REDUCING TRAINING-VALUE LEAKAGE IN SPLIT INFERENCE VIA LEARNED WEIGHT PERTURBATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Split inference avoids transmitting raw inputs, but the intermediate representations exposed at the split point may still reveal valuable information. In proprietary industrial, scientific, and other high-value data settings, the primary concern is not only whether individual samples can be visually reconstructed, but whether the exposed information can be recovered and reused to train another model. We refer to this risk as *training-value leakage*. To mitigate it, we propose learned client-side weight perturbations that modify selected pre-split weights while leaving the server model and deployment interface unchanged. The perturbations are optimized with reconstruction-aware feedback and selected under an explicit validation-accuracy constraint, so that reduced leakage is not obtained by simply degrading the original prediction task. We evaluate the method on CIFAR-10, CIFAR-100, and NEU-CLS using post-selection SplitNN-decoder and UnSplit reconstruction benchmarks, and measure reconstruction MSE and reconstructed-dataset trainability across three evaluator architectures. Learned perturbations reduce split-inference accuracy by only 2.3 percentage points on average, while reconstructed-dataset trainability shows an average positive drop of 23.2 points and a maximum drop of 65.7 points. These results suggest that split representations can retain intended inference utility while becoming substantially less reusable as downstream training data.

1 INTRODUCTION

Split inference enables a client to execute the early layers of a neural network locally and transmit an intermediate representation to a server for the remaining computation. This deployment pattern can reduce client-side computation and avoid sending raw inputs directly to the server, and is closely related to the broader split learning paradigm (Vepakomma et al., 2018; Poirot et al., 2019; Osia et al., 2020). However, the split representation is itself a learned function of the private input. Prior work on feature inversion and model inversion shows that learned representations can retain substantial information about their inputs (Fredrikson et al., 2015; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016; He et al., 2019), and recent attacks demonstrate that split learning protocols can be vulnerable to input reconstruction (Pasquini et al., 2021; Erdogan et al., 2022).

Training-value Leakage Problem Accurate visual reconstruction is not always the most consequential form of data leakage. In proprietary industrial, scientific, and other high-value data settings, an individual sample may contain little sensitive identity information, while the dataset as a whole can still be highly valuable. That value may arise from expensive data collection, expert annotation, rare examples, or long-term accumulation of domain-specific coverage. In such settings, a more relevant risk is whether an adversary can use the exposed intermediate representations to recover a collection of samples that is sufficiently informative to train another model. Even if the recovered samples differ visibly from the originals, they may still be reusable as training data as long as they preserve stable task-relevant structure. The resulting harm is therefore not limited to reconstructing individual samples, but extends to unauthorized reuse of the dataset as a training resource.

This risk is not fully captured by conventional reconstruction-quality metrics. Pixel-level error measures numerical similarity between a reconstruction and its source input, but does not directly measure how much training value remains in the reconstructed data. Visually blurred or distorted

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

samples may still preserve class-discriminative structure and, when aggregated into a dataset, may support downstream learning and generalization. As a result, evaluating reconstruction attacks only through mean squared error (MSE) or visual fidelity can underestimate the value exposed by the split representation. We refer to this risk as **training-value leakage**: the extent to which exposed information can be recovered and reused as training data for an effective downstream model. Rather than asking only how closely an individual sample can be reconstructed, we also ask whether the reconstructed dataset remains useful for learning. This perspective motivates reconstructed-dataset trainability as a primary measure of leakage.

From this perspective, the goal of protection also changes. We do not seek to remove all input-related information from the split representation, since the server still depends on that representation to perform the intended prediction task. The key challenge is instead to preserve the information required for the authorized task while reducing the extent to which the representation can be repurposed as a training resource. Aggressively degrading the representation may make reconstruction harder, but can also damage the information needed by the server classifier. Conversely, a representation that preserves strong task performance may still contain enough structure to support recovery of a dataset with substantial downstream training value. We therefore seek representations that remain useful for normal split inference while becoming less reusable for unintended downstream model training. This tension between intended-task utility and unauthorized data reuse is the central problem studied in this work.

Solution To address this problem, we propose **learned client-side weight perturbations**. Instead of perturbing the input or directly adding noise to the transmitted activation, we learn small perturbations to selected client-side weights before the split point, while leaving the server model and deployment interface unchanged. The perturbations are optimized using reconstruction-aware feedback and selected under an explicit validation-accuracy constraint, so that reductions in reconstructed-data utility cannot be attributed simply to breaking the original prediction task. Compared with unstructured random weight changes, the learned perturbations provide a targeted way to alter how the client produces the exposed representation. The design is also motivated by a local inverse-problem view: changing the client-side forward map can modify reconstruction-sensitive directions while leaving the downstream prediction path largely intact.

To measure whether the defense actually reduces the reuse value of recovered data, we treat **reconstructed-dataset trainability** as the primary privacy metric and use reconstruction MSE as a complementary diagnostic. After selecting a perturbation, we train post-selection reconstruction benchmarks against the defended client, reconstruct a dataset from the exposed representations, and train evaluator classifiers using only the reconstructed samples. The evaluators are then tested on real held-out data. If a model trained solely on reconstructed samples generalizes well to the real data distribution, then the reconstructions still retain substantial training value even if they are visually degraded. Conversely, a large reduction in trainability indicates that the recovered dataset has become less useful as a substitute for the original data. Separating the defense-training surrogate from the final reported reconstruction benchmarks further reduces the risk that the defense merely overfits to a particular reconstruction model.

We evaluate the proposed method on CIFAR-10, CIFAR-100, and the industrial surface-defect dataset NEU-CLS, across three split points and two reconstruction benchmarks: a learned SplitNN decoder and the optimization-based UnSplit attack. Reconstructed-dataset trainability is evaluated using ResNet20, VGG11-BN, and MobileNetV2 to test whether recovered training value transfers across evaluator architectures. Across the evaluated settings, the learned perturbations reduce the original split-inference accuracy by only 2.3 percentage points on average, while producing an average positive drop of 23.2 points in reconstructed-dataset trainability and a maximum drop of 65.7 points. These results suggest that it is possible to preserve the intended inference functionality of a split model while substantially reducing the usefulness of data recovered from its intermediate representations for downstream model training.

This paper makes three main contributions: 1. **Training-value leakage.** We study split-inference leakage through downstream data reuse, distinguishing reconstruction fidelity from the training value of recovered data and using reconstructed-dataset trainability to measure whether exposed representations can be converted into useful training data. 2. **Utility-constrained weight perturbation.** We propose a client-side defense that learns additive perturbations to selected pre-split

weights with reconstruction-aware feedback, while keeping the server model and inference interface unchanged. 3. **Post-selection reconstruction evaluation.** We evaluate the method across three datasets, three split points, two reconstruction benchmarks, and three evaluator architectures, showing that learned perturbations reduce reconstructed-data trainability while largely preserving forward inference utility.

2 Related Work

Split learning partitions computation across client and server, while federated learning and secure aggregation protect decentralized training updates (McMahan et al., 2017; Kairouz et al., 2021; Bonawitz et al., 2017). Split inference instead exposes intermediate activations (Vepakomma et al., 2018; Poirot et al., 2019; Osia et al., 2020), motivating defenses that reduce input–activation dependence or perturb transmitted representations (Vepakomma et al., 2020; Mireshghallah et al., 2020). Our method instead perturbs selected pre-split client weights under an explicit utility constraint. Prior work shows that model outputs, representations, gradients, and shared training signals can reveal private inputs (Fredrikson et al., 2015; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016; Hitaj et al., 2017; Melis et al., 2019; Nasr et al., 2019; Zhu et al., 2019; Geiping et al., 2020; Zhao et al., 2020; Yin et al., 2021; Huang et al., 2021). Split-learning attacks further exploit intermediate activations (He et al., 2019; Pasquini et al., 2021), while UnSplit considers reconstruction, model stealing, and label inference (Erdogan et al., 2022). Leakage can also extend beyond pixel recovery (Shokri et al., 2017); accordingly, we evaluate both reconstruction MSE and reconstructed-dataset trainability. Noise-based protection includes differential privacy and DP-SGD (Dwork et al., 2006; Abadi et al., 2016), selective-sharing defenses (Shokri & Shmatikov, 2015), and cryptographic approaches such as encrypted inference and secure multiparty computation (Gilad-Bachrach et al., 2016; Mohassel & Zhang, 2017). Our method provides neither DP nor cryptographic guarantees; it instead modifies selected pre-split weights, with norm-controlled random perturbations used to separate generic weight changes from reconstruction-aware learning.

3 Problem Setup and Threat Model

We formalize the exposed representation, the attacker's goal, and the defender's allowed modification. Let $(x, y) \sim \mathcal{D}$ be an image-label pair. For split point s, the client computes

$$z^{(s)} = f^{(s)}_{\theta_c}(x), \tag{1}$$

and sends $z^{(s)}$ to the server, which predicts

$$\hat{y} = \arg\max_k [g_{\theta_s}(z^{(s)})]_k. \tag{2}$$

Here $f^{(s)}_{\theta_c}$ is the client-side network before the split and g_{θ_s} is the server-side classifier. Different split points expose different representations and therefore induce different reconstruction risks.

Attacker. We consider an honest-but-curious server-side adversary that follows the split inference protocol but uses $z^{(s)}$ to reconstruct the private input:

$$\hat{x} = a_\omega(z^{(s)}), \tag{3}$$

where a_ω may be a learned decoder or an optimization-based inversion procedure. When a learned attacker is trained, it may use auxiliary images from the same domain, but not the private target images used to train or select the defense. We measure leakage by both reconstruction MSE and reconstructed-dataset trainability, because visually degraded reconstructions can still preserve training-relevant semantic information.

Defender. The defender may modify only selected client-side weights before the split point and keeps the server-side classifier fixed. With additive perturbation $\Delta\theta_c$, the defended representation is

$$z^{(s)}_\Delta = f^{(s)}_{\theta_c + \Delta\theta_c}(x). \tag{4}$$

The defense does not perturb the input image, transmitted activation, or server-side parameters. A candidate is valid only if its validation accuracy satisfies the utility threshold in Eq. 12; moreover, the surrogate decoder used for defense training is not reused as the reported adaptive attacker.

4 Inversion Instability as a Motivation

We use an inverse-problem view as motivation, not as a global reconstruction guarantee for nonlinear networks. Around an input x, the local behavior of the client map can be approximated by

$$f_{\theta_c}^{(s)}(x+\delta x) \approx f_{\theta_c}^{(s)}(x) + J_{\theta_c}^{(s)}(x)\delta x, \tag{5}$$

where $J_{\theta_c}^{(s)}(x)$ is the Jacobian with respect to the input. Thus reconstruction depends on the inverse properties of a local forward operator. In a simplified linear problem $z = Ax$, the minimum-norm estimate is $\hat{x} = A^+ z$, where A^+ is the Moore–Penrose pseudoinverse (Penrose, 1955). If $A = U\Sigma V^\top$, then $A^+ = V\Sigma^+ U^\top$, so small singular values are amplified during inversion. This sensitivity is summarized by

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}. \tag{6}$$

A large condition number indicates sensitivity to small changes in the observed representation or the forward operator, a standard property of ill-conditioned inverse problems (Hansen, 1998).

This observation motivates learned weight perturbations. Replacing θ_c with $\theta_c + \Delta\theta_c$ changes both $z^{(s)}$ and the local Jacobian,

$$J_{\theta_c}^{(s)}(x) \to J_{\theta_c+\Delta\theta_c}^{(s)}(x). \tag{7}$$

The goal is not to remove all input information, which would damage prediction, but to alter reconstruction-sensitive directions while preserving classification-relevant behavior. Learning the perturbation with reconstruction feedback provides a targeted alternative to unstructured random perturbations and motivates the utility-constrained method below.

5 Learned Client-Side Weight Perturbations

We use the notation from Section 3. Given a trained split model and split point s, the defense learns a small additive perturbation to selected client-side weights while keeping the server-side classifier fixed. The defended client therefore transmits

$$z_\Delta^{(s)} = f_{\theta_c+\Delta\theta_c}^{(s)}(x). \tag{8}$$

The perturbation is applied before the split point; it is not added to the input image, the transmitted activation, or the server-side model. Thus the deployment interface is unchanged, but the representation observed by the server is produced by a modified client-side forward map.

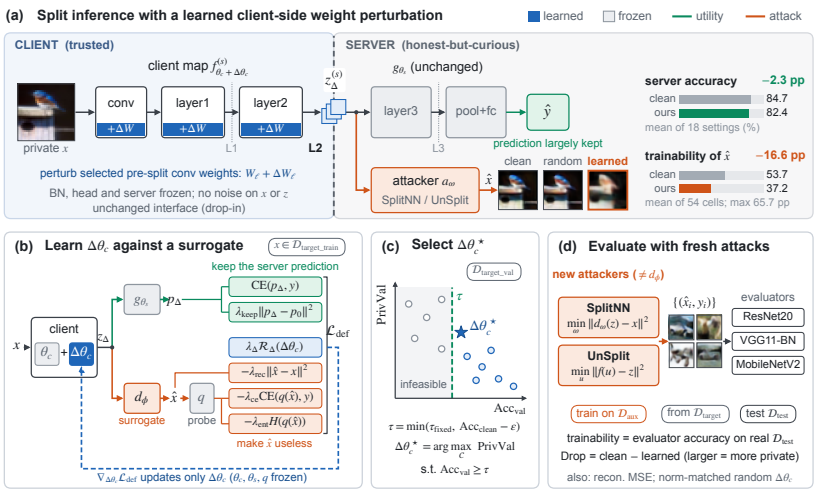


Figure 1: Overview of learned client-side weight perturbation. The client learns an additive perturbation to selected pre-split weights while keeping the server unchanged; the selected perturbation is evaluated with fresh adaptive reconstruction benchmarks through reconstructed-dataset trainability.

5.1 Perturbation Parameterization

Let $\mathcal{S}_{\mathrm{c}}(s)$ denote the client-side weight tensors selected for perturbation before split point s. For each $W_\ell \in \mathcal{S}_{\mathrm{c}}(s)$, the defended tensor is

$$\widetilde{W}_\ell = W_\ell + \Delta W_\ell. \tag{9}$$

The original victim weights are frozen and only the perturbation variables are optimized. In our ResNet experiments, $\mathcal{S}_{\mathrm{c}}(s)$ contains pre-split convolutional weights; normalization layers, classifier heads, and all server-side parameters remain unchanged. This restriction localizes the defense to the feature extraction map that determines the exposed split representation.

5.2 Surrogate-Guided Privacy Objective

Defense training uses a surrogate decoder d_ϕ to obtain a differentiable privacy signal, but this decoder is not reused as the final reported attacker. Let $\hat{x} = d_\phi(z_\Delta^{(s)})$ be the surrogate reconstruction, $p_\Delta(x)$ the defended-model logits, and $p_0(x)$ the logits with the perturbation disabled. With a frozen probe classifier q applied to reconstructions, the perturbation objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{def}} = \mathrm{CE}(p_\Delta(x), y) + \lambda_{\mathrm{keep}} \|p_\Delta(x) - p_0(x)\|_2^2 \\ + \mathcal{L}_{\mathrm{priv}}(\hat{x}, x, y) + \lambda_\Delta \mathcal{R}_\Delta,\end{aligned} \tag{10}$$

where CE is the cross-entropy loss, $\mathcal{R}_\Delta$ regularizes the perturbation magnitude, and the coefficients λ_{keep} and λ_Δ control logit preservation and perturbation regularization. The first two terms preserve prediction behavior; the last two terms provide reconstruction-aware pressure while discouraging unnecessarily large weight changes.

For the default untargeted privacy objective,

$$\begin{aligned}\mathcal{L}_{\mathrm{priv}}(\hat{x}, x, y) = -\lambda_{\mathrm{rec}} \|\hat{x} - x\|_2^2 - \lambda_{\mathrm{ce}} \mathrm{CE}(q(\hat{x}), y) \\ - \lambda_{\mathrm{ent}} H(q(\hat{x})),\end{aligned} \tag{11}$$

where $H(\cdot)$ denotes predictive entropy. Because $\mathcal{L}_{\mathrm{def}}$ is minimized, the negative signs in Eq. 11 encourage larger reconstruction error, larger probe classification loss on the reconstructed image, and higher probe uncertainty. The coefficients λ_{rec}, λ_{ce}, and λ_{ent} balance these three privacy-related signals. Thus the perturbation is trained to reduce task-relevant information in surrogate reconstructions, not merely to increase visual distortion.

5.3 Utility-Constrained Candidate Selection

Let $\mathrm{Acc}_{\mathrm{clean,val}}$ be clean validation accuracy and let τ_{fixed} be a dataset-specific minimum accuracy threshold. Given allowed accuracy drop ϵ, the effective threshold is

$$\tau = \min(\tau_{\mathrm{fixed}}, \mathrm{Acc}_{\mathrm{clean,val}} - \epsilon). \tag{12}$$

This combines an absolute dataset-specific requirement with a relative drop rule from the clean model. A candidate is feasible only if its validation accuracy is at least τ. Among feasible nonzero candidates, we select

$$\Delta\theta_{\mathrm{c}}^\star = \arg\max_{\Delta\theta_{\mathrm{c}} \in \mathcal{C}} \mathrm{PrivVal}(\Delta\theta_{\mathrm{c}}) \quad \text{s.t.} \quad \mathrm{Acc}_{\mathrm{val}}(\Delta\theta_{\mathrm{c}}) \geq \tau. \tag{13}$$

Here $\mathcal{C}$ is the set of saved perturbation candidates, $\mathrm{PrivVal}$ is the validation privacy score computed from the surrogate objective, and $\Delta\theta_{\mathrm{c}}^\star$ is the selected defense. Final privacy metrics are computed only after this selected perturbation is fixed and the post-selection reconstruction benchmarks are trained or rerun.

5.4 Random Perturbation Baselines

Random perturbations test whether privacy gains arise simply from changing client-side weights. We therefore apply norm-controlled random perturbations to the same selected tensors and subject them to the same utility feasibility rule as learned perturbations. This comparison isolates the effect of reconstruction-aware learning from the effect of weight perturbation alone.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

We organize the experiments around one common setup and three targeted questions. First, does the learned perturbation preserve forward inference accuracy? Second, does it reduce the usefulness of reconstructed images as a training set? Third, do the reconstructed samples visually reflect the quantitative privacy trends? This structure separates utility, trainability, and qualitative evidence, and avoids treating visual degradation alone as a privacy claim. The first two experiments provide the main quantitative evidence, while the visualization is used only as supporting analysis.

Models, datasets, and split points. We evaluate on CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and NEU-CLS (Song & Yan, 2013). The victim model is ResNet20 (He et al., 2016), with split points placed after `layer1`, `layer2`, and `layer3`. For each dataset and seed, samples are partitioned by original identity into $\mathcal{D}_{\text{target_train}}$, $\mathcal{D}_{\text{target_val}}$, $\mathcal{D}_{\text{aux_train}}$, $\mathcal{D}_{\text{aux_val}}$, and a held-out test partition. Target partitions are used to train and select perturbations, auxiliary partitions are used to train reconstruction benchmarks, and the held-out test partition is used only for final evaluation.

Benchmarks, defenses, and metrics. We use two reconstruction benchmarks: a decoder-based SplitNN benchmark and an optimization-based UnSplit benchmark. These benchmarks are used only to evaluate the proposed defense; the method itself is the learned client-side weight perturbation described in Section 5. Detailed benchmark protocols are provided in Appendix A.3. The primary defended condition is the selected learned perturbation; norm-controlled random perturbations are reported as auxiliary baselines in the appendix. Forward utility is measured by defended victim accuracy on the original held-out test set. Privacy is measured by reconstruction MSE and reconstructed-dataset trainability: evaluator classifiers are trained on reconstructed images and tested on real held-out test images. We use ResNet20, VGG11-BN (Simonyan & Zisserman, 2015), and MobileNetV2 (Sandler et al., 2018) as evaluator models. Unless otherwise specified, all reported values are aggregated over seeds 2026, 2027, and 2028 under the same dataset partitions, split points, and reconstruction budgets. For the clean and learned trainability entries in Table 3, the median seed standard deviation is 2.6 percentage points and the mean is 5.7 points; the largest variances occur in shallow UnSplit settings with unstable evaluator training. This design keeps the victim model, reconstruction budget, and evaluator protocol fixed when comparing clean and defended representations.

Reading the result tables. The result tables use two complementary notions of drop. In the forward utility table, smaller Drop is better because it indicates less degradation of the original prediction task. In the reconstructed-dataset trainability table, larger Drop is better for privacy because it indicates that reconstructed data becomes less useful for training downstream classifiers.

Table 1: Experimental matrix. All completed experiments use a ResNet20 victim and evaluate learned client-side weight perturbations across three datasets, three split points, two reconstruction benchmarks, and three downstream evaluator architectures. L1–L3 denote splits after `layer1`–`layer3`.

Dataset	Victim	Benchmarks	Evaluators
CIFAR-10	ResNet20	SplitNN, UnSplit	ResNet20, VGG11-BN, MobileNetV2
CIFAR-100	ResNet20	SplitNN, UnSplit	ResNet20, VGG11-BN, MobileNetV2
NEU-CLS	ResNet20	SplitNN, UnSplit	ResNet20, VGG11-BN, MobileNetV2

6.2 FORWARD INFERENCE EXPERIMENT

Purpose and protocol. This experiment evaluates whether the learned perturbation preserves the authorized split-inference function. This is necessary because a defense could trivially reduce reconstruction quality by destroying the information needed by the server classifier. Such a solution would not be useful in deployment. We therefore measure victim accuracy on the original held-out test set before and after applying the selected learned perturbation. The server-side classifier is kept fixed, and the inference interface is unchanged; the only difference is that the client uses the perturbed pre-split weights selected by the utility constraint.

Analysis. Table 2 shows that forward utility is largely preserved after applying the learned perturbation. This result is important because a split-inference defense is useful only if the defended client remains compatible with the original server classifier. NEU-CLS is the most stable dataset, with drops no larger than 1.2 percentage points. CIFAR-10 also remains stable in most settings, while CIFAR-100 is more sensitive, especially under the deeper UnSplit-evaluated setting. This sensitivity is expected because CIFAR-100 requires finer class discrimination and leaves less tolerance for perturbing the client-side feature map.

The split-dependent pattern indicates a realistic privacy–utility tradeoff. At shallow splits, later server layers can often absorb moderate changes in the client representation; at deeper splits, the representation is more task-specialized, so perturbations can affect classification more directly. Overall, Table 2 establishes the feasibility condition for the privacy results: learned perturbations modify the exposed representation while keeping the original split-inference path usable.

Table 2: Forward utility is largely preserved on the original held-out test set. Base is the unperturbed ResNet20 victim accuracy, Def. is the defended victim accuracy after applying the selected learned perturbation, and Drop is the accuracy decrease in percentage points; smaller Drop is better.

Dataset	Benchmark	Split	Base (%)	Def. (%)	Drop (pp)
CIFAR-10	SplitNN	L1	90.7	89.8	0.9
CIFAR-10	SplitNN	L2	90.7	88.0	2.7
CIFAR-10	SplitNN	L3	90.7	90.4	0.3
CIFAR-10	UnSplit	L1	90.7	88.4	2.3
CIFAR-10	UnSplit	L2	90.7	85.1	5.6
CIFAR-10	UnSplit	L3	90.7	90.4	0.3
CIFAR-100	SplitNN	L1	65.2	64.4	0.7
CIFAR-100	SplitNN	L2	65.2	61.2	3.9
CIFAR-100	SplitNN	L3	65.2	63.3	1.9
CIFAR-100	UnSplit	L1	65.2	59.4	5.8
CIFAR-100	UnSplit	L2	65.2	61.2	3.9
CIFAR-100	UnSplit	L3	65.2	56.8	8.4
NEU-CLS	SplitNN	L1	98.2	98.1	0.2
NEU-CLS	SplitNN	L2	98.2	97.1	1.1
NEU-CLS	SplitNN	L3	98.2	97.0	1.2
NEU-CLS	UnSplit	L1	98.2	98.2	0.0
NEU-CLS	UnSplit	L2	98.2	97.1	1.1
NEU-CLS	UnSplit	L3	98.2	97.0	1.2

6.3 Reconstructed-Dataset Trainability Experiment

Purpose and protocol. This experiment provides the main privacy evaluation. Reconstruction MSE and visual quality describe how close individual reconstructed images are to their sources, but they do not directly measure whether the recovered data can be reused to train another model. We therefore evaluate reconstructed-dataset trainability. For each defended representation, a post-selection reconstruction benchmark is applied to generate a reconstructed training set. The reconstructed samples inherit labels from their source images, evaluator classifiers are trained only on these reconstructed samples, and the trained evaluators are tested on real held-out images. A high evaluator accuracy means that the reconstructed data still retains training value; a large drop from Clean to Learned means that the learned perturbation makes the recovered dataset less useful as a substitute for the original private data.

Analysis. Table 3 shows that learned perturbations substantially reduce reconstructed-dataset trainability in many settings while preserving the original victim accuracy reported in Table 2. This separation is important: the defended representation remains useful for the authorized server classifier, but the data recovered from it becomes less useful for training new downstream models. In the SplitNN block, CIFAR-10 and CIFAR-100 exhibit the clearest reductions at L2 and L3 across all three evaluator architectures. This indicates that the effect is not tied to one particular evaluator model; instead, the reconstructed samples lose training-relevant structure that would otherwise transfer across ResNet20, VGG11-BN, and MobileNetV2.

Table 3: Main privacy result: learned perturbations reduce reconstructed-dataset trainability in many split settings. All entries are mean percentages over three seeds, and Drop is Clean minus Learned in percentage points. Across the Clean and Learned trainability entries, the median seed standard deviation is 2.6 percentage points and the mean is 5.7 points; larger variability is concentrated in shallow UnSplit settings. Larger Drop is better for privacy because it indicates that the reconstructed dataset is less useful for training downstream classifiers. Light-blue bold entries mark privacy-positive drops larger than 4 percentage points; random-noise baselines are reported in the appendix.

Benchmark	Dataset	Split	ResNet20			VGG11-BN			MobileNetV2		
			Clean	Learned	Drop	Clean	Learned	Drop	Clean	Learned	Drop
SplitNN	CIFAR-10	L1	89.5	88.3	1.2	88.9	88.4	0.5	80.1	80.7	-0.6
SplitNN	CIFAR-10	L2	84.0	37.1	**46.9**	86.2	56.4	**29.8**	77.9	42.6	**35.4**
SplitNN	CIFAR-10	L3	20.6	16.4	**4.2**	28.4	18.4	**10.0**	26.5	16.9	**9.7**
SplitNN	CIFAR-100	L1	63.7	61.9	1.8	62.5	61.8	0.7	49.3	49.8	-0.5
SplitNN	CIFAR-100	L2	51.5	15.9	**35.5**	57.7	33.2	**24.6**	47.0	21.3	**25.7**
SplitNN	CIFAR-100	L3	11.1	3.7	**7.4**	18.5	9.5	**9.0**	14.4	7.0	**7.3**
SplitNN	NEU-CLS	L1	96.8	95.4	1.4	71.1	58.0	**13.1**	97.5	97.8	-0.3
SplitNN	NEU-CLS	L2	94.2	89.0	**5.2**	79.5	77.2	2.3	98.1	95.1	3.0
SplitNN	NEU-CLS	L3	74.4	58.3	**16.1**	64.9	30.5	**34.4**	89.6	63.0	**26.7**
UnSplit	CIFAR-10	L1	87.5	73.8	**13.8**	61.4	45.5	**15.8**	77.4	56.3	**21.1**
UnSplit	CIFAR-10	L2	78.9	39.5	**39.4**	10.0	10.0	0.0	58.9	25.2	**33.7**
UnSplit	CIFAR-10	L3	19.9	14.8	**5.1**	10.0	10.0	0.0	18.9	12.9	**6.0**
UnSplit	CIFAR-100	L1	61.3	43.6	**17.7**	58.6	24.4	**34.2**	46.6	27.0	**19.6**
UnSplit	CIFAR-100	L2	51.0	28.2	**22.8**	15.6	1.0	**14.6**	34.9	18.2	**16.7**
UnSplit	CIFAR-100	L3	6.2	3.6	2.6	1.0	1.0	0.0	5.3	2.7	2.5
UnSplit	NEU-CLS	L1	86.9	21.2	**65.7**	51.3	21.0	**30.3**	91.9	28.1	**63.9**
UnSplit	NEU-CLS	L2	61.0	21.4	**39.6**	35.4	18.2	**17.1**	75.1	24.8	**50.3**
UnSplit	NEU-CLS	L3	19.2	20.8	-1.7	26.8	17.9	**8.9**	26.0	21.9	**4.1**

The UnSplit block provides a stronger post-selection benchmark because it does not reuse the surrogate decoder used during defense training. The learned perturbation still reduces trainability in many UnSplit settings, especially on CIFAR-10 at L1–L2, CIFAR-100 at L1–L2, and NEU-CLS at L1–L2. This supports the claim that the defense is not merely overfitting to the training-time surrogate decoder. Instead, perturbing the client-side feature map changes the information available to different reconstruction procedures.

The weaker or inconsistent cases are also interpretable. At L1, the exposed features can still contain strong low-level image structure, so reconstruction benchmarks may recover data that remains trainable unless the perturbation is large enough to harm utility. At some L3 settings, clean trainability is already low, leaving less room for additional reduction. NEU-CLS is heterogeneous because industrial defect images contain texture-like cues that can remain partially informative even when visual appearance changes. Taken together, the main conclusion is that learned weight perturbations most consistently reduce training-value leakage at split points where the representation is sufficiently semantic to be useful for the task but still vulnerable to reconstruction-based data reuse.

6.4 Reconstruction Sample Visualization

Purpose and protocol. This experiment provides qualitative support for the quantitative results. The goal is not to judge privacy solely by visual appearance, but to inspect how the learned perturbation changes the character of the reconstructions. Each example compares the original image, the clean reconstruction, a norm-controlled random perturbation control, and the learned perturbation. The random control helps distinguish the effect of simply changing client weights from the effect of learning a reconstruction-aware perturbation.

Analysis. Figure 2 shows that learned perturbations often remove more reliable class-specific structure than clean reconstructions or norm-controlled random perturbations. On CIFAR images, object shape, color layout, and discriminative parts become less stable. On NEU-CLS, the learned perturbation alters local defect textures, which is important because surface classification can depend on

subtle texture-level cues. These examples are consistent with the trainability reductions in Table 3: the reconstructions may remain coarsely recognizable, but they contain less stable information for training downstream classifiers.

Table 4 provides a complementary pixel-level diagnostic, but MSE is not treated as the sole privacy metric. A reconstruction may have moderate pixel error while still preserving useful class-discriminative structure, or remain partially recognizable while losing training value. Thus the qualitative panel and MSE table support the main trainability-based evaluation rather than replacing it.

Table 4: Reconstruction MSE under learned perturbations. MSE is computed between the original and reconstructed images and is not evaluator-model specific. It is reported as an image-level diagnostic to be interpreted together with the trainability results, rather than as the sole privacy metric.

Benchmark	Dataset	L1	L2	L3
SplitNN	CIFAR-10	0.0012	0.0168	0.0470
SplitNN	CIFAR-100	0.0006	0.0096	0.0369
SplitNN	NEU-CLS	0.0015	0.0020	0.0067
UnSplit	CIFAR-10	0.0469	0.0956	0.1305
UnSplit	CIFAR-100	0.0372	0.0693	0.1255
UnSplit	NEU-CLS	0.0924	0.0823	0.0901

MSE as a diagnostic. Table 4 reports reconstruction MSE as a pixel-level diagnostic. MSE complements trainability, but it does not fully determine privacy leakage: a reconstruction may remain partially recognizable while losing task-relevant information needed for training. For this reason, we treat reconstructed-dataset trainability as the primary privacy metric and use MSE to help interpret the character of the reconstructions.

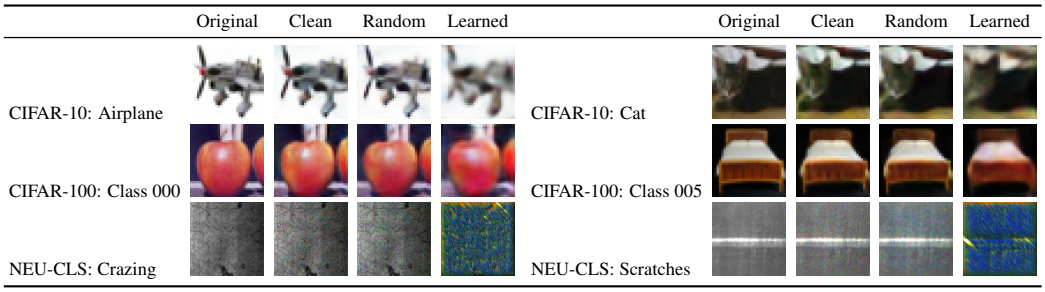


Figure 2: Cross-dataset qualitative reconstruction examples. Each dataset row shows two representative samples under original, clean reconstruction, random perturbation, and learned perturbation settings. Learned perturbations reduce visually reliable class-specific structure in these examples, complementing the trainability reductions in Table 3.

7 Conclusion

We presented a learned client-side weight perturbation defense for mitigating reconstruction leakage in split inference. The defense modifies selected client-side weights before the split point and is selected under an explicit utility constraint, preventing privacy gains from being attributed to degraded classification performance. The inverse-problem motivation explains why changing the client-side feature map can affect reconstruction, while the experiments show that learned perturbations reduce the downstream trainability of reconstructed datasets across multiple reconstruction benchmarks and datasets. These results support learned weight perturbation as a practical direction for mitigating reconstruction leakage in split inference.

AI use statement

The authors used generative AI tools for language polishing, LaTeX editing, translation support, and feedback on clarity. AI tools were not used to produce experimental results or fabricate data; the authors reviewed all AI-assisted content and remain responsible for the final manuscript.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4829–4837, 2016.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284. Springer, 2006.

Ege Erdogan, Alptekin Kupcu, and A. Ercument Cicek. UnSplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. In *Proceedings of the 21st Workshop on Privacy in the Electronic Society*, pp. 115–124. ACM, 2022.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, pp. 16937–16947, 2020.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 201–210, 2016.

Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1998.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Zecheng He, Tianwei Zhang, and Ruby B. Lee. Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pp. 148–162, 2019.

Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 603–618, 2017.

Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. Evaluating gradient inversion attacks and defenses in federated learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 7232–7241, 2021.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5188–5196, 2015.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 691–706, 2019.

Fatemehsadat Mireshghallah, Mohammad Taram, Prachi Ramrakhyani, and Dean Tullsen. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 3–18, 2020.

Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, pp. 19–38, 2017.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*, pp. 739–753, 2019.

Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R. Rabiee, Nicholas D. Lane, and Hamed Haddadi. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal*, 7(5):4505–4518, 2020.

Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2113–2129, 2021.

Roger Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.

Mael Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1310–1321, 2015.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, pp. 3–18, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Kechen Song and Yunhui Yan. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, 285:858–864, 2013.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. NoPeek: Information leakage reduction to share activations in distributed deep learning. *arXiv preprint arXiv:2008.09161*, 2020.

Hongxu Yin, Arun Mallya, Arash Vahdat, Jose M. Alvarez, Jan Kautz, and Pavlo Molchanov. See through gradients: Image batch recovery via GradInversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16337–16346, 2021.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

A Additional Results

A.1 Evaluator Upper Bounds on Real Data

Table 5 reports evaluator performance when the same architectures are trained on real training data and tested on the real held-out test set. These numbers provide an upper-bound reference for interpreting reconstructed-dataset trainability: a reconstructed dataset is more useful when an evaluator trained on it approaches the corresponding real-data accuracy.

Table 5: Evaluator upper bounds on real held-out data. Values are mean accuracy $\pm$ standard deviation over seeds 2026, 2027, and 2028.

Dataset	ResNet20	VGG11-BN	MobileNetV2
CIFAR-10	89.9±0.1	89.3±0.1	82.3±2.2
CIFAR-100	64.4±0.5	62.4±0.2	50.5±0.3
NEU-CLS	97.8±1.0	72.1±40.1	98.7±0.4

The large VGG11-BN variance on NEU-CLS is caused by one anomalous seed; the main text therefore interprets NEU-CLS trainability trends together with the ResNet20 and MobileNetV2 evaluators rather than relying on VGG11-BN alone.

A.2 Additional Protocol Details

This appendix provides implementation-facing details that are omitted from the main text for compactness. These details are not additional assumptions; they clarify how defense training, candidate selection, adaptive attack training, and reconstructed-dataset evaluation are separated in the reported experiments.

Defense-training data flow. For a fixed dataset, victim model, split point, seed, and perturbation protocol, we first construct the data partitions described in Section 6. The clean victim model is kept fixed. During defense training, only the perturbation variables associated with selected client-side tensors are updated. The server-side classifier remains unchanged, so a candidate perturbation must remain compatible with the original downstream prediction path. This design ensures that any privacy gain is evaluated under the same task model rather than under a retrained or weakened server.

Surrogate reconstruction loss and perturbation scale. The surrogate decoder d_ϕ is trained to minimize pixel reconstruction error on its training partition,

$$\mathcal{L}_{\mathrm{dec}}(\phi) = \mathbb{E}\left[\|d_\phi(f^{(s)}_{\theta_c+\Delta\theta_c}(x)) - x\|_2^2\right]. \tag{14}$$

For learned perturbations, we report the relative perturbation norm

$$\rho_\Delta = \left(\frac{\sum_\ell \|\Delta W_\ell\|_F^2}{\sum_\ell \|W_\ell\|_F^2}\right)^{1/2}, \tag{15}$$

where the sums are over perturbed tensors. This normalized scale is also used to construct the norm-controlled random perturbation baselines.

Surrogate decoder usage. The surrogate decoder is used only as a training signal for selecting the client-side perturbation. In the auxiliary-surrogate protocol, the surrogate is trained on $\mathcal{D}_{\text{aux_train}}$ and the perturbation is trained on $\mathcal{D}_{\text{target_train}}$. In the target-surrogate protocol, the surrogate decoder and perturbation are optimized in an alternating loop on $\mathcal{D}_{\text{target_train}}$. In both cases, the selected candidate is chosen using $\mathcal{D}_{\text{target_val}}$ and must satisfy the utility threshold in Eq. 12. The surrogate decoder used during this stage is not reported as the final adaptive attacker.

Candidate filtering and selection. At validation time, candidates that fail the accuracy threshold are discarded before privacy scores are compared. This filtering is important because reconstruction can be made difficult by destroying task-relevant information, which is not an acceptable defense. Among feasible nonzero candidates, we select the candidate with the strongest validation privacy score. The score is used only for selection; the final reported metrics are computed after training new attackers against the selected defended client.

For learned-decoder attacks, the reported adaptive attacker is trained after defense selection. Specifically, auxiliary images are first passed through the selected defended client model to obtain defended split representations, and a new decoder is trained to map these representations back to image space. This decoder is selected on the auxiliary validation partition and then evaluated on held-out target examples. This protocol separates the decoder used as a surrogate privacy signal during defense training from the final attacker used for reporting results.

For trainability evaluation, reconstructed images inherit the labels of their corresponding source images. Each evaluator model is trained only on the reconstructed training set and is tested on the real held-out test set. This metric measures whether reconstructed samples remain useful as a substitute for private training data, which is complementary to pixel-level MSE.

A.3 Additional Benchmark and Metric Details

The main text treats reconstruction methods as evaluation benchmarks rather than as contributions of this paper. We include the protocol details here to make clear how the defended representations are reconstructed and how the reported reconstructed datasets are produced. In all cases, the learned weight perturbation is selected first and is then frozen before the final reconstruction benchmark is run. Therefore, the reported reconstructions evaluate the selected defense rather than the surrogate decoder state used during defense training.

SplitNN decoder benchmark. The SplitNN decoder benchmark follows the common reconstruction setting for split learning, where an adversary fits an image decoder from observed split representations (Pasquini et al., 2021). For a split point s, the attacker constructs paired auxiliary examples (z_i, x_i), where $z_i = f^{(s)}_{\theta_c}(x_i)$ in the clean setting and $z_i = f^{(s)}_{\theta_c+\Delta\theta_c}(x_i)$ in the defended setting. A decoder d_ω is trained on $\mathcal{D}_{\text{aux_train}}$ by minimizing

$$\min_{\omega} \frac{1}{|\mathcal{D}_{\text{aux_train}}|} \sum_{x_i \in \mathcal{D}_{\text{aux_train}}} \|d_\omega(z_i) - x_i\|_2^2. \tag{16}$$

The decoder checkpoint is selected using $\mathcal{D}_{\text{aux_val}}$. After selection, the decoder reconstructs the target partitions from their clean or defended split representations. This adaptive evaluation is important: the defended decoder is retrained on features produced by the selected perturbed client, rather than reusing a decoder trained on clean features.

UnSplit benchmark. UnSplit-style reconstruction is used as an optimization-based benchmark inspired by prior attacks on split learning (Erdogan et al., 2022). Unlike the decoder benchmark, it does not train a representation-to-image decoder. Given an observed target representation z, it directly optimizes an image variable $u \in [0,1]^d$ so that the client-side feature map of u matches z:

$$\min_{u \in [0,1]^d} \|f^{(s)}_{\theta_c}(u) - z\|_2^2 + \lambda_{\text{tv}}\text{TV}(u) + \lambda_{\ell_2}\|u\|_2^2. \tag{17}$$

In our implementation, z is computed from the clean or defended client according to the evaluated condition, while the input optimization uses the fixed clean client map as the inversion model. The optimization is restarted from sample-dependent random initializations, and the reconstruction with the lowest feature-matching error is retained. This benchmark tests whether the learned perturbation transfers beyond the decoder used as the surrogate privacy signal during defense training.

Reconstruction MSE. MSE is computed between the original and reconstructed images after applying the same image scaling convention used by the reconstruction pipeline. It measures pixel-level fidelity but does not fully characterize privacy leakage. For this reason, the main text reports MSE together with reconstructed-dataset trainability rather than using MSE as the sole privacy metric.

Reconstructed-dataset trainability. To evaluate trainability, reconstructed images are treated as a synthetic training set with labels inherited from their source images. Evaluator models are trained on this reconstructed set and tested on real held-out test images. If an evaluator trained on reconstructed images performs well on real test data, then the reconstruction retains training-relevant semantic information even if the images are visually imperfect. Conversely, a large trainability drop indicates that the reconstructed data has become less useful as a substitute for private data.

A.4 Random Perturbation Baselines

The main text reports clean and learned perturbation settings for compactness. Random perturbation baselines with relative norm 0.05 are used as the primary random comparison and are included in the experiment outputs. These baselines perturb the same selected client-side weights but do not use reconstruction feedback during training.

For a selected client-side tensor W_ℓ, the random baseline samples a noise tensor with the same shape and rescales it to a prescribed relative norm, $\|\Delta W_\ell^{\mathrm{rand}}\|_F = \rho\|W_\ell\|_F$. We evaluate relative norms $\rho \in \{0.02, 0.05\}$ when available. Random perturbations are subject to the same utility feasibility rule as learned perturbations. This comparison separates the effect of simply changing client-side weights from the effect of using reconstruction feedback to learn a targeted perturbation.

Table 6: Random perturbation baseline trainability accuracy with relative norm $\rho = 0.05$. All entries are percentages, and Drop is Clean minus Random in percentage points. Bold marks drops larger than 10 percentage points.

Benchmark	Dataset	Split	ResNet20			VGG11-BN			MobileNetV2		
			Clean	Random	Drop	Clean	Random	Drop	Clean	Random	Drop
SplitNN	CIFAR-10	L1	89.5	89.4	0.1	88.9	88.9	0.0	80.1	82.7	-2.6
SplitNN	CIFAR-10	L2	84.0	83.7	0.2	86.2	86.3	-0.1	77.9	77.6	0.4
SplitNN	CIFAR-10	L3	20.6	22.2	-1.6	28.4	30.8	-2.4	26.5	25.7	0.9
SplitNN	CIFAR-100	L1	63.7	63.3	0.4	62.5	62.2	0.3	49.3	49.7	-0.4
SplitNN	CIFAR-100	L2	51.4	52.2	-0.8	57.7	58.5	-0.7	47.0	47.3	-0.3
SplitNN	CIFAR-100	L3	11.1	13.0	-1.9	18.5	20.8	-2.4	14.3	16.6	-2.3
SplitNN	NEU-CLS	L1	96.8	95.9	0.8	71.1	49.9	**21.2**	97.5	98.2	-0.7
SplitNN	NEU-CLS	L2	94.2	92.8	1.4	79.5	49.4	**30.2**	98.1	97.6	0.5
SplitNN	NEU-CLS	L3	74.4	85.0	-10.6	64.9	57.2	7.7	89.6	86.9	2.8
UnSplit	CIFAR-10	L1	87.5	87.2	0.3	61.4	86.1	-24.7	77.4	78.8	-1.4
UnSplit	CIFAR-10	L2	78.9	75.1	3.8	10.0	29.6	-19.6	59.0	56.6	2.3
UnSplit	CIFAR-10	L3	19.9	20.0	-0.1	10.0	10.0	0.0	18.9	18.1	0.8
UnSplit	CIFAR-100	L1	61.3	59.9	1.4	58.6	58.5	0.2	46.6	46.7	-0.1
UnSplit	CIFAR-100	L2	51.0	52.4	-1.4	15.7	30.0	-14.3	34.9	35.0	-0.1
UnSplit	CIFAR-100	L3	6.2	5.5	0.7	1.0	1.0	0.0	5.3	4.9	0.3
UnSplit	NEU-CLS	L1	86.9	83.0	4.0	51.3	56.5	-5.2	91.9	91.7	0.3
UnSplit	NEU-CLS	L2	61.0	56.5	4.5	35.4	35.6	-0.2	75.1	61.4	**13.7**
UnSplit	NEU-CLS	L3	19.2	31.2	-12.0	26.8	25.8	0.9	26.0	28.2	-2.2

A.5 Interpretation of Qualitative Examples

Qualitative examples are intended to help readers inspect the character of the reconstructions, but they are not used as the primary privacy evidence. Visual plausibility and training usefulness can diverge: an image may look blurred or distorted while still preserving class-discriminative structure, and an image may remain partially recognizable while losing enough consistent structure to become

less useful for training downstream classifiers. The main paper therefore uses qualitative panels as supporting evidence and relies on trainability and MSE for quantitative evaluation.

A.6 Additional Qualitative Examples

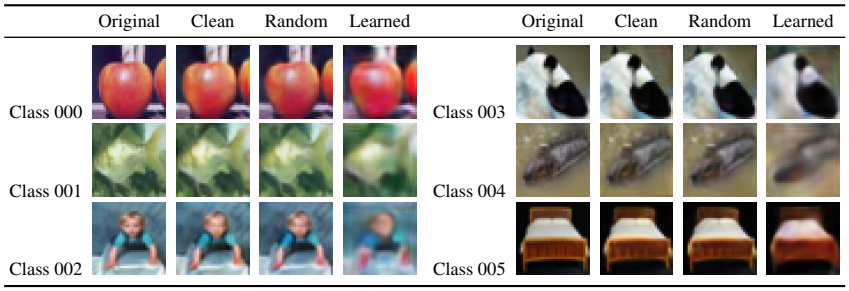


Figure 3: Additional CIFAR-100 qualitative examples for the L2 SplitNN setting.

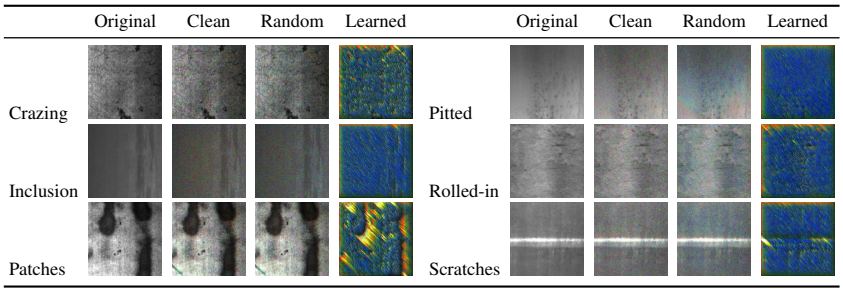


Figure 4: Additional NEU-CLS qualitative examples.