Supplementary Material for FedErase: Personalized Federated Unlearning for Text-to-Image Diffusion Models

This supplementary document provides additional technical and experimental details complementing the main paper. In Sec. 1, we elaborate on the design of FedErase-R and FedErase-S. Sec. 2 reports extended ablation studies and training details.

1. Additional Details of FedErase Variants

1.1. Additional Details for FedErase-R

Motivation. FedErase-R is built on the observation that, under PFU, each client requests to forget a *different* concept. If all clients share the same LoRA adapter (as in FedAvg), the combined forget signals conflict in parameter space, causing cross-client interference and catastrophic degradation of unrelated concepts. FedErase-R resolves this by *decoupling* the model into:

$$M_i = (f_r, f_{h,i}),$$

where f_r is a shared U-Net representation collaboratively trained across clients, and $f_{h,i}$ is a client-specific adapter that exclusively encodes client i's unlearning behaviour.

Model decomposition. Let θ_r denote the parameters of the shared representation, and $\theta_{h,i}$ denote the LoRA parameters for client i. The diffusion model prediction can be written as

$$\hat{x}_\theta(x_t, t, c) = f_r\big(x_t, t, c; \theta_r\big) + f_{h,i}\big(x_t, t, c; \theta_{h,i}\big).$$

Only θ_r is aggregated; $\theta_{h,i}$ remains private to client i.

Two-stage local optimization. For each communication round, client i performs:

1. Head update (personalization)

$$\theta_{h,i} \leftarrow \theta_{h,i} - \eta_h \nabla_{\theta_{h,i}} \mathcal{L}_{\text{unlearn},i}.$$

This step injects *client-specific forgetting*: the adapter learns how to suppress *only* its own concept.

2. Representation update (federated)

$$\Delta\theta_{r,i} = -\eta_r \nabla_{\theta_r} \mathcal{L}_{\text{unlearn},i},$$

which is uploaded to the server and aggregated:

$$\theta_r \leftarrow \theta_r + \frac{1}{K}\sum_{i=1}^{K} \Delta\theta_{r,i}.$$

Why representation-head decoupling works in PFU. Federated unlearning with heterogeneous client concepts requires two competing goals:

- Each client must *forget its own concept*. - All clients must *share* a stable generative representation of the world.

FedErase-R achieves this by:

- Encoding **forgetting signals** *only in* the private $\theta_{h,i}$, which prevents interference across clients.
- Allowing θ_r to learn **universal low-level and mid-level diffusion features** that benefit all clients.
- Ensuring the global model remains stable even when clients forget different concepts.

Effective decoupling of concept-relevant directions. Although both f_r and $f_{h,i}$ contribute to the diffusion output, PFU requires the model to modify primarily the locally relevant directions for forgetting. Empirically, we observe:

- Gradients w.r.t. θ_r tend to be generic (background, color distribution, noise-level prediction).
- Gradients w.r.t. $\theta_{h,i}$ carry *high-frequency, concept-specific signals*.

This natural separation aligns with the design of parameter-efficient fine-tuning (PEFT) methods, and confirms that forgetting is most effective when localized within $\theta_{h,i}$.

Regularization across rounds. To avoid overfitting to the forget set, we apply a lightweight proximal regularizer during the head update:

$$\mathcal{L}_{\text{reg},i} = \gamma \|\theta_{h,i} - \theta_{h,i}^{(0)}\|_2^2,$$

where $\theta_{h,i}^{(0)}$ is the hypernetwork- or zero-initialized LoRA state. This stabilizes per-client forgetting and limits deviation from the original diffusion behaviour.

Interpretation. FedErase-R can be interpreted as a *personalized gradient routing* mechanism: forgetting gradients are routed into private heads, while general gradients flow into the shared backbone. This design is lightweight, communication-efficient, and stable across heterogeneous PFU settings.

1.2. Additional Details for FedErase-S

Motivation. FedErase-S aims to localize concept-relevant parameters such that unlearning can be performed precisely with minimal collateral damage. Given a diffusion model $\hat{x}_\theta$ with parameter vector $\theta = (\theta_1, \ldots, \theta_N) \in \mathbb{R}^N$, our goal is to identify a small subset of parameters $\theta_{\mathcal{H}} = \{\theta_{h_1}, \ldots, \theta_{h_m}\}$ with $m \ll N$ that dominantly encode the target concept. Updating only this sparse subset allows the model to forget the concept while preserving unrelated generative capabilities.

Concept contribution score. To measure the importance of each parameter for the unlearning objective $\mathcal{L}_{\text{unlearn}}$, we consider the scalar quantity

$$s_h \triangleq \theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}},$$

which captures both the magnitude of θ_h and the sensitivity of the loss to perturbations in that direction. Intuitively, $s_h > 0$ implies that scaling down θ_h decreases $\mathcal{L}_{\text{unlearn}}$, making θ_h an effective node for removing the concept.

First-order analysis. Let $\alpha \in (0, 1)$ be a scaling factor applied to a single parameter: $\theta_h \mapsto \alpha\theta_h$. A first-order Taylor expansion yields

$$\mathcal{L}_{\text{unlearn}}(\alpha\theta_h) = \mathcal{L}_{\text{unlearn}}(\theta_h) + (\alpha - 1)\,\theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}} \quad (1)$$
$$+ \mathcal{O}((\alpha - 1)^2). \quad (2)$$

Thus, the loss decreases when

$$\theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}} > 0. \quad (3)$$

We parameterize the scaling as

$$\alpha = 1 - \mu\,\theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}}, \qquad 0 < \mu \ll 1,$$

giving the approximate update

$$\mathcal{L}_{\text{unlearn}}(\alpha\theta_h) \approx \mathcal{L}_{\text{unlearn}}(\theta_h) - \mu\,\theta_h^2 \left(\nabla_{\theta_h} \mathcal{L}_{\text{unlearn}}\right)^2,$$

which guarantees a decrease in the loss whenever Eq. (3) holds.

Theorem (Identification of Sparse Nodes). *A parameter θ_h contributes positively to concept removal if and only if*

$$\theta_h \nabla_{\theta_h} \mathcal{L}_{unlearn} > 0.$$

Moreover, the expected reduction in $\mathcal{L}_{unlearn}$ from scaling down θ_h is proportional to $(\theta_h \nabla_{\theta_h} \mathcal{L}_{unlearn})^2$. Thus, θ_h is a valid sparse node precisely when the above condition holds. This provides a principled criterion for identifying concept-relevant parameters.

Hard sparsity selection

Directly thresholding $\theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}}$ may be numerically unstable when gradients are close to zero. To robustly assess whether a parameter is consistently aligned with the unlearning direction, we propose sampling a sequence:

$$\theta_h^1 = \theta_h, \qquad \theta_h^{k+1} = \theta_h^k \left(1 - \mu\,\theta_h^k \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}}(\theta_h^k)\right),$$

for $k = 1, \ldots, K-1$. Accumulating the contribution scores along this trajectory,

$$\sum_{k=1}^{K} \theta_h^k \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}}(\theta_h^k) > \tau,$$

with threshold $\tau > 0$, determines whether θ_h should be included in the sparse set. This adaptive sampling densifies evaluation in regions with small gradients, improving robustness.

Soft sparsity with adaptive weights

Instead of binary selection, we may assign a continuous weight

$$w(\theta_h) = \frac{1}{1 + \exp\left[-\alpha\left(\theta_h \nabla_{\theta_h} \mathcal{L}_{\text{unlearn}} + \beta\right)\right]},$$

where α, β control sharpness and bias. Large contribution scores yield $w(\theta_h) \approx 1$, while ambiguous parameters receive small weights. This soft approach updates all parameters but prioritizes concept-relevant ones, yielding smoother optimization and improved stability in PFU.

Summary. FedErase-S leverages first-order analysis to derive a principled, interpretable measure of concept relevance for each parameter. Both hard and soft sparsity mechanisms are supported, offering a practical trade-off between precision, stability, and computational cost in personalized federated unlearning.

2. Additional Experimental Details

This section provides additional experimental details for the FedErase family under the personalized federated unlearning (PFU) setting. All experiments follow the same five-client PFU protocol as described in the main paper, where each client erases a different target concept. Metrics are averaged across clients and random seeds.

2.1. Training Details and Hyperparameters

All experiments are conducted on a single NVIDIA RTX 4090 GPU with 24GB memory. We implement our PFU framework using PyTorch and the `diffusers` library, and apply LoRA-based fine-tuning for all adapter updates. Stable Diffusion v1.5 is used as the base model for all methods unless otherwise specified.

Table 1. Instance concepts and representative prompts for the 5 clients.

Client	Concept	Example forget prompt
C_1	Winnie the Pooh	"Winnie the Pooh walking in a peaceful forest, holding a honey pot, soft lighting"
C_2	Mickey Mouse	"Mickey Mouse standing in a colorful amusement park, full-body, realistic photo"
C_3	Superman	"Superman flying above a modern city skyline, blue suit and red cape, cinematic lighting"
C_4	Harry Potter	"Harry Potter casting a spell with a glowing wand, wearing glasses and robe, Hogwarts background"
C_5	Iron Man	"Iron Man in red-gold armor launching into the sky, glowing arc reactor, action shot"

Optimization. We use AdamW for both representation and adapter parameters. For all PFU variants, the LoRA adapter uses a learning rate of 1×10^{-4}, while the shared representation f_r uses a smaller learning rate of 5×10^{-5} to prevent instability during federated aggregation. Each client performs 5 local epochs per communication round, and we train for 50 communication rounds for all federated experiments.

LoRA configuration. We adopt rank $r = 16$ and scaling factor $\alpha = 8$ for all LoRA modules. Unless explicitly stated, the same LoRA configuration is shared across all FedErase variants and baseline methods (FMN, ESD, FedAvg).

Batching and sampling. Due to single-GPU memory constraints, each client uses a batch size of 4. We follow the original Stable Diffusion noise schedule and uniformly sample timestep t during the forward diffusion process. All experiments are run with a fixed seed (42) unless otherwise stated in stability evaluations.

PFU-specific hyperparameters. For the prior-preserving loss, we set $\lambda = 1.0$. For FedErase-G, we use a dummy-gradient regularization coefficient $\lambda_{\text{reg}} = 10^{-4}$. For FedErase-S, we use a sparsity ratio of top-3% unless otherwise stated in the ablation studies. For FedErase-H, client embeddings e_i are randomly initialized in a 64-dimensional space.

Inference. All generated images in the qualitative sections use a DDIM sampler with 50 steps and classifier-free guidance scale 7.5. Prompts are standardized across methods to ensure fair comparisons.

This configuration fits comfortably within the 24GB memory budget of a RTX4090 GPU and allows reproducible training for all PFU variants evaluated in the paper.

2.2. Concepts and Prompt Design

We list here the exact concepts and representative prompts used in our PFU experiments. For all settings, we use a single-sentence prompt to reduce ambiguity and to ensure that the target concept is explicitly grounded in the text. For example, when forgetting *"Winnie the Pooh"*, we use: *"Winnie the Pooh walking in a peaceful forest, holding a honey pot, soft lighting"* This makes the concept visually and textually entangled, thereby providing a challenging yet well-defined target for unlearning.

As shown in Table 1, each client is assigned a distinct instance-level concept for forgetting. Taking the instance category as an example, client C_1 is required to forget "Winnie the Pooh", client C_2 to forget "Mickey Mouse", and so on, ensuring that the PFU setting reflects heterogeneous client-specific forgetting requests.

2.3. Component-wise Ablation of FedErase-U

We begin by analyzing the contribution of each component in the unified variant (FedErase-U). We remove one module at a time from {R, H, G, S} and measure forgetting strength (CLIP↓), fidelity (FID↓), and deletion effectiveness (MIA↑) on *instance-concept* PFU.

Table 2. Component-wise ablation of FedErase-U on instance concepts. Removing R or S significantly weakens forgetting; removing H or G mainly affects stability.

Variant	**CLIP↓**	**FID↓**	**MIA↑**
Full FedErase-U	0.330	21.0	0.63
–R (no decoupling)	0.371	24.8	0.54
–H (no hypernetwork)	0.344	19.3	0.59
–S (no sparsity)	0.356	22.7	0.57
–G (no dummy updates)	0.338	**17.5**	0.61

Observations. Removing R leads to degraded PFU due to severe cross-client interference. Removing S results in weaker forgetting and stronger collateral damage. Removing H or G causes higher variance, consistent with our main-paper findings.

2.4. Sparsity Ratio Ablation (FedErase-S)

We vary the sparsity ratio (top-k% updated parameters) to understand how localized modifications influence forgetting.

Observations. A moderate sparsity ratio (3–5%) provides the best forgetting–fidelity trade-off.

Table 3. Effect of sparsity ratio on PFU (instance concepts). Too sparse → insufficient forgetting; too dense → degraded fidelity.

Sparsity (Top-%)	**CLIP↓**	**FID↓**	**MIA↑**
1%	0.342	15.2	0.65
3%	0.309	**15.1**	**0.71**
5%	**0.287**	16.5	0.73
10%	0.301	17.8	0.70
20%	0.335	19.6	0.62

2.5. Hypernetwork Size Ablation (FedErase-H)

We examine three hypernetwork configurations: *Small-HN* (2-layer MLP), *Medium-HN* (3-layer lightweight transformer), *Large-HN* (5-layer transformer).

Table 4. Impact of hypernetwork size on PFU (style concepts). Larger models improve forgetting but cause higher client-to-client variance.

HN Size	**CLIP↓**	**FID↓**	**Variance↓**
Small	0.332	17.9	**0.008**
Medium	0.310	**16.8**	0.021
Large	**0.297**	19.7	0.041

Observations. Medium-HN offers the best balance between forgetting performance and stability.

2.6. Dummy Gradient Steps & Stability (FedErase-G)

FedErase-G relies on synthetic dummy updates. We test different numbers of dummy-generator optimization steps per round.

Table 5. FedErase-G becomes unstable with excessive dummy steps.

Dummy Steps	**CLIP↓**	**FID↓**	**MIA↑**
1	0.355	18.1	0.60
2	0.344	**17.8**	0.61
5	**0.332**	19.4	0.58
10	0.341	22.7	**0.63**

Observations. 2–3 steps are optimal. Larger step counts cause significant generative collapse (FID↑).

2.7. Multi-Run Stability (Across Random Seeds)

We repeat PFU training three times and report standard deviations.

Table 6. Run-to-run stability under PFU (instance concepts). FedAvg is unstable; R/S are most stable; H/G/U show higher variance.

Method	CLIP std↓	FID std↓	MIA std↓
FedAvg	0.027	0.92	0.031
FedErase-R	**0.011**	**0.41**	0.014
FedErase-S	0.013	0.45	**0.012**
FedErase-G	0.032	1.35	0.028
FedErase-H	0.019	0.77	0.022
FedErase-U	0.028	1.21	0.027

Observations. FedErase-R/S exhibit strong run-to-run reliability, while G/H/U are more sensitive to initialization and local optimization noise.

2.8. Overall Summary

Across all ablations:

- **R (decoupling)** is essential for preventing cross-client interference.
- **S (sparse unlearning)** yields the best forgetting–fidelity trade-off.
- **H (hypernetwork)** improves initialization but introduces variance.
- **G (dummy gradients)** can achieve strong forgetting but is unstable.
- **FedAvg cannot satisfy PFU**, as a single global adapter cannot forget different concepts for different clients.

These findings support the conclusions drawn in the main paper and highlight the importance of personalized and sparse mechanisms for robust PFU.