BEYOND IMPAIR–REPAIR: TUNING-EFFICIENT MACHINE UNLEARNING WITH ANTI-OVERREPAIR

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning aims to remove the influence of selected training data without retraining a model from scratch while preserving performance on the remaining data. However, many approximate unlearning methods jointly optimize competing forgetting and retention objectives, introducing unlearning-specific hyperparameters that can be sensitive to the forgetting ratio, model architecture, and dataset. Repeated scenario-specific recalibration therefore adds nontrivial tuning and computational overhead, potentially reducing the practical efficiency advantage over retraining. We propose **TriTune**, a three-stage framework that decomposes this coupled optimization into a sequence of stage-wise, single-objective optimization problems. By avoiding directly competing objectives within each stage, TriTune simplifies hyperparameter selection and reduces sensitivity to hyperparameter choices. TriTune further introduces an adaptive anti-overrepair mechanism that prevents utility repair from restoring forgotten behavior, together with lightweight self-calibration for learning-rate initialization and stage termination. Experiments across diverse datasets, architectures, forgetting ratios, and learning tasks demonstrate effective forgetting, strong utility preservation, and reduced dependence on manual hyperparameter tuning.

1 INTRODUCTION

Over the past decade, machine learning has progressed rapidly, from early deep neural networks such as AlexNet (Krizhevsky et al., 2012) to today's large-scale foundation models and large language models. Alongside this progress, both model capacity and the scale of training datasets have increased substantially, making modern learning systems increasingly dependent on large and heterogeneous data collections. Such datasets may contain sensitive, private, or user-specific information, raising growing concerns about data ownership, privacy, and deletion (Guo et al., 2020; Bourtoule et al., 2021). These concerns have been further reinforced by privacy regulations and the "right to be forgotten," which motivate the removal of selected training data and their influence from deployed models (Bourtoule et al., 2021; Wang et al., 2024). A straightforward solution is to delete the target data and retrain the model from scratch; however, full retraining can be computationally expensive and time-consuming, particularly for modern large-scale models (Golatkar et al., 2020; Bourtoule et al., 2021). This challenge has motivated the development of machine unlearning, which aims to remove the influence of selected training data from a trained model without requiring complete retraining (Guo et al., 2020; Bourtoule et al., 2021).

Current machine unlearning methods can be broadly categorized into exact and approximate unlearning (Wang et al., 2024). Because exact unlearning is often computationally expensive or difficult to scale to modern models, much recent research has focused on approximate alternatives. A central motivation behind these methods is to reduce the computational cost of retraining from scratch (Fan et al., 2024). However, approximate unlearning often introduces a number of unlearning-specific hyperparameters that must be carefully tuned. Recent work has shown that existing unlearning methods can be highly sensitive to hyperparameter choices and may exhibit unstable performance across different unlearning scenarios (Zhong et al., 2025). This issue is further complicated by the inherent tension between forgetting the target data and preserving utility on the retained data, which naturally gives rise to competing optimization objectives (Zhou et al., 2025). As a result, hyperparameter settings that work well in one scenario may fail when the forgetting ratio, dataset, model architecture, or task changes, often requiring repeated manual trials and recalibration (Zhong et al., 2025; Fan

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.

et al., 2024). In contrast, full retraining can typically reuse the training recipe established for the original model and therefore requires substantially less additional tuning. Consequently, once tuning overhead is taken into account, the practical time-efficiency advantage of approximate unlearning over retraining may be much smaller than suggested by the runtime of a single successful run. This motivates a natural question: can we design a machine unlearning method that adapts across diverse unlearning scenarios with minimal manual tuning, ideally requiring only a single calibrated run?

A second challenge arises from the competing objectives of forgetting and retention. When the corresponding losses are optimized jointly, the resulting trade-off can become highly unbalanced: aggressive forgetting may substantially degrade performance on retained data, whereas strong utility preservation may lead to insufficient forgetting (Zhou et al., 2025; Khosrowshahli et al., 2026). To mitigate this conflict, another line of work adopts a two-stage impair–repair paradigm, in which the model is first deliberately impaired to suppress the influence of the forget set and is subsequently repaired using retained data to recover lost utility (Rizwan et al., 2026). By separating these two objectives temporally, the repair stage can partially or substantially restore retained performance after a strong impairment, provided that the repair process is properly calibrated (Rizwan et al., 2026). However, this separation introduces a different risk: continued repair may inadvertently restore behavior associated with the forget set, partially reversing the forgetting effect achieved during impairment. We refer to this phenomenon as *over-repair*. This raises a second question: how can we preserve the benefits of repair while preventing forgotten behavior from being recovered?

To address the first challenge, we reduce the sensitivity of unlearning to hyperparameter choices by decomposing the optimization into separate impair and repair stages. Instead of jointly optimizing competing forgetting and retention objectives, each stage focuses on a simpler goal: the impair stage only needs to reduce performance on the forget set, while the repair stage only needs to recover performance on the retain set. This stage-wise decomposition reduces the interaction between competing objectives and makes the optimization less sensitive to hyperparameter choices, addressing a practical limitation that has been observed in existing machine unlearning methods (Zhong et al., 2025; Zhou et al., 2025). Similar impair–repair strategies have also shown that separating forgetting from subsequent utility recovery can provide an effective mechanism for selective unlearning (Rizwan et al., 2026). This reduced sensitivity further enables simple automatic calibration: during impairment, we perform a one-step gradient-ascent update and increase the step size only when the forget-set performance does not decrease sufficiently; during repair, we reduce the learning rate whenever retain performance fails to improve, thereby avoiding repeated manual tuning. However, as discussed above, repair itself may eventually lead to *over-repair*: once retain utility has been largely restored, further optimization may begin to recover features shared by the retain and forget sets, causing the forget-set performance to improve as well. To prevent this, we introduce a third anti-overrepair stage that is activated automatically once the repair process reaches a predefined level of utility recovery. At this stage, an additional objective is introduced to constrain the behavioral gap between the forget and retain sets. Rather than manually specifying the desired gap, we use validation-derived references, such as the retain–validation KL divergence and loss difference, and update these references adaptively during optimization. In this way, the overall three-stage framework reduces both manual hyperparameter tuning and the risk of restoring forgotten information while preserving retained utility.

Our contributions are fourfold. **(1)** We identify scenario-specific hyperparameter tuning as an overlooked practical cost in approximate machine unlearning. **(2)** We introduce a three-stage *impair–repair–anti-overrepair* framework to prevent continued repair from restoring forgotten behavior. **(3)** We develop simple self-calibration mechanisms that reduce dependence on manually tuned hyperparameters across different unlearning scenarios. **(4)** Extensive experiments across multiple forgetting ratios, datasets, model architectures, and tasks demonstrate effective forgetting, strong retained utility, and reduced tuning overhead.

2 Related Work

2.1 Machine Unlearning

Machine unlearning aims to remove the influence of selected training data without full retraining. Existing approaches include certified or exact removal methods, model scrubbing, decision-boundary manipulation, sparsity-based unlearning, and saliency-guided parameter updates (Guo et al., 2020;

Golatkar et al., 2020; Bourtoule et al., 2021; Chen et al., 2023; Jia et al., 2023; Fan et al., 2024). Recent surveys further categorize these approaches into exact and approximate unlearning, with approximate methods receiving increasing attention because of their lower computational cost (Wang et al., 2024; Nguyen et al., 2025).

2.2 Optimization and Tuning in Machine Unlearning

Approximate unlearning commonly requires balancing multiple objectives and carefully selecting learning rates, loss weights, stopping criteria, and other unlearning-specific hyperparameters. SalUn improves stability through gradient-based weight saliency, while RUM shows that unlearning difficulty can vary substantially with the characteristics of the forget set (Fan et al., 2024; Zhao et al., 2024). More recent work explicitly studies optimization instability: DualOptim uses separate adaptive optimizers to reduce hyperparameter sensitivity, while gradient-conflict-aware methods such as Learning to Unlearn while Retaining, MUNBa, and EUPMU improve the balance between forgetting and utility preservation (Zhong et al., 2025; Patel & Qiu, 2025; Wu & Harandi, 2025; Zhou et al., 2025). Unlike these methods, our focus is on reducing scenario-specific tuning itself through automatic stage-wise calibration.

2.3 Forgetting–Retention Balance

The competing goals of forgetting and retention can lead to gradient conflicts, objective dominance, or excessive utility degradation (Patel & Qiu, 2025; Wu & Harandi, 2025; Zhou et al., 2025). Sequential impair–repair methods instead separate forgetting from utility recovery, first degrading knowledge associated with the forget set and then repairing the model using retained data (Rizwan et al., 2026). Although this separation simplifies optimization, continued repair may also recover behavior associated with the forget set. Our work explicitly studies this failure mode, which we term *over-repair*, and extends the conventional two-stage paradigm with a third anti-overrepair stage.

3 Preliminaries

Notation. Let f_{θ_0} be the original model and f_θ the model being unlearned. The training, forget, retain, and held-out validation sets are denoted by $\mathcal{D}$, $\mathcal{D}_f$, $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$, and $\mathcal{D}_v$, respectively. The collection of forgetting units is $\mathcal{G}_f$; each $g \in \mathcal{G}_f$ identifies corresponding forget and validation subsets $\mathcal{D}_f^g$ and $\mathcal{D}_v^g$. For an input or conditioning variable x, $b_\theta(x) = \Phi(f_\theta, x)$ denotes a task-appropriate model behavior, and $\bar{b}_\theta(\mathcal{S}) = \mathrm{Agg}\{b_\theta(x) : x \in \mathcal{S}\}$ denotes its aggregate over a subset $\mathcal{S}$. We use $d_\mathcal{B}$ and $d_\mathcal{R}$ for behavioral-alignment and retain-repair discrepancies. Given a task loss $\ell_{\mathrm{task}}(\theta; z)$, the empirical risk is $\mathcal{R}(\theta; \mathcal{S}) = |\mathcal{S}|^{-1} \sum_{z \in \mathcal{S}} \ell_{\mathrm{task}}(\theta; z)$, and $\epsilon > 0$ ensures numerical stability. In Section 4, η is a learning rate, t indexes repair epochs, q_t is normalized repair progress, ρ is the anti-overrepair activation threshold, and λ_t is the adaptive gate.

4 TriTune: Stage-wise Tuning-Efficient Machine Unlearning

We introduce **TriTune**, a three-stage machine unlearning framework based on *stage-wise objective decomposition*. Conventional unlearning often jointly optimizes competing objectives, e.g., $\mathcal{L}_{\mathrm{joint}} = \lambda_r \mathcal{L}_{\mathrm{retain}} + \lambda_f \mathcal{L}_{\mathrm{forget}}$, where the effective balance depends on the loss scales, learning rate, training duration, and the choice of λ_r and λ_f, making the optimization sensitive to different datasets, models, and forget-set configurations (Zhong et al., 2025; Patel & Qiu, 2025; Zhou et al., 2025). TriTune instead separates the process into *Forgetting Alignment*, *Utility Repair*, and *Anti-overrepair Tuning*, so that each stage is dominated by a single objective. Normalized objectives reduce scale mismatch, stage duration is controlled by objective improvement, and forgetting preservation is activated through an adaptive gate rather than a fixed trade-off coefficient or manually selected transition epoch. In this way, TriTune transforms a coupled tuning problem into a sequence of stage-wise, single-objective optimization problems without directly competing objectives, making hyperparameter selection simpler and more direct, and thereby reducing sensitivity to hyperparameter choices.

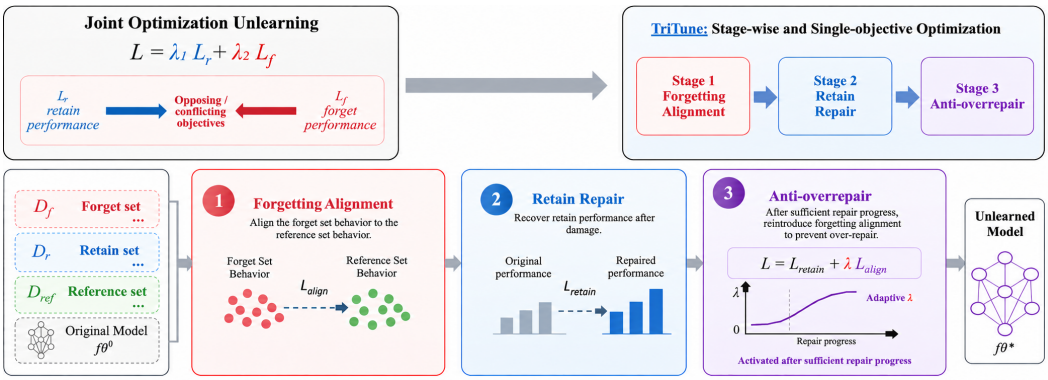


Figure 1: Overview of TriTune. Joint forgetting–retention optimization creates conflicting objectives (top left), whereas TriTune applies reference-guided forgetting alignment, retain repair, and progress-gated anti-overrepair in sequence.

4.1 Stage 1: Behavior–Task Forgetting Alignment

A common strategy in approximate unlearning is to impair the original model by increasing a task loss on the forget set (Golatkar et al., 2020; Chen et al., 2023; Fan et al., 2024). However, an unconstrained forgetting objective such as

$$\max_{\theta} \mathcal{R}(\theta; \mathcal{D}_f) \tag{1}$$

does not specify what constitutes an appropriate amount of forgetting. Excessive optimization can unnecessarily damage retained utility, whereas insufficient optimization may leave substantial information about the forget set.

TriTune replaces unbounded degradation with *reference-guided forgetting alignment*. The objective is to move forgotten samples toward behavior observed on held-out samples rather than simply making their predictions arbitrarily worse. Reference-aligned formulations similarly use external behavior to define a bounded unlearning target (Khosrowshahli et al., 2026).

Behavioral Reference. Let $\mathcal{G}_f$ denote the collection of forgetting units, where a unit may represent a sample group, class, domain, or concept. For each $g \in \mathcal{G}_f$, let $\mathcal{D}_f^g$ and $\mathcal{D}_v^g$ denote the corresponding forget and reference subsets. Using the frozen original model, we construct the reference behavior

$$\bar{b}_g^{\text{ref}} = \text{Agg}\left(\{b_{\theta_0}(x) : x \in \mathcal{D}_v^g\}\right). \tag{2}$$

We define the normalized behavioral alignment loss as

$$\mathcal{L}_{\text{behavior}}(\theta) = \frac{1}{|\mathcal{G}_f|} \sum_{g \in \mathcal{G}_f} \frac{d_{\mathcal{B}}\left(\bar{b}_\theta(\mathcal{D}_f^g), \bar{b}_g^{\text{ref}}\right)}{d_{\mathcal{B}}\left(\bar{b}_{\theta_0}(\mathcal{D}_f^g), \bar{b}_g^{\text{ref}}\right) + \epsilon}. \tag{3}$$

The denominator normalizes the current discrepancy by the initial discrepancy of the original model. As a result, the term has a consistent relative interpretation across datasets and forget settings.

Task-level Reference. Behavioral alignment alone may not ensure that forgotten samples reach an appropriate task-level state. We therefore also align a task-specific empirical risk with a held-out reference value. For any subset $\mathcal{S}$, let

$$\mathcal{R}(\theta; \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{z \in \mathcal{S}} \ell_{\text{task}}(\theta; z) \tag{4}$$

denote the empirical risk induced by the task-appropriate per-sample loss ℓ_{task}. We define

$$\mathcal{R}_f(\theta) = \mathcal{R}(\theta; \mathcal{D}_f), \qquad \mathcal{R}_{\text{ref}} = \mathcal{R}(\theta_0; \mathcal{D}_v) \tag{5}$$

as the forget-set risk and the reference risk of the original model, respectively.

We define

$$\mathcal{L}_{\text{task}}(\theta) = \frac{|\mathcal{R}_f(\theta) - \mathcal{R}_{\text{ref}}|}{|\mathcal{R}_{\text{ref}}| + \epsilon}. \tag{6}$$

The two components are combined as

$$\mathcal{L}_{\text{align}}(\theta) = \mathcal{L}_{\text{behavior}}(\theta) + \mathcal{L}_{\text{task}}(\theta). \tag{7}$$

Both terms are normalized by their reference scales, reducing scale mismatch and avoiding an additional weighting coefficient. Stage 1 therefore retains a single overall target: forgetting alignment.

Self-Calibrated Update Selection. Because Stage 1 optimizes only $\mathcal{L}_{\text{align}}$, update selection is direct. Following the motivation of adaptive unlearning optimization (Zhong et al., 2025), TriTune evaluates the ordered candidate step sizes

$$\eta \in \eta_0 \cdot \{10^2, 10^1, \ldots, 10^{-6}\} \cup \{0\}. \tag{8}$$

For gradient $g_0 = \nabla_\theta \mathcal{L}_{\text{align}}(\theta_0)$, each candidate produces

$$\theta_1(\eta) = \theta_0 - \eta g_0. \tag{9}$$

Candidates are tested from left to right, and the search stops at the first candidate satisfying

$$\mathcal{L}_{\text{align}}\left(\theta_1(\eta)\right) < \mathcal{L}_{\text{align}}(\theta_0). \tag{10}$$

If no candidate improves the loss, $\eta = 0$ is used and the update is skipped. Later epochs apply the same first-decrease rule by reducing the trial rate tenfold until an improving step is found; the best alignment checkpoint is retained. Because all trials share one checkpoint and run, this is local calibration rather than an end-to-end grid search.

4.2 Stage 2: Retain Utility Repair

Although Stage 1 is designed to target forgotten behavior, parameter updates may also perturb behavior on retained samples. Retain-data repair and teacher-matching objectives are commonly used to recover utility after unlearning (Kurmanji et al., 2023; Rizwan et al., 2026). Stage 2 therefore restores the current model toward the original model on $\mathcal{D}_r$.

For every retained sample $x \in \mathcal{D}_r$, we freeze the original behavioral representation

$$b_0(x) = b_{\theta_0}(x). \tag{11}$$

The retain objective is

$$\mathcal{L}_{\text{retain}}(\theta) = \frac{1}{|\mathcal{D}_r|} \sum_{x \in \mathcal{D}_r} d_{\mathcal{R}}\left(b_\theta(x), b_0(x)\right). \tag{12}$$

Here, $d_{\mathcal{R}}$ is a task-appropriate retain discrepancy and may differ from the behavioral discrepancy used for forgetting alignment.

At the beginning of repair, no forgetting loss is included:

$$\lambda_t = 0. \tag{13}$$

Therefore,

$$\mathcal{L}_{\text{repair}} = \mathcal{L}_{\text{retain}}. \tag{14}$$

Stage 2 is therefore a pure utility-recovery stage. Starting from a fixed trial value, it backtracks by a factor of ten and accepts the first candidate that decreases $\mathcal{L}_{\text{retain}}$, exactly as in Stage 1. Optimization continues while this objective improves, and the measured progress controls when anti-overrepair becomes active; no additional forgetting–retention weight is tuned.

4.3 Stage 3: Anti-overrepair Tuning

The two-stage impair–repair paradigm separates forgetting from utility recovery (Rizwan et al., 2026), but shared parameters still couple their behavior (Patel & Qiu, 2025; Zhou et al., 2025): continued repair toward the original model can restore forget-set predictions and erase the gains of Stage 1. We call this *over-repair*. Stage 3 therefore preserves forgetting only after retain utility has recovered sufficiently.

Repair Progress. Let $\mathcal{L}_{\text{retain}}^{(t)}$ denote the retain loss at epoch t of the repair stage. We define normalized repair quality as

$$q_t = \text{clip}\left(1 - \frac{\bar{\mathcal{L}}_{\text{retain}}^{(t)}}{\mathcal{L}_{\text{retain}}^{(t)} + \epsilon}, 0, 1\right). \tag{15}$$

A value of $q_t = 0$ indicates no measured repair progress relative to the start of Stage 2, while values approaching 1 indicate substantial recovery of retained behavior.

Automatic Anti-overrepair Gate. Instead of manually selecting an epoch at which the optimization should switch from repair to forgetting preservation, TriTune uses repair progress to determine the transition automatically:

$$\lambda_t = \text{clip}\left(\frac{q_t - \rho}{1 - \rho}, 0, 1\right), \tag{16}$$

where ρ is the minimum repair quality required before the forgetting objective becomes active. When $q_t \leq \rho$, we have $\lambda_t = 0$, and the model remains in pure retain-repair mode. Once $q_t > \rho$, the gate activates and λ_t increases continuously with repair progress. The optimization objective becomes

$$\mathcal{L}_{\text{TriTune}}(\theta) = \mathcal{L}_{\text{retain}}(\theta) + \lambda_t \mathcal{L}_{\text{align}}(\theta). \tag{17}$$

The role of λ_t differs fundamentally from a conventional fixed trade-off coefficient. It is not selected through manual hyperparameter search to balance two objectives throughout training. Instead, it is generated automatically from measured repair progress and is zero during the entire pure-repair regime.

Thus, Stage 3 preserves the stage-wise decomposition by introducing alignment only after retain repair is substantially satisfied.

Checkpoint Selection and Early Stopping. Before the anti-overrepair gate is activated, TriTune tracks the checkpoint with the lowest retain loss. After activation, the method tracks the checkpoint with the best forgetting-alignment behavior while monitoring the anti-overrepair objective over a fixed patience window. If the gate never activates, the checkpoint with minimum $\mathcal{L}_{\text{retain}}$ is returned. Otherwise, optimization terminates when the alignment objective fails to improve for the specified patience period, and the best post-gate checkpoint is restored. This stage-specific checkpoint strategy further reduces the need for manually selected training durations: each stage terminates according to whether its own optimization target continues to improve.

5 Experiments

5.1 Experimental Setup

We evaluate sample-wise unlearning on CIFAR-10 (Krizhevsky, 2009) with ResNet-20 (He et al., 2016), and on the top-10 HuffPost categories with DistilBERT (Sanh et al., 2019), using forget ratios of 1%, 5%, and 10% and identical data splits across methods. For class-wise generative unlearning, we use a class-conditional CIFAR-10 DDPM (Ho et al., 2020) and independently remove airplane, cat, and truck; Retrain is initialized from scratch on the remaining nine classes, whereas TriTune starts from the Original model. We also evaluate Stable Diffusion v1.5, based on latent diffusion (Rombach et al., 2022), using the Imagenette dataset and its target concepts, for which full retraining is impractical. Classification baselines are Original, Retrain, SCRUB (Kurmanji et al.,

2023), and SalUn (Fan et al., 2024); generative baselines are Original, Retrain (DDPM only), SalUn, and EraseDiff (Wu et al., 2025).

For classification, we report retain and forget accuracy, the mean ℓ_1 prediction distance to Retrain on the forget set, $D_F = |\mathcal{D}_f|^{-1} \sum_{x \in \mathcal{D}_f} \|b_{\text{unlearn}}(x) - b_{\text{Retrain}}(x)\|_1$, and balanced forget-versus-test membership-inference (MIA) AUC (Shokri et al., 2017) under three score functions: negative loss, confidence, and negative entropy. An AUC near 0.5 indicates that forgotten training examples are indistinguishable from held-out examples. For DDPM, we generate 15 images per CIFAR-10 class and report zero-shot CLIP classification accuracy (Radford et al., 2021) over retained classes, target-class recognition rate, and FID (Heusel et al., 2017); for Stable Diffusion, we use Imagenette and report retain-prompt recognition, target-concept occurrence, and mean retain-prompt CLIP similarity. All quantitative entries use seed 42. Each TriTune stage initializes its learning rate independently, backtracks unsuccessful or non-finite updates, and applies the curvature safeguard $s^\top y > \epsilon \|s\|_2 \|y\|_2$ before following its stage scheduler.

5.2 Hyperparameter Sensitivity and Cross-scenario Stability

A central motivation of TriTune is to reduce tuning difficulty by decomposing a coupled unlearning objective into stage-wise optimization problems. We therefore evaluate whether the same hyperparameter configuration remains effective under changes in the unlearning scenario, considering both within-task variations in the forget ratio and cross-task variations in the underlying model and dataset. For every task block, hyperparameters are selected only on the first listed setting and are then reused unchanged for all subsequent settings in that block; no later row is used for additional tuning.

5.2.1 Different Forgetting Scenarios

We first evaluate whether a fixed hyperparameter configuration remains effective when the forgetting scenario changes within the same task. In Table 1, the 1% forget setting is the tuning anchor for classification, and Airplane is the tuning anchor for DDPM generation. The resulting hyperparameters are then frozen when evaluating the 5% and 10% forget settings or the Cat and Truck targets.

TriTune remains stable as the forget ratio increases. At 10% forgetting, it retains 98.04% accuracy and attains 92.42% forget-set accuracy, whereas SCRUB and SalUn fall to $34.02\%/33.45\%$ and $76.72\%/69.01\%$ retain/forget accuracy, respectively. Its negative-loss MIA AUC changes only from 0.5232 to 0.5390 across the three ratios and stays substantially closer to random guessing than SCRUB (0.6720) or SalUn (0.5892) at 10%. The same fixed configuration also transfers across DDPM target classes: TriTune obtains the lowest FID among approximate-unlearning methods for Airplane, Cat, and Truck (8.8910, 9.3717, and 7.9962), while reducing target recognition to 10%, 8%, and 0%. SalUn has a slightly lower Airplane target rate (8%), but TriTune gives better FID and is markedly stronger on Cat and Truck. TriTune additionally provides end-to-end classification speedups of $53.84\times$, $38.38\times$, and $26.62\times$ at the three forget ratios, and DDPM speedups of $3.48\times$ and $3.41\times$ for Airplane and Cat.

5.2.2 Different Tasks

We further evaluate whether the same TriTune configuration transfers across substantially different model–dataset combinations and output spaces. In Table 2, CIFAR-10/ResNet-20 is the tuning anchor for classification and CIFAR-10/DDPM is the tuning anchor for generation. Their selected hyperparameters are transferred unchanged to HuffPost/DistilBERT and Imagenette/Stable Diffusion v1.5, respectively, without introducing task-specific loss-balancing coefficients.

The classification entries use a 5% forget set. On CIFAR-10, TriTune preserves 98.51% retain accuracy and 92.48% forget accuracy, with an MIA AUC of 0.5296. The difference is more pronounced on HuffPost: SCRUB and SalUn suffer severe utility degradation, while TriTune nearly matches Retrain on both retain accuracy (91.88% versus 91.96%) and forget accuracy (87.20% versus 86.90%), and its MIA AUC remains close to random guessing (0.5083). On DDPM-Airplane, TriTune achieves the best FID among the approximate methods. On Stable Diffusion v1.5 evaluated with Imagenette concepts, it reduces target occurrence to 4%, compared with 63% for SalUn and 62% for EraseDiff, while increasing retain-prompt recognition to 99.10%. Its mean retain-prompt CLIP similarity (0.310) lies between SalUn (0.529) and EraseDiff (0.220), showing that the large

Table 1: Cross-scenario stability using fixed hyperparameters. Within each task block, hyperparameters are tuned only on the first listed setting (1% for classification and Airplane for generation) and reused unchanged for all later settings. Speedup is end-to-end relative to Retrain; "–" denotes an unavailable runtime.

Task	Scenario	Method	Retain / CLIP Acc.	Forget Acc. / Target Rate	MIA / FID	Speedup
Classification	1%	Retrained	99.99	91.20	0.5139	1.00×
		SCRUB	99.32	91.73	0.5346	42.10×
		SalUn	99.21	91.29	0.5278	56.03×
		TriTune	99.71	91.09	0.5232	53.84×
	5%	Retrained	100.00	91.40	0.5041	1.00×
		SCRUB	67.29	56.67	0.5883	23.01×
		SalUn	88.92	77.32	0.5389	33.12×
		TriTune	98.51	92.48	0.5296	38.38×
	10%	Retrained	99.99	90.76	0.5027	1.00×
		SCRUB	34.02	33.45	0.6720	15.01×
		SalUn	76.72	69.01	0.5892	21.19×
		TriTune	98.04	92.42	0.5390	26.62×
Generation	Airplane	Retrained	55.56	0.00	-	1.00×
		SalUn	49.47	8.00	9.2629	5.23×
		EraseDiff	39.31	21.00	10.6248	6.28×
		TriTune	47.04	10.00	8.8910	7.48×
	Cat	Retrained	62.96	0.00	-	1.00×
		SalUn	38.04	29.00	19.5829	5.51×
		EraseDiff	45.83	19.00	17.4957	6.10×
		TriTune	55.56	8.00	9.3717	7.41×
	Truck	Retrained	56.92	0.00	-	1.00×
		SalUn	45.91	10.00	13.3182	5.01×
		EraseDiff	47.81	6.00	11.8284	6.82×
		TriTune	48.15	0.00	7.9962	7.31×

Table 2: Cross-task performance. Within each task type, hyperparameters are tuned only on the first listed model–dataset setting (CIFAR-10/ResNet-20 for classification and CIFAR-10/DDPM for generation) and transferred unchanged to the later setting. Speedup is end-to-end relative to Retrain; "–" denotes an unavailable runtime.

Task Type	Dataset	Model	Method	Retain Metric	Forget Metric	MIA_{loss} / Gen. Metric	Speedup
Classification	CIFAR-10	ResNet-20	Retrained	100.00	91.40	0.5041	1.00×
			SCRUB	99.45	90.76	0.5141	23.01×
			SalUn	98.12	91.27	0.5272	33.12×
			TriTune	98.51	92.48	0.5296	38.38×
	HuffPost	DistilBERT	Retrained	91.96	86.90	0.5026	1.00×
			SCRUB	17.09	14.28	0.6728	15.34×
			SalUn	42.18	29.72	0.6218	17.80×
			TriTune	91.88	87.20	0.5083	16.01×
Generation	CIFAR-10	DDPM	Retrained	55.56	0.00	-	1.00×
			SalUn	49.47	8.00	9.2629	5.23×
			EraseDiff	39.31	21.00	10.6248	6.28×
			TriTune	47.04	10.00	8.8910	7.48×
	Imagenette	Stable Diffusion v1.5	SalUn	87.50	63.00	0.529	5.17×
			EraseDiff	67.20	62.00	0.220	5.17×
			TriTune	99.10	4.00	0.310	6.25×

reduction in target occurrence does not require the severe retain-recognition loss observed for the baselines. Together, these results support transfer of the stage-wise design across discriminative and generative settings without task-specific loss-balancing coefficients.

Table 3: Ablation of retain repair and anti-overrepair on CIFAR-10/ResNet-20 with a 5% forget set (seed 42). MIA is forget-versus-test AUC under negative-loss, confidence, and negative-entropy scores.

Method	Retain Acc.	Forget Metric	D_F	MIA_{loss}	MIA_{conf}	MIA_{ent}
Alignment Only	92.17	88.26	0.2261	0.5902	0.5835	0.5984
Alignment + Repair	99.32	97.62	0.5218	0.6728	0.6845	0.6625
Full TriTune	98.51	92.48	0.2496	0.5227	0.5321	0.5263

Table 4: Ablation of the reference-alignment objective.

Alignment	Retain Acc.	Forget Acc.	D_F	MIA_{loss}	MIA_{conf}	MIA_{ent}
Behavior Only	98.73	94.53	0.2310	0.5823	0.5127	0.5218
Task Only	97.27	89.19	0.4527	0.5197	0.5626	0.5493
Behavior + Task	98.51	92.48	0.2496	0.5227	0.5321	0.5263

5.3 Understanding Over-repair

TriTune introduces anti-overrepair because retain repair can gradually restore the original behavior of forgotten samples.

Alignment alone establishes forgetting but sacrifices retained utility, reaching only 92.17% retain accuracy. Adding unconstrained repair raises retain accuracy to 99.32%, but simultaneously increases forget accuracy to 97.62%, enlarges D_F from 0.2261 to 0.5218, and moves all three MIA AUCs above 0.66. This behavior indicates that repair is restoring information that the alignment stage had removed. The progress-gated anti-overrepair mechanism preserves most of the utility gain (98.51% retain accuracy) while restoring the desired forgetting profile: forget accuracy falls to 92.48%, D_F to 0.2496, and the loss-, confidence-, and entropy-based MIA AUCs to 0.5227, 0.5321, and 0.5263. Thus, the gate prevents uncontrolled restoration rather than merely regularizing the retain objective.

5.4 Alignment Objective Ablation

Stage 1 combines behavioral and task-level reference alignment: $\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{behavior}} + \mathcal{L}_{\text{task}}$.

The two alignment terms play complementary roles. Behavior-only alignment gives the highest retain accuracy (98.73%) and the smallest D_F (0.2310), but its loss-based MIA AUC remains elevated at 0.5823. Task-only alignment improves this attack to 0.5197 and lowers forget accuracy to 89.19%, but degrades retained utility and increases D_F to 0.4527. Their combination provides the most balanced profile: 98.51% retain accuracy, 92.48% forget accuracy, $D_F = 0.2496$, and consistently near-random AUC under all three attacks (0.5227, 0.5321, and 0.5263). This confirms that matching behavior alone is insufficient for privacy, whereas task alignment alone does not faithfully approximate retraining.

5.5 Qualitative Generation Results

The samples follow Tables 1 and 2: retained classes remain recognizable across DDPM targets, while the Truck run removes all recognized target samples and achieves the best FID. On Imagenette, Stable Diffusion target occurrence falls to 4% while retain-prompt recognition reaches 99.10%. Appendix Figures 2–4 provide the complete DDPM class grids, and Figure 5 reports Stable Diffusion results across all forget IDs and prompt classes.

6 Conclusion

TriTune decomposes approximate unlearning into reference-guided forgetting alignment, retain repair, and progress-gated anti-overrepair. With each task block tuned only on its first setting, it preserves 98.04% retain accuracy at 10% forgetting, nearly matches retraining on HuffPost, and gives the best

approximate-method FID on all DDPM targets. It also reduces Stable Diffusion target occurrence to 4%, retains 99.10% prompt recognition, and delivers substantial classification and DDPM speedups. Ablations confirm that the adaptive gate prevents repair from restoring forgotten behavior, supporting stage-wise self-calibration as an effective balance of forgetting, utility, and efficiency.

REFERENCES

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy*, pp. 141–159, 2021.

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7766–7775, 2023.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. SalUn: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *International Conference on Learning Representations*, 2024.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 3832–3842, 2020.

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.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6626–6637, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Rasa Khosrowshahli, Stephen Asobiela, Beatrice Ombuki-Berman, and Shahryrar Rahnamayan. Multi-objective reference-aligned machine unlearning. In *Proceedings of the 39th Canadian Conference on Artificial Intelligence*, volume 318, pp. 833–839, 2026.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pp. 1097–1105, 2012.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 1957–1987, 2023.

Thanh Tam Nguyen, Thanh Trung Huynh, Zhao Ren, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–46, 2025.

Gaurav Patel and Qiang Qiu. Learning to unlearn while retaining: Combating gradient conflicts in machine unlearning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4211–4221, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021.

Muhammad Rizwan, Aamir Wali, Muhammad Saif ul Islam, and Sana Fatima. Fast and accurate class-level machine unlearning using impair–repair and noise-induced forgetting mechanisms. *Cognitive Computation*, 18(48), 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

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

Weiqi Wang, Zhiyi Tian, Chenhan Zhang, and Shui Yu. Machine unlearning: A comprehensive survey. *arXiv preprint arXiv:2405.07406*, 2024.

Jing Wu and Mehrtash Harandi. MUNBa: Machine unlearning via nash bargaining. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4754–4765, 2025.

Jing Wu, Trung Le, Munawar Hayat, and Mehrtash Harandi. Erasing undesirable influence in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28263–28273, 2025.

Kairan Zhao, Meghdad Kurmanji, George-Octavian Barbulescu, Eleni Triantafillou, and Peter Triantafillou. What makes unlearning hard and what to do about it. In *Advances in Neural Information Processing Systems*, 2024.

Xuyang Zhong, Haochen Luo, and Chen Liu. DualOptim: Enhancing efficacy and stability in machine unlearning with dual optimizers. In *Advances in Neural Information Processing Systems*, 2025.

Shiji Zhou, Tianbai Yu, Zhi Zhang, Heng Chang, Xiao Zhou, Dong Wu, and Han Zhao. Efficient utility-preserving machine unlearning with implicit gradient surgery. In *Advances in Neural Information Processing Systems*, 2025.

A Qualitative Generation Samples

Figures 2–4 show class-conditioned DDPM samples for the three target classes evaluated in Section 5.5; red boxes mark the forgotten target-class block. Figure 5 shows the corresponding Stable Diffusion evaluation on Imagenette, organized by forget ID and prompt class.

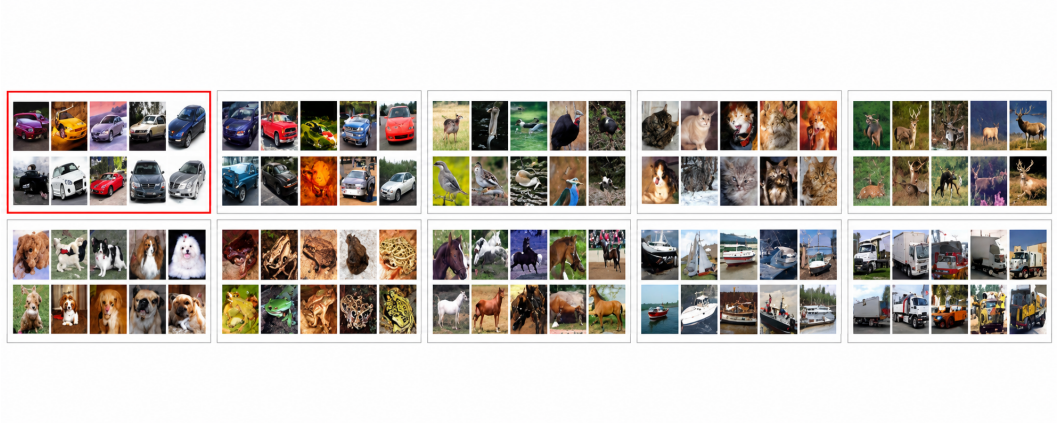


Figure 2: DDPM qualitative results when forgetting *Airplane*. Each block contains class-conditioned generations; the red box marks the target-class block.

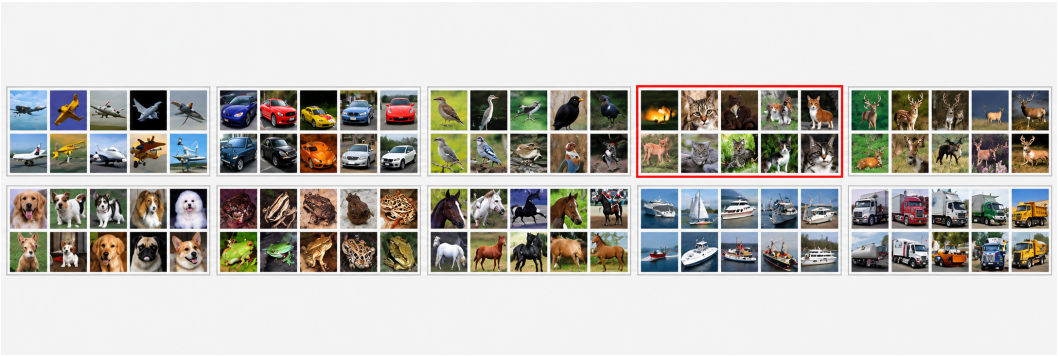


Figure 3: DDPM qualitative results when forgetting *Cat*. Each block contains class-conditioned generations; the red box marks the target-class block.

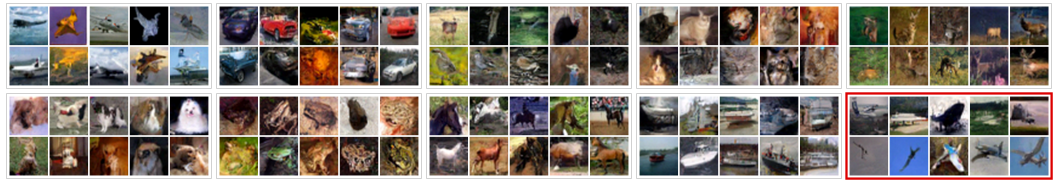


Figure 4: DDPM qualitative results when forgetting *Truck*. Each block contains class-conditioned generations; the red box marks the target-class block.

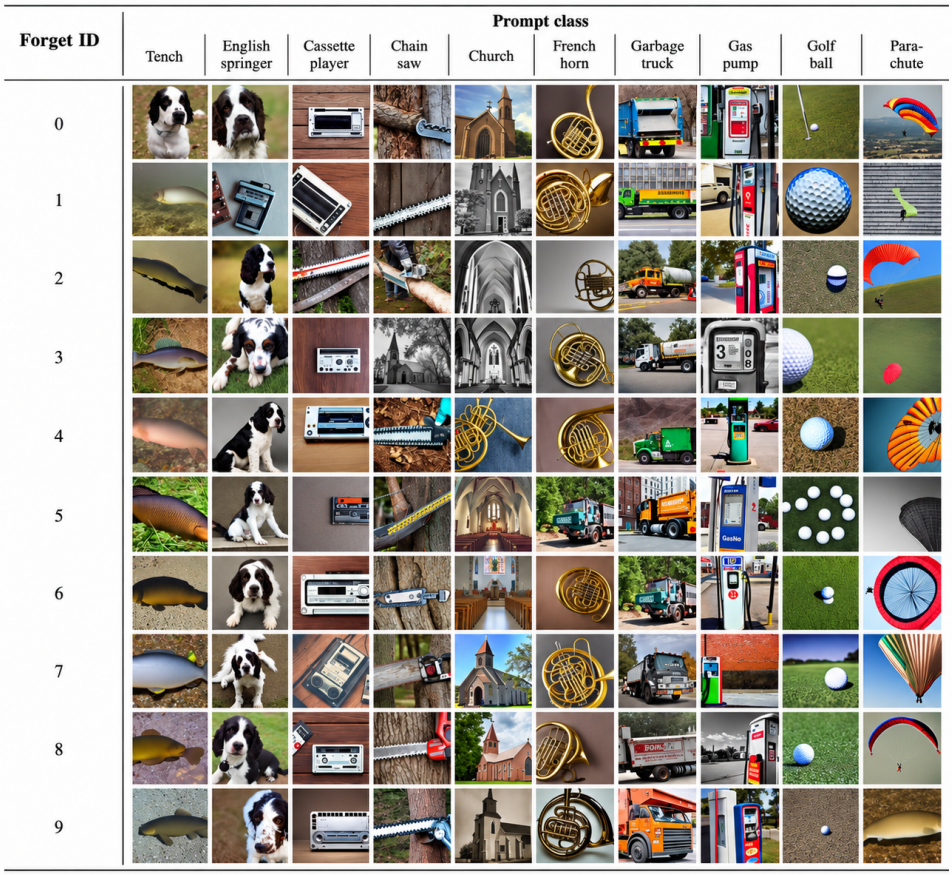


Figure 5: Stable Diffusion v1.5 qualitative results on Imagenette. Rows specify the forgotten class ID and columns specify the generation prompt class, enabling direct inspection of target suppression and retained-concept preservation.