%	8.722
	$s = 2$	3.817	0.76%	8.272
	$s = 4$	3.857	0.75%	8.212
	$s = 6$	3.926	**0.64%**	8.141
	$s = 8$	3.968	0.72%	8.085
	$s = 10$	**3.991**	0.66%	**8.065**
CONVERGENT	none	3.096	3.64%	9.345
	$s = 2$	**3.275**	2.87%	**9.063**
	$s = 4$	3.195	2.57%	9.147
	$s = 6$	3.104	2.67%	9.267
	$s = 8$	3.131	2.18%	9.223
	$s = 10$	3.011	**2.08%**	9.393

Model Architectures. To address the varying image sizes, dataset scales, and task complexities, we employed two distinct neural network architectures. For the MNIST and FMNIST datasets, we utilized LeNet-5 [47], a 5-layer neural network suited for simpler datasets. For the CIFAR-10 and CIFAR-100 datasets, we adopted ResNet-50 [20], a deeper architecture designed for more complex classification tasks. This selection ensures that the model capacity aligns with the complexity of each dataset.

Attack Setting. We adopt the InvGrad method [6] as the attack strategy. Consistent with common attack assumptions

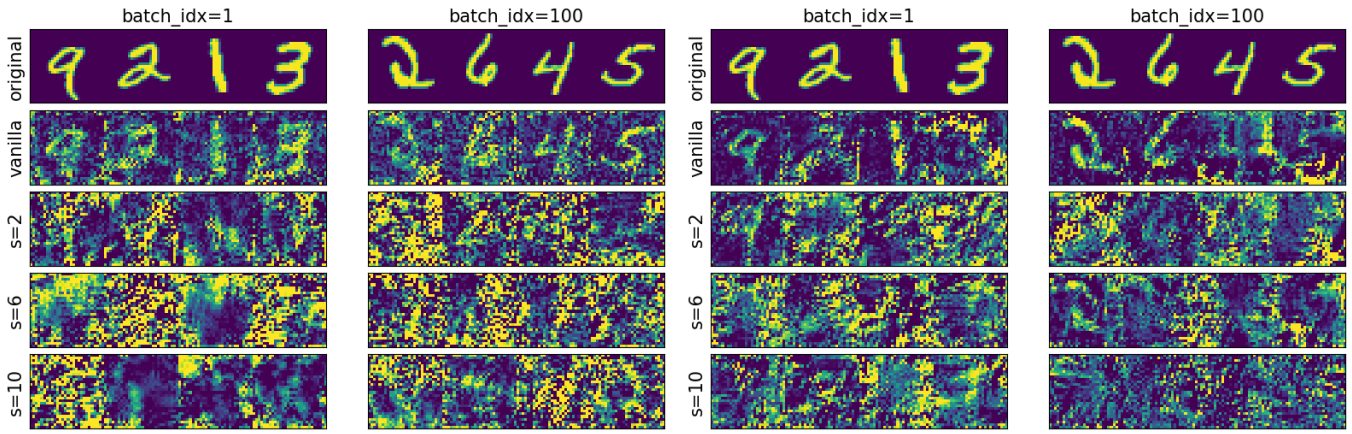

Fig. 3: Visualization of InvGrad attack results under varying privacy protection severity levels (s=0, 2, 6, 10). The second row shows reconstructions without protection, while the following rows display reconstructions with Fed-AugMix at different augmentation severities.

in federated learning, the attacker is presumed to have access to weight updates. Additionally, we assume the attacker has knowledge of the true labels to simplify the attack. Experiments are conducted on MNIST, CIFAR-10, and CIFAR-100, as these datasets are relatively easier to reconstruct. For each target batch, 2500 attack iterations are performed using the Adam optimizer with a learning rate of 0.1 and a total variation coefficient of 1×10^{-6}. Given the limitations of gradient leakage attacks in large-batch attacks [5], we use a small batch size of 4 and set the local training epoch to 5 to increase the vulnerability of the training process. For each of the 100 clients, the attack is performed on a specific batch at two distinct training stages: UNTRAINED and CONVERGENT.

Privacy Setting. We vary the augmentation severity across a range of values to explore the trade-off between performance and privacy. In loss scaling procedure, we set $Scale$ to 5×10^4, $LargeVal$ to 5×10^3, and λ to 50, which effectively mitigate privacy leakage, particularly during the early stages of training. The defense effectiveness is evaluated using metrics such as Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR).

B. Results and Analysis

Protecting Privacy. We evaluated privacy protection on the MNIST dataset under varying stages and levels of protection severity using metrics such as Mean Squared Error (MSE), Structural Similarity Index Measure (SSIM), and Peak Signal-to-Noise Ratio (PSNR). Figure 3 presents the results of our privacy protection experiments. Without protection, the original images expose a notable amount of private information when subjected to the InvGrad attack. However, integrating Fed-AugMix and Loss Scaling significantly reduces privacy leakage, rendering the reconstructed images unrecognizable.

Across different stages and levels of protection severity, as summarized in Table II, our method consistently demonstrates a high level of effectiveness. Notably, the SSIM values remain below 5%, indicating minimal similarity between the reconstructed and original images. Furthermore, as the severity of augmentations increases, the MSE also rises, reflecting an enhanced degree of privacy preservation over protection severity.

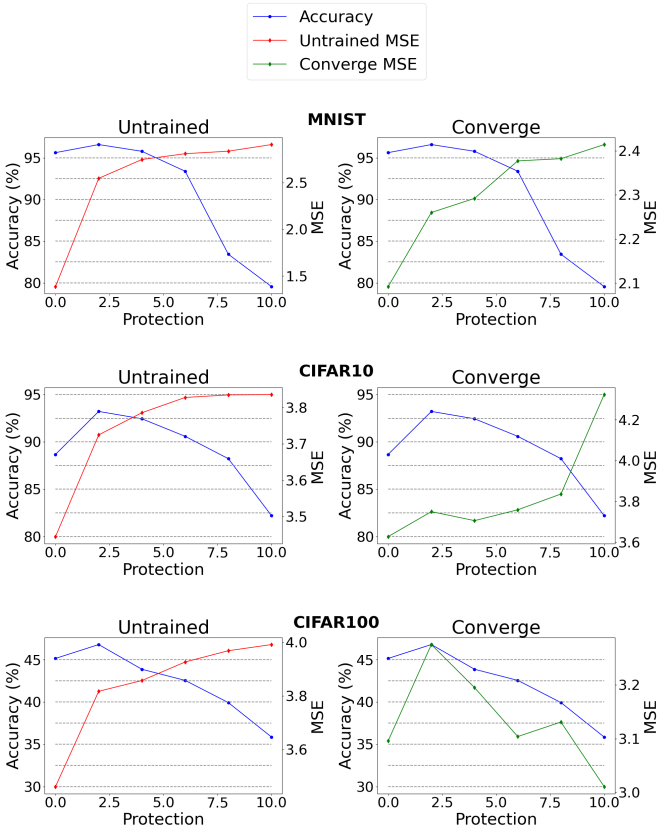

Fig. 4: Relationship between test accuracy and MSE of reconstructed images under varying protection severity. Lower severity reduces MSE while improving accuracy compared to no protection. However, as augmentation severity increases, accuracy decreases for both untrained and converged models.

Privacy-Utility Trade-off. We conduct experiments to evaluate test accuracy under varying levels of augmentation severity

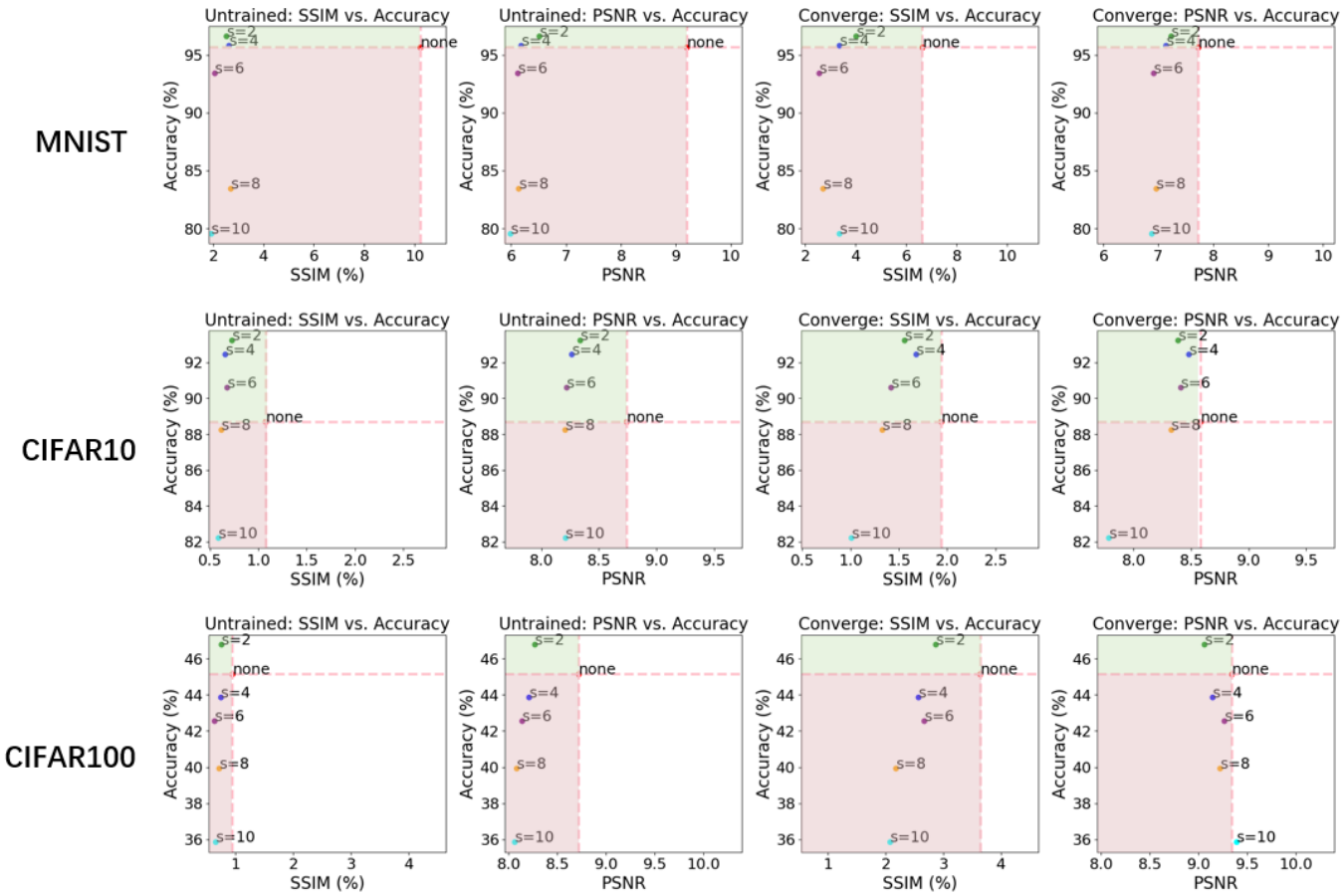

Fig. 5: An illustration of the relationship between accuracy, SSIM, and PSNR for untrained and converged models across different datasets. The green region indicates improved accuracy alongside better privacy protection, while the red region reflects enhanced privacy protection at the cost of performance degradation. Most privacy protection mechanisms of FL fall within the red region.

and analyze the privacy-utility trade-off of Fed-AugMix using metrics such as MSE, SSIM, and PSNR. As shown in Figure 4, test accuracy declines with increasing augmentation severity, while the MSE between original and reconstructed images rises. When employing a converged model for image recovery, raising the augmentation severity from 0 to 10 increases the MSE from 2.092 to 2.414. Similarly, using an untrained model, the MSE increases from 1.387 to 2.909.

The experimental results demonstrate the framework's ability to balance privacy and accuracy, highlighting that the trade-off between privacy and utility can be controlled by adjusting the augmentation severity s. Specifically, applying augmentations with lower severity initially improves model performance. Additionally, the distortions introduced to the images, which propagate back to the gradients, help protect privacy. Figure 5 illustrates the effectiveness of our framework in achieving this balance by showing SSIM and PSNR versus accuracy across different augmentation severity levels. To enhance clarity, the UNTRAINED and CONVERGED states are represented as separate lines. In the figure, points in the upper-left corner indicate a favorable trade-off between privacy and utility. Notably, augmentation severities $s = 2$, $s = 4$, and $s = 6$ are located in this region, demonstrating their effectiveness in balancing privacy and utility.

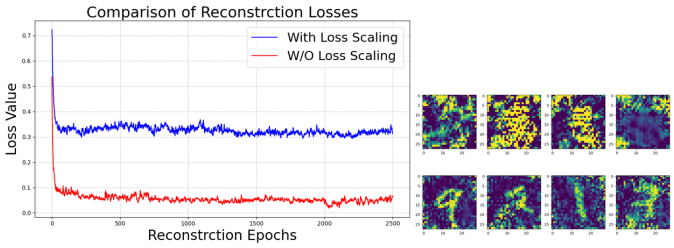

Fig. 6: An example demonstrating the effectiveness of Loss Scaling: the reconstruction loss is higher with Loss Scaling than without it, reducing the likelihood of privacy leakage.

Loss Scaling. To address privacy leakage during the early stages of training, we propose Loss Scaling, which increases the JS divergence between different augmentations of the same image. To assess its effectiveness, we conducted two InvGrad attacks on an untrained model over 2500 iterations using the same batch: one with Loss Scaling and one without. As illustrated in Figure 6, without Loss Scaling, the attacker achieves a lower reconstruction loss, nearly recovering the original images. In contrast, with Loss Scaling, the reconstructed images become

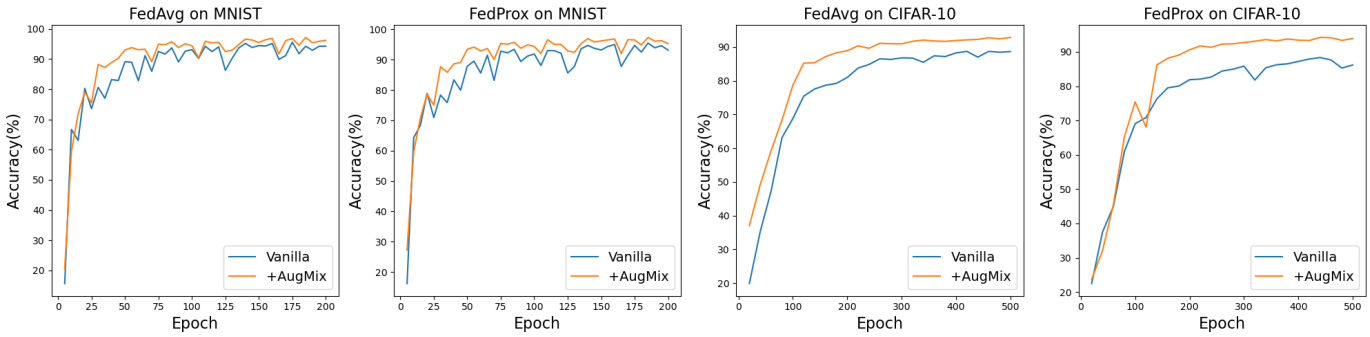

Fig. 7: These charts illustrate the test accuracy of two federated learning methods, FedAvg and FedProx, during the training process. The left charts show the test accuracy on the MNIST dataset, comparing the results with and without the use of Fed-AugMix. The right charts display the same comparison on the CIFAR-10 dataset.

indistinguishable. These results indicate that Fed-AugMix is vulnerable to privacy leakage during early training stages. However, the inclusion of Loss Scaling significantly reduces reconstruction loss, effectively mitigates privacy leakage.

Compatibility and Effectiveness. We evaluate the performance of different FL algorithms on multiple datasets, both with and without our proposed Fed-AugMix framework. Table III highlights the compatibility and effectiveness of our framework across different datasets, including MNIST, CIFAR10 and CIFAR100. As shown in Table 7, integrating Fed-AugMix with each FL method consistently improves test accuracy across all datasets. For example, when applying FedProx to CIFAR10, the vanilla FedProx achieves a test accuracy of 88.33%, while FedProx combined with Fed-AugMix improves to 94.19%, representing a 5.86% increase. These results indicate that our framework is highly compatible with a range of FL algorithms and demonstrates significant improvements in performance, making it a versatile solution for diverse applications.

VI. Conclusion and Discussion

This paper investigates the trade-off between privacy and utility in FL scenarios. To address this challenge, we propose a novel framework, Fed-AugMix, which applies the AugMix algorithm at the client level. This approach enhances model performance, robustness, and generalization while introducing perturbations to the original data that propagate to the gradients, thereby preserving client privacy. Furthermore, we incorporate Loss Scaling to ensure consistent privacy preservation throughout the entire training process.

Limitations. Data augmentation introduces only limited distortions to the images, resulting in relatively minimal noise being added to the gradients. However, differential privacy can overcome this limitation by injecting noise with predefined parameters μ and σ, allowing precise control over the level of privacy protection. Another limitation is the computational overhead of using AugMix, which typically increases the training time due to its data augmentation operations.

Future Work. Future work should focus on providing deeper interpretations and conducting rigorous theoretical analyses of integrating data augmentation into FL frameworks. Additionally, a systematic evaluation of various augmentation strategies is needed to assess their impact on both model performance and privacy. Building on these insights, future research could aim to design more controllable and effective data augmentation methods tailored to FL scenarios.

References

[1] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[3] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[4] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," 2019. [Online]. Available: https://arxiv.org/abs/1906.08935

[6] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients – how easy is it to break privacy in federated learning?" 2020. [Online]. Available: https://arxiv.org/abs/2003.14053

[7] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *ArXiv*, vol. abs/2001.02610, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:210064455

[8] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," 2021. [Online]. Available: https://arxiv.org/abs/2104.07586

[9] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[10] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, nov 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[11] G. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the 1979 AFIPS National Computer Conference*. Monval, NJ, USA: AFIPS Press, 1979, pp. 313–317.

[12] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.

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

[14] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[15] M. Lam, G.-Y. Wei, D. Brooks, V. J. Reddi, and M. Mitzenmacher, "Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 5959–5968.

[16] W. Wei, L. Liu, Y. Wut, G. Su, and A. Iyengar, "Gradient-leakage resilient federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 797–807.

[17] M. Noble, A. Bellet, and A. Dieuleveut, "Differentially private federated learning on heterogeneous data," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., vol. 151. PMLR, 28–30 Mar 2022, pp. 10 110–10 145.

[18] L. Zhu, X. Liu, Y. Li, X. Yang, S.-T. Xia, and R. Lu, "A fine-grained differentially private federated learning against leakage from gradients," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 500–11 512, 2021.

[19] X. Shen, Y. Liu, and Z. Zhang, "Performance-enhanced federated learning with differential privacy for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 079–24 094, 2022.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[21] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *ICLR*, 2017.

[22] Y. Tokozume, Y. Ushiku, and T. Harada, "Between-class learning for image classification," *CVPR*, 2018.

[23] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," in *AAAI*, 2019.

[24] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," 2018.

[25] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," 2020. [Online]. Available: https://arxiv.org/abs/1912.02781

[26] A. B. de Luca, G. Zhang, X. Chen, and Y. Yu, "Mitigating data heterogeneity in federated learning with data augmentation," 2022. [Online]. Available: https://arxiv.org/abs/2206.09979

[27] H. Zhang, Q. Hou, T. Wu, S. Cheng, and J. Liu, "Data-augmentation-based federated learning," *IEEE Internet of Things Journal*, vol. 10, no. 24, pp. 22 530–22 541, 2023.

[28] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2019, p. 739–753. [Online]. Available: http://dx.doi.org/10.1109/SP.2019.00065

[29] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 691–706.

[30] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:207229839

[31] I. S. Reed, "Information theory and privacy in data banks," in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, 1973, pp. 581–587.

[32] H. Yamamoto, "A source coding problem for sources with additional outputs to keep secret from the receiver or wiretappers (corresp.)," *IEEE Transactions on Information Theory*, vol. 29, no. 6, pp. 918–923, 1983.

[33] L. Sankar, S. R. Rajagopalan, and H. V. Poor, "Utility-privacy tradeoffs in databases: An information-theoretic approach," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 838–852, 2013.

[34] W. Wang, L. Ying, and J. Zhang, "On the relation between identifiability, differential privacy, and mutual-information privacy," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5018–5029, 2016.

[35] H. Wang and F. P. Calmon, "An estimation-theoretic view of privacy," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 886–893.

[36] X. Zhang, Y. Kang, K. Chen, L. Fan, and Q. Yang, "Trading off privacy, utility and efficiency in federated learning," *arXiv preprint arXiv:2209.00230*, 2022.

[37] F. Pittaluga, S. Koppal, and A. Chakrabarti, "Learning privacy preserving encodings through adversarial training," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 791–799.

[38] T. Liu, H. Wu, X. Sun, C. Niu, and H. Yin, "Fl-

apb: Balancing privacy protection and performance optimization for adversarial training in federated learning," *Electronics*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:273635882

[39] X. Zhang, W. Li, K. Chen, S. Xia, and Q. Yang, "Theoretically principled federated learning for balancing privacy and utility," 2023. [Online]. Available: https://arxiv.org/abs/2305.15148

[40] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017.

[41] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, p. 100258, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005622000911

[42] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3365–3373. [Online]. Available: http://papers.nips.cc/paper/5487-learning-with-pseudo-ensembles.pdf

[43] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," *CVPR*, 2016.

[44] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *NeurIPS*, 2018.

[45] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

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

[47] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

APPENDIX

We evaluated InvGrad attacks on CIFAR-10 and CIFAR-100 datasets using ConvNet, an 8-layer CNN because of the susceptibility to gradient inversion attacks.

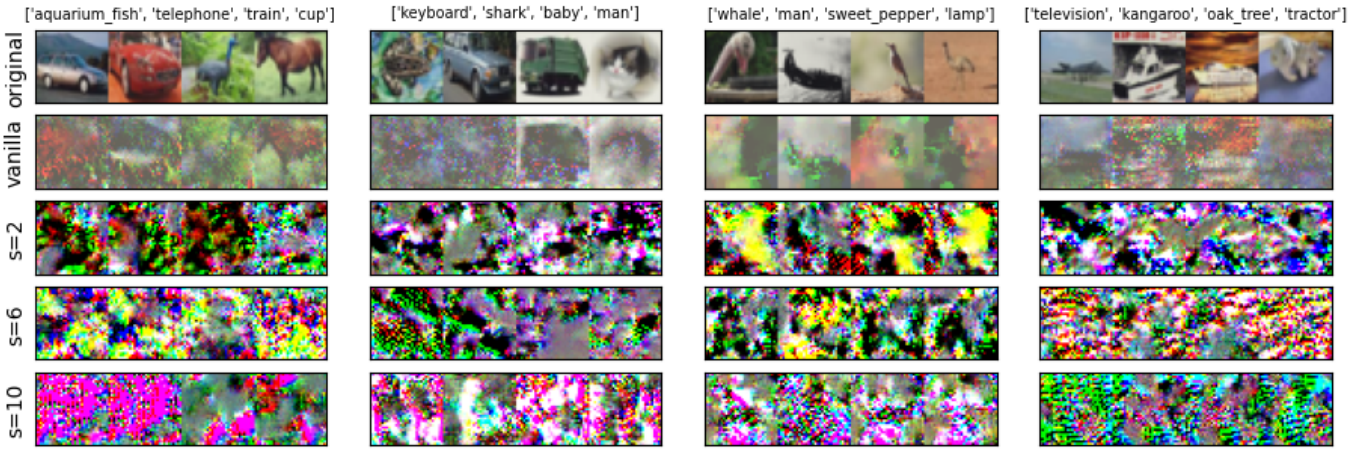

Fig. 8: CIFAR10

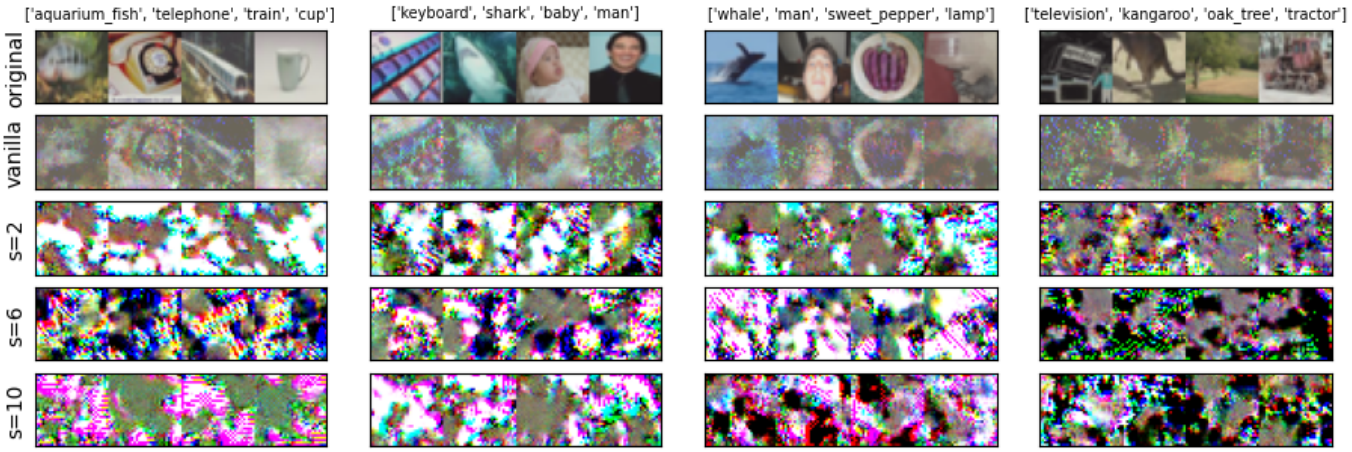

Fig. 9: CIFAR100

As Figure 8 and Figure 9 show, without the protection of Fed-AugMix, the reconstructed images reveal significant privacy leakage, as the original content can be effectively recovered. In contrast, applying Fed-AugMix with varying augmentation severities substantially mitigates this risk, effectively preventing the recovery of private image content and enhancing privacy protection.

We evaluated the test accuracy of Fed-AugMix on MNIST, CIFAR-10 and CIFAR-100 datasets. We employed lenet-5 for MNIST 10-class image classification, and ResNet50 for CIFAR10 and CIFAR100.

As Table III demonstrates, with the introduction of data augmentation and JS divergence in Fed-AugMix, the model performance increase in all 3 datasets. Our framework can employ other FL algorithms as backbone, in order to fit a specific task and function.

TABLE III: The test accuracy of different FL methods, with and without Fed-AugMix, on the MNIST, CIFAR10, and CIFAR100 datasets with $C = 10\%$. We have bolded the highest test accuracy. The numbers within parentheses represent the improvement of accuracy, and $\star$ on the cell means in that case, the model will collapse.

Dataset	MNIST		CIFAR10		CIFAR100	
Method	vanilla(%)	Fed-AugMix(%)	vanilla(%)	Fed-AugMix(%)	vanilla(%)	Fed-AugMix(%)
FedAvg	95.63	96.60	88.68	93.22	45.13	46.77
FedProx	95.88	97.34	88.33	94.19	45.24	47.83