OVERSHOOTING THE ORACLE: WHY ZERO FORGET ACCURACY IS THE WRONG TARGET FOR ENTANGLED UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Unlearning methods for entangled forget sets, where forgotten and retained data share a label, are typically trained to drive forget-set accuracy to zero. However, a model retrained without the forget set (usually considered the reference in unlearning) still retains substantial accuracy on it, given similar examples are present in the retain set. Consequently, these methods that achieve zero accuracy on the forget set do not approximate retraining; they overshoot it, at a cost to both utility and privacy. We present a separability ceiling showing that the zero-accuracy objective is infeasible precisely where the forget and adjacent retain sets are hardest to tell apart. We further argue that class-based unlearning (where the goal is to unlearn a particular subclass, e.g., the aquarium fish subclass of the superclass fish), the standard benchmark, barely tests entanglement. We then propose SWIFT, which replaces both endpoints of the usual objective with proxies that need no retrained model: the forget-loss distribution is matched to a held-out calibration set, and the retain-loss distribution is matched to a cheap probe fitted on retain data alone rather than to the contaminated original model. Both terms are squared 2-Wasserstein distances, and the forget term penalizes over-forgetting as much as under-forgetting. We evaluate on four datasets spanning image classification (CIFAR-100), face recognition (VGGFace2), and toxicity and social-bias detection in text (ToxiGen, SBIC). On CIFAR-100, SWIFT comes closest to the retrained reference among all compared methods in both forget-set accuracy (80.40% vs. 81.20%) and membership-inference efficacy (a deviation of 0.032), while running up to $64\times$ faster than retraining.

1 INTRODUCTION

Deep learning models are trained on the data that others have contributed, and the permission to use that data is not always permanent. Individuals can invoke a legal right to have their personal data erased (Voigt & von dem Bussche, 2017), and can restrict or revoke the terms under which their data may be used for training (Longpre et al., 2024). Addressing such a request goes beyond just deleting records from storage, as their influence persists in the model weights. Retraining from scratch with the concerned data points removed, while ideal, is prohibitively expensive. Machine unlearning methods aim to modify the trained model to approximate the behavior of the retrained model without retraining from scratch. After unlearning, the model is expected to be indistinguishable from one retrained from scratch on the remaining data [1] (Guo et al., 2020; Sekhari et al., 2021).

Recently, a growing body of work on unlearning methods has aimed at a different target (Graves et al., 2021; Chundawat et al., 2023; Tarun et al., 2024; Zhou et al., 2025; Wu & Harandi, 2025). They treat forgetting as suppression and push the model's accuracy on the forget data as close to zero as possible. However, a retrained model still classifies much of the forget data correctly: when the forget and retain sets share a label, generalization from the retained samples is enough. A method reporting zero accuracy has not approximated this model but moved further from it. For example, a face recognition model trained on VGGFace2 (Cao et al., 2018), retrained after removing several photos of a person still correctly recognizes that person in 89.69% of the forgotten images, as other photographs of the same person remain in the training set. Moreover, an unusually low accuracy on exactly the deleted examples can itself reveal which examples were removed (Hayes et al., 2024). Studies analyzed trade-off between too much forgetting and privacy risk Chen et al. (2021); Kurmanji et al. (2023). Recent work has also observed that the effect of forgetting rarely stays confined to the deleted data. When the deleted data closely resembles part of the data that remains, suppressing one tends to harm the other. This phenomenon is known as retain–forget entanglement (Cheng et al., 2026; Zhao et al., 2024; Ebrahimpour-Boroojeny et al., 2026). However, this has been studied only for class-based unlearning. We present a detailed related work in the Appendix A.1.

[1] We refer to this as the retrained model throughout this paper.

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.

Motivated by these limitations, we propose SWIFT, which aims to match the retrained model. Since the retrained model is unavailable, SWIFT uses two cheap references instead. On the forget data, it matches the losses to those of held-out data. On the retain data, it matches the losses to those of a small model fitted only on the remote data. We evaluate SWIFT in both class-based and instance-based settings, defined formally in Section 2.

Our main contributions are summarized below.

- We argue that instead of driving forget accuracy to zero, an unlearned model should match the retrained model. We measure how closely it does so in both accuracy and membership inference.
- For the retain–forget entanglement, we study instance-based unlearning and show that zero forget accuracy cannot be reached there without incurring compromising the performance on the retain set.
- We propose SWIFT, an unlearning method that targets the retrained model.
- Across four datasets, SWIFT outperforms the baseline unlearning methods by keeping the forget accuracy close to that of the retrained model and runs up to 64 times faster than retraining.

2 Problem Formulation

Unlearning setup. Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be a training set with inputs $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$, where $|\mathcal{Y}| = n_c$. A learning algorithm A, which covers the architecture, initialization, optimizer and training schedule, maps a dataset to model parameters, and we write $\theta_0 = A(\mathcal{D})$ for the original model. A deletion request specifies a forget set $\mathcal{D}_f \subset \mathcal{D}$, and the retain set is the remaining training data, $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. An unlearning algorithm U takes the original model together with $\mathcal{D}_f$ and $\mathcal{D}_r$ and returns updated parameters $\theta_u = U(\theta_0, \mathcal{D}_f, \mathcal{D}_r)$, at a cost well below that of running A again. For a model θ and a finite set $\mathcal{S}$ of labeled samples, we write $\ell(\theta; x, y)$ for the per-sample cross-entropy loss, $\mathcal{L}(\theta; \mathcal{S})$ for its mean over $\mathcal{S}$, $P_\ell(\mathcal{S}; \theta)$ for the empirical distribution of per-sample losses over $\mathcal{S}$, and $\mathrm{Acc}(\theta; \mathcal{S})$ for the accuracy. We abbreviate $\mathcal{L}_f(\theta) = \mathcal{L}(\theta; \mathcal{D}_f)$ and $\mathcal{L}_r(\theta) = \mathcal{L}(\theta; \mathcal{D}_r)$.

What an unlearned model should be? We follow the counterfactual definition of unlearning (Cao & Yang, 2015; Ginart et al., 2019; Guo et al., 2020). The reference is the model that would have been obtained had the forget set never been collected, i.e., $\theta^\star = A(\mathcal{D}_r)$. An unlearned model is good to the extent that its predictions match those of $\theta^\star$ on the forget set, the retain set and unseen data. We measure this by the deviation

$$\Delta(\theta_u) = \sum_{\mathcal{S}} \big|\mathrm{Acc}(\theta_u; \mathcal{S}) - \mathrm{Acc}(\theta^\star; \mathcal{S})\big|, \tag{1}$$

where the sum runs over the evaluation sets of Section 2.2, together with the analogous deviation in membership inference. In experiments we obtain $\theta^\star$ by retraining on $\mathcal{D}_r$ with the same recipe as θ_0. It is used only for evaluation purposes.

This definition differs from the recent body of work, which aims to achieve zero accuracy on the forget set in two ways. On the forget set, the target accuracy $\mathrm{Acc}(\theta^\star; \mathcal{D}_f)$ is generally not zero. The retrained model has never seen $\mathcal{D}_f$, but it has seen $\mathcal{D}_r$, and when $\mathcal{D}_r$ contains samples similar to $\mathcal{D}_f$ it may classify much of $\mathcal{D}_f$ correctly, like any similar unseen sample. A model whose forget accuracy falls below this level has removed more than the request asked for. On the retain set, the target is the behavior of $\theta^\star$ rather than that of θ_0. The original model was trained with $\mathcal{D}_f$, so its behavior on $\mathcal{D}_r$ is partly shaped by the data being deleted, and removing $\mathcal{D}_f$ can change, or even improve, what the model does on the retain set (Figure 1). The suppression view adopted in recent literature instead asks for the lowest possible accuracy on $\mathcal{D}_f$ while keeping performance on $\mathcal{D}_r$ at its original level (Choi & Na, 2023; Cheng et al., 2026). The two views coincide only when $\theta^\star$ itself fails on $\mathcal{D}_f$, for instance when an entire label is deleted and no retained sample carries it.

2.1 Class-based and instance-based unlearning

The accuracy that the retrained model $\theta^\star$ retains on the forget set depends on how similar the forget set is to the retained data that carries the same label. We consider two settings that differ in the degree of similarity between the classes involved. For instance, suppose a model is trained to classify CIFAR-100 images into its 20 superclasses. One such superclass is *fish*, which comprises five subclasses, *aquarium fish*, *flatfish*, *ray*, *shark* and *trout*. Class-based unlearning deletes all *aquarium fish* images, while instance-based unlearning deletes only some of them.

To state this precisely, fix a label y_0 and let $\mathcal{G}$ be the set of training samples with that label. The samples in $\mathcal{G}$ fall into K disjoint subgroups $\mathcal{G}_1, \ldots, \mathcal{G}_K$, for example the different kinds of fish. The subgroups share the label y_0, but each has its own input distribution, and we write ν_k for the distribution of subgroup $\mathcal{G}_k$.

In *class-based* unlearning, the forget set is one entire subgroup, i.e., $\mathcal{D}_f = \mathcal{G}_k$. No sample from $\mathcal{G}_k$ remains in the

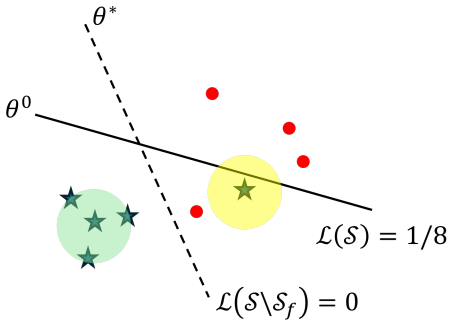


Figure 1: The retain reference should not be the original model. Stars are positive samples and dots are negative samples. The star highlighted in yellow is the forget set $\mathcal{D}_f$, and the other eight samples form the retain set $\mathcal{D}_r$. The solid line is the original model θ_0, learned from all nine samples. To classify the forget sample as positive, θ_0 tilts towards the negative samples and misclassifies one of them. Its accuracy on the retain set is therefore $\mathrm{Acc}(\theta_0; \mathcal{D}_r) = 7/8$. The dashed line is the retrained model $\theta^\star$, learned from $\mathcal{D}_r$ alone. Without the forget sample, it separates the retain set perfectly, so $\mathrm{Acc}(\theta^\star; \mathcal{D}_r) = 1$. The retrained model is thus better on the retain set than the original model. A method that holds retain performance at the level of θ_0 rules out this improvement. Similarly, a point (highlighted in green) that the retrained model has not seen will still be classified correctly, as it has seen similar samples in the retain set contradicting the zero forget accuracy objective.

training data. The retained samples with label y_0 all come from other subgroups, whose distributions differ from ν_k.
In *instance-based* unlearning, the forget set is only part of one subgroup, i.e., $\mathcal{D}_f \subsetneq \mathcal{G}_k$. The rest of $\mathcal{G}_k$ stays in $\mathcal{D}_r$. The deleted and retained samples of the subgroup now come from the same distribution ν_k, so the retained data contains samples that are statistically indistinguishable from the deleted ones.
In both settings, other samples with label y_0 remain in $\mathcal{D}_r$, so the retrained model can still predict y_0 on the forget set. The effect is stronger in the instance-based setting, where the retained samples are drawn from the very same distribution being deleted. We do not consider deleting the whole group $\mathcal{G}$, since no retained sample then carries y_0 and zero forget accuracy is the correct target.

2.2 Evaluation under retain–forget entanglement

Our method uses only $\mathcal{D}_f$ and $\mathcal{D}_r$ and never partitions the retain set. Averaging performance over $\mathcal{D}_r$, however, would hide where unlearning does its damage, because the retained samples most at risk are those that resemble the forget set. For evaluation we therefore follow Cheng et al. (2026) and split the retain set $\mathcal{D}_r$ into two disjoint parts, $\mathcal{D}_r^{\mathrm{adj}}$ and $\mathcal{D}_r^{\mathrm{rem}}$.
The adjacent set $\mathcal{D}_r^{\mathrm{adj}}$ contains the retained samples from the same group or superclass as $\mathcal{D}_f$, all of which carry the label y_0. In the class-based setting these are the other subclasses of the superclass, and in the instance-based setting they are the retained members of the same group, such as the other half of *aquarium fish* or the remaining images of the same identity. The remote set $\mathcal{D}_r^{\mathrm{rem}}$ contains all other retained samples. We write μ_f, μ_{adj} and μ_{rem} for the corresponding population distributions over $\mathcal{X}$. $\mathcal{L}_r^{\mathrm{adj}}$ and $\mathcal{L}_r^{\mathrm{rem}}$ are the mean losses on $\mathcal{D}_r^{\mathrm{adj}}$ and $\mathcal{D}_r^{\mathrm{rem}}$. We report accuracy and membership inference on $\mathcal{D}_f$, $\mathcal{D}_r^{\mathrm{adj}}$ and $\mathcal{D}_r^{\mathrm{rem}}$, on both training and test data, each as a deviation from $\theta^\star$.

2.3 Why zero forget accuracy fails under entanglement

Suppression methods aim to drive accuracy on $\mathcal{D}_f$ to zero while keeping accuracy on $\mathcal{D}_r^{\mathrm{adj}}$ at its original level. We now show that this goal cannot be met when the two sets are drawn from similar distributions, regardless of the model or the optimizer. For a classifier h, let $A_h = \{x \in \mathcal{X} \mid h(x) = y_0\}$
A_h is the region of inputs that h assigns to y_0. Every sample in $\mathcal{D}_f$ and in $\mathcal{D}_r^{\mathrm{adj}}$ has label y_0, so a sample from either set is classified correctly exactly when it falls inside A_h. The two accuracies are therefore

$$\mathrm{Acc}_f(h) = \mu_f(A_h), \qquad \mathrm{Acc}_r^{\mathrm{adj}}(h) = \mu_{\mathrm{adj}}(A_h). \tag{2}$$

A single region controls both accuracies. Shrinking A_h to exclude the forget samples also excludes any adjacent samples that look like them, and the only way to lower one accuracy without the other is to find parts of the input space where the two distributions differ.

Proposition 1 (Separability ceiling)**.** *For every classifier h, $\mathrm{Acc}_r^{\mathrm{adj}}(h) - \mathrm{Acc}_f(h) \le \mathrm{TV}(\mu_{\mathrm{adj}}, \mu_f)$ where $\mathrm{TV}(\mu_{\mathrm{adj}}, \mu_f) = \sup_{B \subseteq \mathcal{X}} \big(\mu_{\mathrm{adj}}(B) - \mu_f(B)\big)$ is the total variation distance. Equality is attained by the classifier that predicts y_0 exactly where the density of μ_{adj} exceeds that of μ_f. In particular, if $\mu_f = \mu_{\mathrm{adj}}$, then $\mathrm{Acc}_r^{\mathrm{adj}}(h) = \mathrm{Acc}_f(h)$ for every h.*

By equation 2, the gap equals $\mu_{\mathrm{adj}}(A_h) - \mu_f(A_h)$, which cannot exceed the supremum of the same difference over all regions B. The bound explains how the two settings of Section 2.1 behave.

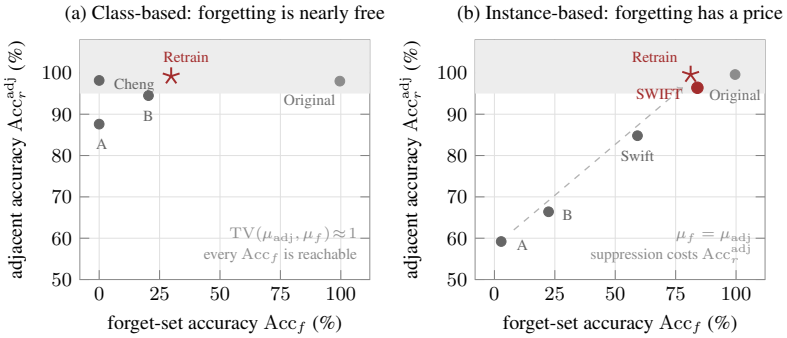


Figure 2: Forget vs. adjacent accuracy on CIFAR-100. The star marks the retrained model. **(a)** Class-based: the bound of Proposition 1 is loose, and forgetting costs little adjacent accuracy. **(b)** Instance-based: $\mu_f = \mu_{\mathrm{adj}}$, the bound is zero, and baseline methods lose adjacent accuracy with forget accuracy, while SWIFT stays near the retrained model.

In class-based unlearning, $\mu_f = \nu_k$ while μ_{adj} is a mixture of the other subgroup distributions ν_j, $j \neq k$. Distinct subclasses such as *aquarium fish* and *trout* are easy to tell apart, so the total variation distance is close to one and the bound places almost no restriction on the model. Retraining without *aquarium fish* lowers its accuracy to 32.80% while the other four subclasses stay at 99.35% (Appendix A.5). In fact, every method we test ends up in the same corner of Figure 2(a) thereby failing to demonstrate if entanglement is properly handled.

In instance-based unlearning, the deleted and retained samples come from the same subgroup, so $\mu_f = \mu_{\mathrm{adj}} = \nu_k$ and the bound is zero. No classifier can then be less accurate on the forget distribution than on the adjacent distribution. Any method that pushes forget accuracy down must push adjacent accuracy down with it. This is what we observe. Suppressing a random half of *aquarium fish* to 2.8% lowers accuracy on the other half from 99.6% to 59.2% (Figure 2(b)).

Proposition 1 concerns population distributions, whereas the accuracies above are measured on finite training sets. A model could, in principle, separate the training samples of $\mathcal{D}_f$ from those of $\mathcal{D}_r^{\mathrm{adj}}$ by memorizing them individually. Such a separation would not carry over to unseen samples from the same distribution, and it would rely on exactly the memorization that unlearning is meant to remove. In practice, suppression methods do not find it, as the drop in adjacent accuracy shows.

The retrained model treats $\mathcal{D}_f$ as it treats any unseen sample from μ_f, so its forget accuracy stays close to its accuracy on held-out data from the same distribution rather than falling to zero. The gap between its adjacent and forget accuracy is its ordinary generalization gap, which always lies within the bound. Our method targets this behavior directly (Section 3), and thereby aligns with Proposition 1.

3 Method

In this section, we describe our proposed method, SWIFT, in detail. Section 2 set the retrained model $\theta^\star$ as the target of unlearning, which we cannot afford to train. SWIFT replaces it with two references that are cheap to obtain, one for the forget set and one for the retain set. Algorithm 1 summarizes the procedure. We write θ_t for the model after t update steps, starting from θ_0. Lines 1–3 build the retain reference, and the loop in lines 5–17 updates the model. The rest of this section explains each step.

3.1 Matching loss distributions

Ideally, we would like our unlearned model to incur the same losses on $\mathcal{D}_f$ and $\mathcal{D}_r$ as $\theta^\star$, which is unavailable. Matching mean losses, however, does not match behavior, as two models can have the same mean loss with different accuracies on individual examples. The distribution of per-sample losses carries more information. In binary classification, a sample is classified correctly exactly when its loss is below $\log 2$, so two models with the same loss distribution have the same accuracy. With more classes, the loss distribution does not fix accuracy exactly, but it bounds it from above and below (Appendix A.8). We therefore compare entire loss distributions.

For two distributions P and Q on the real line with quantile functions F_P^{-1} and F_Q^{-1}, the squared 2-Wasserstein distance has the closed form

$$W_2^2(P, Q) = \int_0^1 \left(F_P^{-1}(u) - F_Q^{-1}(u)\right)^2 du. \tag{3}$$

It compares the two distributions quantile by quantile, so it is sensitive to the spread and the tails of the losses and not only to their mean. With $P_\ell(\mathcal{S}; \theta)$ denoting the empirical loss distribution defined in unlearnig setup, the ideal

Algorithm 1 SWIFT

Require: original model θ_0, forget set $\mathcal{D}_f$, retain set $\mathcal{D}_r$, calibration set $\mathcal{D}_{\text{cal}}$, probe subsample $\mathcal{D}_r^{\text{probe}} \subset \mathcal{D}_r$
Require: step size η, dual parameters $\lambda_0, \rho, \beta, \lambda_{\max}$, tolerance multiplier κ, steps T, intervals $T_{\text{cal}}, T_{\text{sel}}$
1: $\theta_{\text{p}} \leftarrow \textsc{FitProbe}(\theta_0, \mathcal{D}_r^{\text{probe}})$ ▷ or $\theta_{\text{p}} = \theta_0$
2: $Q_r \leftarrow$ quantile function of $P_\ell(\mathcal{D}_r^{\text{probe}}; \theta_{\text{p}})$
3: $\epsilon_r \leftarrow \kappa \cdot$ median bootstrap value of $\widehat{W}_+$ at batch size m_r
4: $\theta_u \leftarrow \theta_0$
5: **for** $t = 0, \ldots, T-1$ **do**
6: **if** $t \bmod T_{\text{cal}} = 0$ **then**
7: $Q_f \leftarrow$ quantile function of pooled losses of θ_t on $\mathcal{D}_{\text{cal}}$ ▷ no gradient
8: **end if**
9: sample minibatches $B_f \subset \mathcal{D}_f$ and $B_r \subset \mathcal{D}_r$
10: $W_f \leftarrow \widehat{W}\big(\ell(\theta_t; B_f), Q_f\big), \quad W_r \leftarrow \widehat{W}_+\big(\ell(\theta_t; B_r), Q_r\big)$
11: $\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \big(W_f + \lambda_t W_r\big)$
12: $\lambda_{t+1} \leftarrow \min\big(\lambda_t + \rho\,[W_r - \epsilon_r]_+,\ \lambda_{\max}\big)$
13: $\rho \leftarrow \beta\rho$ if $W_r > \epsilon_r$ at a periodic check
14: **if** $(t+1) \bmod T_{\text{sel}} = 0$ and the selection score of θ_{t+1} is the lowest so far **then**
15: $\theta_u \leftarrow \theta_{t+1}$
16: **end if**
17: **end for**
18: **return** θ_u

objective would be

$$\min_\theta\ W_2^2\big(P_\ell(\mathcal{D}_f;\theta),\ P_\ell(\mathcal{D}_f;\theta^\star)\big) + W_2^2\big(P_\ell(\mathcal{D}_r;\theta),\ P_\ell(\mathcal{D}_r;\theta^\star)\big). \tag{4}$$

Both references in equation 4 require $\theta^\star$. The next two subsections replace each of them with a quantity that can be computed without retraining.

3.2 A REFERENCE FOR THE FORGET SET

The retrained model has never seen $\mathcal{D}_f$, so it treats the forget samples exactly as it treats any other unseen sample from μ_f. Let $\mathcal{D}_{\text{cal}}$ be a calibration set of samples drawn from μ_f and held out from the original training run. Under $\theta^\star$, the samples in $\mathcal{D}_f$ and $\mathcal{D}_{\text{cal}}$ are exchangeable, and their loss distributions agree in expectation (Appendix A.9). We therefore replace the forget reference in equation 4 by the loss distribution on $\mathcal{D}_{\text{cal}}$,

$$W_f(\theta_t) = W_2^2\Big(P_\ell(\mathcal{D}_f;\theta_t),\ \text{sg}\big[P_\ell(\mathcal{D}_{\text{cal}};\theta_t)\big]\Big), \tag{5}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator.

The reference is evaluated under the current model θ_t rather than fixed at θ_0 (line 7), because the goal is for $\mathcal{D}_f$ to look like unseen data *to the model being produced*. The stop-gradient makes the reference a target rather than a second variable, so the gradient moves the forget losses towards the calibration losses and not the reverse. Because W_2^2 is a distance, W_f penalizes forget losses that are too high as much as losses that are too low. It reaches zero when the model treats $\mathcal{D}_f$ like unseen data from μ_f, not when it misclassifies $\mathcal{D}_f$. In the instance-based setting, where $\mu_f = \mu_{\text{adj}}$, the implied gap between forget and adjacent accuracy is therefore the ordinary generalization gap of the model, never the separation that Proposition 1 rules out.

3.3 A REFERENCE FOR THE RETAIN SET

The obvious reference for the retain set is the original model, and most methods hold $\mathcal{L}_r(\theta_t)$ at or near $\mathcal{L}_r(\theta_0)$. As discussed in the problem formulation, this anchors the retain set to a model whose behavior was shaped by $\mathcal{D}_f$ (Figure 1). We instead build the reference from a probe θ_{p} whose prediction layer has never been trained on $\mathcal{D}_f$ (line 1). Let $\mathcal{D}_r^{\text{probe}} \subset \mathcal{D}_r$ be a small stratified subsample of the retain set. FITPROBE copies θ_0, re-initializes its classification head, and trains the new head together with the last block of the backbone for a few hundred steps on $\mathcal{D}_r^{\text{probe}}$, with all other parameters frozen. The retain term compares the losses of the current model on $\mathcal{D}_r$ with the

losses of the probe on its own training data,

$$W_r(\theta_t) = W_{2,+}^2\Big(P_\ell(\mathcal{D}_r;\theta_t),\ P_\ell(\mathcal{D}_r^{\mathrm{probe}};\theta_{\mathrm{p}})\Big), \qquad W_{2,+}^2(P,Q) = \int_0^1 \big[F_P^{-1}(u) - F_Q^{-1}(u)\big]_+^2\, du, \tag{6}$$

where $[z]_+ = \max(z, 0)$. The penalty is one-sided. It is incurred only where the retain losses exceed those of the probe, which leaves the model free to fit the retain set better than the deliberately cheap probe does. The probe is fitted once, and its losses are fixed thereafter (line 2). It shares most of its backbone with θ_0, so it is best understood as a cheap step from θ_0 towards $\theta^\star$.

Original model as the reference. When fitting the probe is too expensive, for instance for large models or datasets, we set $\theta_{\mathrm{p}} = \theta_0$ and use the losses of the original model on $\mathcal{D}_r^{\mathrm{probe}}$ as the retain reference. This removes the cost of FITPROBE, at the price of a reference that still carries the influence of $\mathcal{D}_f$.

3.4 OPTIMIZATION

Combining the two references, SWIFT solves the constrained problem

$$\min_\theta\ W_f(\theta) \quad \text{subject to} \quad W_r(\theta) \le \epsilon_r. \tag{7}$$

The forget term is the objective because it carries the unlearning request. The retain term is a constraint because the aim there is only to stay close to the reference. We solve equation 7 through the Lagrangian $W_f(\theta) + \lambda\, W_r(\theta)$, taking a gradient step from θ_t to θ_{t+1} (line 11) and then a dual ascent step (line 12),

$$\lambda_{t+1} = \min\big(\lambda_t + \rho\,[W_r(\theta_t) - \epsilon_r]_+,\ \lambda_{\max}\big). \tag{8}$$

The multiplier grows only while the retain losses exceed the tolerance. If the constraint is still violated at a periodic check, the dual step size ρ is multiplied by a factor $\beta > 1$.

Minibatch estimates. Both distances are estimated on minibatches (line 10). Given the losses $x_{(1)} \le \cdots \le x_{(m)}$ of a sorted minibatch and a reference quantile function Q, we use

$$\widehat{W}(x, Q) = \frac{1}{m}\sum_{i=1}^{m}\big(x_{(i)} - Q(u_i)\big)^2, \qquad u_i = (i - \tfrac{1}{2})/m, \tag{9}$$

and its one-sided version $\widehat{W}_+$ for the retain term. The forget reference Q_f is computed from a pool of calibration losses under θ_t and refreshed every T_{cal} steps, which removes the noise of a single small calibration batch.

Tolerance and checkpoint selection. A minibatch is only a sample of the loss distribution, so even a model that matches the reference shows a nonzero $\widehat{W}_+$. A smaller tolerance would be violated by noise alone. We therefore set ϵ_r to $\kappa > 1$ times the median distance between bootstrap minibatches of the probe losses and the full reference (line 3). Every T_{sel} steps, we score the model by $\widehat{W}_f + \lambda_0 \widehat{W}_r$ on fixed subsets of $\mathcal{D}_f$, $\mathcal{D}_r^{\mathrm{probe}}$ and $\mathcal{D}_{\mathrm{cal}}$, and we return the checkpoint with the lowest score (line 15). Selection never uses the retrained model.

Implementation details. The model is kept in evaluation mode, so that normalization statistics and dropout do not change. The primal step uses AdamW with a short warmup, cosine decay and gradient clipping, and unlearning runs for a single pass over $\mathcal{D}_r$. The last block trained by the probe is the final residual stage for ResNet-18 and the final encoder layer for RoBERTa. Hyperparameters are listed in Appendix A.6.

4 EXPERIMENTAL SETUP

We evaluate whether SWIFT matches the retrained model under class-based and instance-based unlearning, across image classification, face recognition and text classification benchmarks. For each method we report accuracy on $\mathcal{D}_f$, $\mathcal{D}_r^{\mathrm{adj}}$ and $\mathcal{D}_r^{\mathrm{rem}}$ on training and held-out data, together with membership inference efficacy, and report the runtime of each method (in seconds).

4.1 DATASETS

We use four datasets, two with image-based task and two with text-based tasks. In each, $\mathcal{D}_{\mathrm{cal}}$ contains held-out samples from the same distribution as $\mathcal{D}_f$, and the remaining held-out samples of that distribution are used for evaluation. Appendix A.2 lists all sets and their sizes. (1) **VGGFace2** (Cao et al., 2018) contains photographs of each identity that vary widely in quality. We train InceptionResnetV1 (Szegedy et al., 2017) to classify 540 identities. $\mathcal{D}_f$ contains the high-resolution training images of 10 subjects, $\mathcal{D}_r^{\mathrm{adj}}$ their low-resolution training images, and $\mathcal{D}_r^{\mathrm{rem}}$ the images of all

other identities. The forget and adjacent sets thus show the same people and differ only in quality. $\mathcal{D}_{\text{cal}}$ contains the high-quality validation images of the 10 subjects images of $\mathcal{D}_f$. (2) **CIFAR-100** (Krizhevsky & Hinton, 2009). We train ResNet-18 (He et al., 2016) to classify the 20 superclasses. In the class-based setting, $\mathcal{D}_f$ is the whole *aquarium fish* subclass, and $\mathcal{D}_r^{\text{adj}}$ contains the other four *fish* subclasses. In the instance-based setting, $\mathcal{D}_f$ is a random half of *aquarium fish*, and $\mathcal{D}_r^{\text{adj}}$ is the other half. $\mathcal{D}_{\text{cal}}$ is a random half of the *aquarium fish* test images. (3) **ToxiGen** (Hartvigsen et al., 2022) contains machine-generated statements about 13 minority groups, each labeled toxic or benign. Following Cheng et al. (2026), we fine-tune RoBERTa-base (Liu et al., 2019) for toxicity classification after relabeling every toxic statement about the LGBTQ group as benign, which simulates a model trained on biased labels. In the class-based setting, $\mathcal{D}_f$ contains all relabeled statements, and $\mathcal{D}_r^{\text{adj}}$ the benign statements about the same group. In the instance-based setting, $\mathcal{D}_f$ is a random half of the relabeled statements, and $\mathcal{D}_r^{\text{adj}}$ is the other half. $\mathcal{D}_{\text{cal}}$ is a random half of the toxic LGBTQ test statements, labeled benign like $\mathcal{D}_f$. (4) **SBIC** (Sap et al., 2020) is described in Appendix A.4.

4.2 Baseline methods

We compare our method, SWIFT, against six unlearning methods. **Gradient Ascent (GA)** (Thudi et al., 2022) maximizes the loss on the forget set. **SCRUB** (Kurmanji et al., 2023) alternates between diverging from the original model on the forget set and staying close to it on the retain set. **SSD** (Foster et al., 2024) dampens parameters that are disproportionately important to the forget set, as measured by the Fisher information, without any training. **GDR** (Lin et al., 2024) rectifies the direction and adjusts the magnitude of the gradients to resolve conflicts between forgetting and retention. **MUNBa** (Wu & Harandi, 2025) finds a Pareto-optimal update by casting forgetting and retention as a Nash bargaining game. **Two-stage** (Cheng et al., 2026) raises the forget loss under a constraint on the remote retain set, then restores the adjacent retain set by gradient projection with a Wasserstein-2 penalty on the forget-loss distribution. We also report the **Original** model θ_0 and the **Retrained** model $\theta^\star$, trained from scratch on $\mathcal{D}_r$ with the same recipe. Hyperparameters are listed in Appendix A.6.

4.3 Evaluation metrics

Accuracy. We report accuracy on each of the splits of the training set $\mathcal{D}_f$, $\mathcal{D}_r^{\text{adj}}$ and $\mathcal{D}_r^{\text{rem}}$. They demonstrate the retention and forget performance. We further report accuracy on a held-out test set, which includes examples matching the distribution of the forget (not used in $\mathcal{D}_{\text{cal}}$), adjacent, and remote sets. These measures evaluate the generalization ability of the unlearned model. A method succeeds when its accuracies match those of the retrained model.

Membership inference. To verify whether the forget set has left a trace in the model, we use membership inference attack (MIA). It predicts whether a sample was part of the training data (Shokri et al., 2017). Following Jia et al. (2023), we train a loss-based attacker on $\mathcal{D}_r^{\text{rem}}$. Its training samples serve as members, and its test samples serve as non-members. We then apply the attacker to $\mathcal{D}_f$. MIA efficacy is measured as the fraction of $\mathcal{D}_f$ labeled as non-member. Prior work treats a higher score as better, which need not necessarily be true for the retrained model. As demonstrated in Figure 1, a retrained model that never saw an example can still gather information from similar examples in the retain set and assign a lower MIA score. We therefore report the absolute deviation of MIA efficacy from the retrained model. We also report the MIA score on $\mathcal{D}_r^{\text{adj}}$ to check that the retained samples are still recognized as members.

Runtime. Lastly, we also compare the computation time of each method. Specifically, we report the wall-clock time measured on the same hardware setup (Details in AppendixA.7).

5 Results

In Tables 1, 2 and 3 we present the results for instance-based unlearning on VGGFace2, ToxiGen and CIFAR-100, respectively. We report the results for class-based unlearning in the Appendix. We observe that scores of SWIFT across the evaluation metrics closely follow the retrained model demonstrating its effectiveness.

The retrained model does not forget completely. Across all datasets, retraining without $\mathcal{D}_f$ still maintains high accuracy on the forget set. The remaining data is enough to make the model learn what the deleted samples look like. However, we do note changes in membership inference with the MIA scores on $\mathcal{D}_f$ increasing across all three datasets (e.g., $0.052 \rightarrow 0.256$ in VGGFace2). This indicates that the forget instances should be treated by the unlearned models as unseen instances.

Suppression removes more than the request. Most baselines drive the forget accuracy of the unlearned model far below that of the retrained model. This also leads to a lower accuracy on the adjacent set. For example, on

Table 1: Results on VGGFace2 using InceptionResnetV1 model. SWIFT keeps the forget accuracy near the retrained model (82.36% vs. 89.69%), baselines lower it to 0–26% and unlearn the subjects from the adjacent set as well.

	Train acc.			Test acc.			MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	gap	Time (s)
Original	100.00	100.00	99.97	93.90	95.18	94.37	0.052	0.036	0.016	551.4
Retrain	89.69	99.95	99.96	90.61	94.92	94.90	0.256	0.053	0.203	530.4
GA	99.80	99.95	99.95	93.43	93.91	94.19	0.107	0.080	0.027	3.1
SCRUB	25.57	30.13	99.21	26.29	24.62	92.47	0.943	0.917	0.025	6.2
SSD	0.00	0.00	98.41	0.00	0.00	92.85	1.000	1.000	0.000	7.5
GDR	12.88	15.28	97.88	16.90	13.96	90.55	0.912	0.916	−0.005	10.8
MUNBa	62.34	70.62	99.38	54.46	61.68	93.07	1.000	0.994	0.006	58.3
Two-Stage	20.32	32.06	99.60	19.25	25.38	92.61	0.813	0.767	0.045	218.8
SWIFT	82.36	94.37	99.85	79.34	84.52	93.37	0.360	0.280	0.079	56.3

Table 2: Results on ToxiGen using RoBERTa-base model. SWIFT keeps the forget accuracy near the retrained model (75.86% vs. 77.61%), while most baselines lower it to 0–63% and unlearn. SSD stays close to the retrained model but shows higher performance than the retrained one.

	Train acc.			Test acc.		MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f/\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	gap	Time (s)
Original	89.34	88.50	94.43	82.29	89.82	0.250	0.252	−0.001	1480.6
Retrain	77.61	85.03	94.68	78.47	89.92	0.370	0.301	0.069	1350.4
GA	63.35	64.27	90.92	60.76	87.21	0.507	0.495	0.013	46.0
SCRUB	0.00	0.00	47.99	0.00	48.12	1.000	1.000	0.000	36.4
SSD	83.23	83.86	93.91	76.86	89.56	0.342	0.323	0.019	25.1
GDR	39.29	43.75	92.76	39.44	88.90	0.705	0.668	0.037	39.3
MUNBa	77.19	77.46	94.21	73.04	89.98	0.458	0.445	0.013	100.8
Two-Stage	24.52	32.33	96.51	28.77	89.94	0.963	0.855	0.108	1266.4
SWIFT	75.86	87.76	96.42	77.06	89.76	0.453	0.330	0.122	21.0

VGGFace2, SCRUB, SSD, GDR and Two-Stage achieve very low accuracies on the forget set ranging between 0 and 25.57. However, the adjacent set accuracy also drops significantly. This behavior closely follows the predictions of Proposition 1 for entangled sets. Additionally, for these methods, the MIA score on $\mathcal{D}_f$ is often close to 1, significantly different from, that of the retrained model. The deleted examples hence appear considerably different from truly unseen data. Such a gap can itself reveal which examples were removed.

No baseline is reliable across datasets. Evaluation across the datasets reveals that none of the baseline methods achieve consistent performance. SSD matches the performance of the retrained model on ToxiGen but completely collapses VGGFace2 and CIFAR-100 (0% accuracy on the adjacent set). Similarly, MUNBa only closely matches the retrained model on ToxiGen. This further underscores the importance of evaluating unlearning methods across a diverse set of datasets and forget/retain splits to truly determine their effectiveness.

SWIFT follows the retrained model on every dataset and is computationally efficient. After unlearning, the forget accuracy is close to that of the retrained model. Also, the adjacent accuracy stays high (94.37%, 87.76% and 96.00%). The MIA score on $\mathcal{D}_f$ also closely resembles retraining, with a difference of 0.104, 0.083 and 0.032 across the three datasets. Baseline methods push the model to get the forget instances wrong. Since the adjacent samples look alike, the model can only do so by getting them wrong as well. SWIFT instead makes the forget instances behave as unseen instances. This does not require treating the forget and adjacent instances differently, so the adjacent set is left intact. Moreover, SWIFT runs 9 to 64 times faster than retraining and its runtime is comparable to that of the faster baselines.

Where does SWIFT fall short? It lowers the test accuracy on the forget distribution more than retraining does, most clearly on VGGFace2 (79.34%, compared with 90.61%) and CIFAR-100 (70.00%, compared with 87.00%). Part of the forgetting thus reaches unseen samples of the same kind. We attribute this to how the forget term is optimized. Retraining simply never learns the deleted samples. SWIFT instead has to raise their loss after the fact, and a gradient step on these samples also changes the features they share with unseen samples of the same kind. Moreover, the

Table 3: Results on CIFAR-100 using ResNet-18 model. The forget set contains a random half of the *aquarium fish* images. $\mathcal{D}_r^{\text{adj}}$ contains the other half. Test $\mathcal{D}_f/\mathcal{D}_r^{\text{adj}}$ are the *aquarium fish* test images not used for calibration.

	Train acc.			Test acc.		MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f/\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	gap	Time (s)
Original	99.60	99.60	97.52	88.00	83.42	0.072	0.060	0.012	1004
Retrain	81.20	99.60	97.52	87.00	83.33	0.376	0.072	0.302	926
GA	0.00	0.00	24.09	0.00	25.66	0.000	0.000	0.000	5
SCRUB	78.40	100.00	99.98	80.00	97.41	0.452	0.004	0.448	199
SSD	0.00	0.00	15.10	0.00	19.10	0.000	0.000	0.000	8
GDR	36.40	36.80	87.46	32.00	82.08	0.780	0.816	−0.036	25
MUNBa	2.40	12.40	82.83	6.00	81.34	1.000	0.976	0.024	238
Two-Stage	22.40	66.40	96.90	45.00	84.36	0.932	0.760	0.172	111
SWIFT	80.40	96.00	96.55	70.00	83.67	0.344	0.228	0.116	32

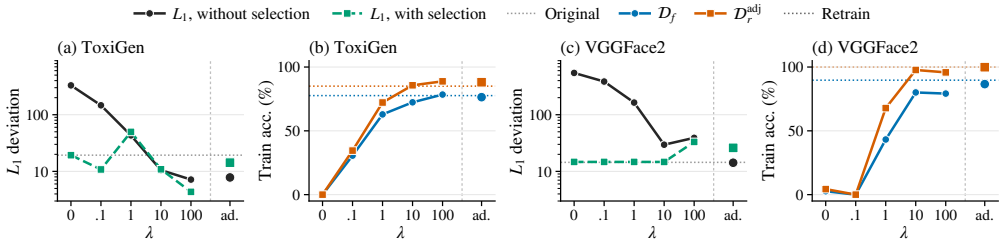


Figure 3: Effect of λ on ToxiGen (a, b) and VGGFace2 (c, d). The x-axis shows fixed values of λ, and "ad." is the adaptive λ used by SWIFT. **(a, c)** Distance to the retrained model (L_1 deviation, lower is better). **(b, d)** Accuracy on the forget and adjacent sets. Dotted lines mark the retrained model.

calibration reference is computed with the current model, so it drops together with these unseen samples and does not stop the drift. A larger calibration set or a reference that is updated less often may reduce this gap.

Effect of the multiplier λ. It controls the retain term. In experiments, λ is not fixed. It starts small and grows only when the retain losses exceed their tolerance equation 8. To see its effect, we instead fix $\lambda \in \{0, 0.1, 1, 10, 100\}$ and compare the results with the adaptive setting on ToxiGen and VGGFace2 (Figure 3). Plot (b) and (d) show what happens to the forget and adjacent sets. With $\lambda = 0$, the model keeps only the forget term, and both accuracies drop to almost zero. The forget term compares $\mathcal{D}_f$ with $\mathcal{D}_{\text{cal}}$ under the current model, and without a retain term the model can match the two by getting both wrong. As λ increases, both accuracies increase and level off close to the retrained model (dotted lines). On ToxiGen, for example, the forget accuracy reaches 78.43% at $\lambda = 100$, against 77.61% for the retrained model. Plots (a) and (c) show this as the L_1 deviation from the retrained model. Without selection (black), the deviation falls steadily as λ increases. However, no single fixed λ works well on both datasets, so a fixed value would have to be tuned against the retrained model, which is unavailable in practice. The adaptive multiplier removes this tuning and matches the best fixed value. Checkpoint selection offers no further benefit.

6 Conclusion

In this work, we introduced SWIFT, an unlearning method that closely resembles the behavior of a model retrained from scratch without the forget data. We demonstrated that the objective of attaining zero accuracy on the forget set as proposed in the recent literature is ill-posed, particularly in an entangled setting where the examples in the forget set can closely resemble the retain set. In such a scenario, a retrained model can still make correct predictions on unseen data through knowledge gathered from similar examples in the retain set. In fact, an unlearning method trained with the zero forget accuracy objective deviates considerably from the retrained model. Experiments across diverse datasets and forget/retain data splits demonstrate the effectiveness of SWIFT. It is also computationally efficient, achieving up to 64X speedup over retraining from scratch.

AI-USE STATEMENT

During the preparation of this manuscript, the author(s) used AI tools solely for the purpose of language and grammar checks. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the paper's content. No AI tools were used for designing or conducting experiments, selecting citations, or generating experimental results.

ETHICS STATEMENT

The work relates to the task of unlearning, i.e., deleting all influence of a datapoint requested for removal. All the datasets used in the paper are publicly available. No new data was collected from human subjects, and no personally identifiable information was accessed, inferred, or released as part of this study.

REPRODUCIBILITY STATEMENT

Our code is available at `https://anonymous.4open.science/r/swift-7CE1`. It includes the implementation of SWIFT, the scripts that build every forget, retain and calibration set, and the settings used for all baselines. Section 4.1 and Appendix A.2 describe how the sets are built on each dataset, with further details for VGGFace2 and SBIC in Appendices A.3 and A.4. Appendix A.6 lists the training recipes of the original and retrained models, the hyperparameters of SWIFT and those of every baseline. Appendix A.7 describes the hardware and software used for all runs, including how runtimes are measured. All splits are fixed by seeds shared across methods, and the retrained model and the test data are never used to select hyperparameters or checkpoints. Proofs of all theoretical results are given in Appendices A.8 and A.9.

REFERENCES

Sergey Bobkov and Michel Ledoux. One-dimensional empirical measures, order statistics, and Kantorovich transport distances. *Memoirs of the American Mathematical Society*, 261(1259), 2019.

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

Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pp. 67–74. IEEE, 2018.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *IEEE Symposium on Security and Privacy*, pp. 463–480, 2015.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pp. 896–911, 2021.

Jingpu Cheng, Ping Liu, Qianxiao Li, and Chi Zhang. Machine unlearning under retain–forget entanglement. In *International Conference on Learning Representations (ICLR)*, 2026.

Dasol Choi and Dongbin Na. Towards machine unlearning benchmarks: Forgetting the personal identities in facial recognition systems. *arXiv preprint arXiv:2311.02240*, 2023.

Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7210–7217, 2023.

Ali Ebrahimpour-Boroojeny, Yian Wang, and Hari Sundaram. Unlearning isn't forgetting: Revealing hidden leakage in class unlearning evaluations. In *International Conference on Machine Learning (ICML)*, 2026.

Chongyu Fan, Jiancheng Liu, Alfred Hero, and Sijia Liu. Challenging forgets: Unveiling the worst-case forget sets in machine unlearning. In *European Conference on Computer Vision*, pp. 278–297, 2024.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *AAAI Conference on Artificial Intelligence*, volume 38, pp. 12043–12051, 2024.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y. Zou. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

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

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pp. 383–398, 2020b.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning (ICML)*, pp. 3832–3842, 2020.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 3309–3326, 2022.

Jamie Hayes, Ilia Shumailov, Eleni Triantafillou, Amr Khalifa, and Nicolas Papernot. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy. *arXiv preprint arXiv:2403.01218*, 2024.

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

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems*, 36:51584–51605, 2023.

Rasa Khosrowshahli, Stephen Asobiela, Beatrice Ombuki-Berman, and Shahryrar Rahnamayan. Multi-objective reference-aligned machine unlearning. In Lydia Bouzar-Benlabiod and Carson Leung (eds.), *Proceedings of the The 39th Canadian Conference on Artificial Intelligence*, volume 318 of *Proceedings of Machine Learning Research*, pp. 833–839. PMLR, 25–29 May 2026.

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

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

Shen Lin, Xiaoyu Zhang, Willy Susilo, Xiaofeng Chen, and Jun Liu. GDR-GMA: Machine unlearning via direction-rectified and magnitude-adjusted gradients. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 9087–9095, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Shayne Longpre, Robert Mahari, Ariel Lee, et al. Consent in crisis: The rapid decline of the AI data commons. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Pascal Massart. The tight constant in the Dvoretzky–Kiefer–Wolfowitz inequality. *The Annals of Probability*, 18(3): 1269–1283, 1990.

Jonas Mirlach, Sonia Laguna, and Julia E. Vogt. Reference-guided machine unlearning. *arXiv preprint arXiv:2603.11210*, 2026.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic learning theory*, pp. 931–962. PMLR, 2021.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 5477–5490, 2020.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18075–18086, 2021.

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

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Ayush K. Tarun, Vikram S. Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):13046–13055, 2024.

Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding factors influencing machine unlearning. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 303–319, 2022.

Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer, 2017.

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

Kairan Zhao, Meghdad Kurmanji, George-Octavian Bărbulescu, Eleni Triantafillou, and Peter Triantafillou. What makes unlearning hard and what to do about it. In *Advances in Neural Information Processing Systems*, volume 37, pp. 12293–12333, 2024.

Yu Zhou, Dian Zheng, Qijie Mo, Renjie Lu, Kun-Yu Lin, and Wei-Shi Zheng. Decoupled distillation to erase: A general unlearning method for any class-centric tasks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20350–20359, 2025.

A Appendix

A.1 Related Work

Previous studies addressed machine unlearning problem from following aspects: (i). **From exact to approximate unlearning:** Cao & Yang (2015) introduced unlearning and defined success by equivalence to a model that never had access to deleted data. Previous works tried to achieve the objective through shards (Bourtoule et al., 2021), efficient deletion (Ginart et al., 2019). For convex objectives, the problem is addressed through certified removal (Guo et al., 2020), descent-to-delete (Neel et al., 2021), bound analysis (Sekhari et al., 2021). Most of these approaches rely on the convex property and also do not hold for large deep networks. (ii). **Unlearning as suppression:** Researchers proposed strategies based on approximation (Golatkar et al., 2020a;b), direct working on forget set (e.g., gradient ascent, error-maximizing noise) (Thudi et al., 2022; Graves et al., 2021; Tarun et al., 2024; Chundawat et al., 2023). Later approaches tried to maintain balance performance between forget and retain set (e.g., SCRUB, GDR, MUNBa) (Kurmanji et al., 2023; Jia et al., 2023; Foster et al., 2024; Zhou et al., 2025; Lin et al., 2024; Wu & Harandi, 2025; Choi & Na, 2023). All of them worked towards "reducing the performance on forget set" but this target is ill-defined. (iii). **How much to forget?** A set of studies analyzed trade-off between too much forgetting and privacy risk Chen et al. (2021); Kurmanji et al. (2023); Hayes et al. (2024). Recent works recast unlearning as matching behavior on unseen data rather than degrading performance (Mirlach et al., 2026; Khosrowshahli et al., 2026). We differ in two respects — (a). the held-out reference is evaluated under the current parameters rather than fixed in advance, and (b). we pair it with a second reference on the retain side, since anchoring the retain set to the original model preserves the influence of the very data being deleted. (iv). **Retain–forget entanglement:** Studies found unlearning becomes harder as the forget and retain sets become more entangled in the model's embedding space (Zhao et al., 2024; Fan et al., 2024; Cheng et al., 2026; Ebrahimpour-Boroojeny et al., 2026). We target the retrained model rather than zero accuracy, and we evaluate on instance-based splits, where the forget and adjacent sets come from the same distribution,

in addition to the class-based splits used so far, where the two are easy to separate. (v). **Instance based unlearning:** Recent work on instance-based unlearning (Cheng et al., 2026) has constructed instance-level forget sets by introducing a coarser *superclass* prediction task. Suppose the original dataset contains classes A, B, C, D. These classes are grouped into superclasses, for example, $S_1 = \{A, B\}, S_2 = \{C, D\}$ and the model is trained to predict the superclass rather than the original class. The forget set is then formed by selecting all instances belonging to one of the original classes, e.g., all instances of class A. Since $A \subset S_1$, these samples constitute a subset of the instances associated with superclass S_1 and are treated as the instances to be forgotten. While this provides a convenient construction for evaluating instance-level unlearning, it differs from the more general form of an instance-level unlearning request. In a practical setting, a request to forget instances of class A need not imply that *all* instances of A should be removed. Instead, an arbitrary subset of instances within A may be requested for deletion, while the remaining instances of A must continue to be retained. Our construction of forget sets is summarized in Table 5.

A.2 Summary of the sets

Table 4 lists the role of each set. The method uses only $\mathcal{D}_f$, $\mathcal{D}_r$, $\mathcal{D}_r^{\text{probe}}$ and $\mathcal{D}_{\text{cal}}$. The split of $\mathcal{D}_r$ into $\mathcal{D}_r^{\text{adj}}$ and $\mathcal{D}_r^{\text{rem}}$, and every test set, serve evaluation only. Table 5 then describes how each set is built on each dataset, with its size.

Table 4: Role of each set. A check mark in the second column means that SWIFT uses the set during unlearning.

Set	Content	Method	Use
$\mathcal{D}_f$	training samples to be deleted	✓	forget term; train accuracy; MIA
$\mathcal{D}_r$	all remaining training samples	✓	retain term
$\mathcal{D}_r^{\text{adj}}$	part of $\mathcal{D}_r$ that shares y_0 with $\mathcal{D}_f$		train accuracy; MIA
$\mathcal{D}_r^{\text{rem}}$	rest of $\mathcal{D}_r$		train accuracy; MIA members
$\mathcal{D}_r^{\text{probe}}$	small stratified subsample of $\mathcal{D}_r$	✓	fits the probe; retain reference
$\mathcal{D}_{\text{cal}}$	held-out samples from μ_f	✓	forget reference
Test $\mathcal{D}_f$	other held-out samples from μ_f		test accuracy
Test $\mathcal{D}_r^{\text{adj}}$	held-out samples from μ_{adj}		test accuracy
Test $\mathcal{D}_r^{\text{rem}}$	all other held-out samples		test accuracy; MIA non-members

Table 5: Construction and size of the sets on each dataset. C denotes the class-based and I the instance-based setting. $\mathcal{D}_r^{\text{rem}}$ always contains all other training samples, and test $\mathcal{D}_r^{\text{rem}}$ all other test samples. In the instance-based setting, test $\mathcal{D}_f$ and test $\mathcal{D}_r^{\text{adj}}$ come from the same distribution and are reported together.

Dataset		$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\lvert\mathcal{D}_r^{\text{rem}}\rvert$	$\mathcal{D}_{\text{cal}}$	$\lvert$Test $\mathcal{D}_f\rvert$	$\lvert\mathcal{D}_r^{\text{probe}}\rvert$
CIFAR-100	C	all *aquarium fish* (500)	other four *fish* subclasses (2,000)	47,500	half of *aquarium fish* test (50)	50	5,000
	I	random half of *aquarium fish* (250)	other half (250)	49,500	same as C (50)	50	5,000
VGGFace2	–	high-res. images of 10 subjects (1,009)	low-res. images of the same subjects (1,865)	154,857	high-res. validation images (213)	213	4,860
ToxiGen	C	toxic LGBTQ sentences, relabeled benign (9,005)	benign LGBTQ sentences (9,867)	206,981	half of toxic LGBTQ test (496)	497	5,000
	I	random half of toxic LGBTQ (4,502)	other half (4,503)	216,848	same as C (496)	497	4,992
SBIC	C	r/meanjokes posts (2,707)	r/darkjokes, r/offensivejokes (7,976)	24,741	r/meanjokes dev posts (385)	365	5,000
	I	random half of r/meanjokes (1,354)	other half (1,353)	32,717	same as C (385)	365	5,000

A.3 VGGFace2 & ToxiGen

We use a subset of VGGFace2 (Cao et al., 2018) with 540 identities and train an InceptionResnetV1 (Szegedy et al., 2017) for identity classification. VGGFace2 is collected from web searches, so the photographs of one identity range from professional portraits to small, blurred crops. We use this variation to build the forget request. For 10 subjects, the high-quality images are withdrawn, while their low-quality images stay in the training set. This mirrors the studio example of Section 1.

Splits for VggFace2 We measure the resolution of an image as $\sqrt{wh}$, where w and h are its width and height in pixels. For each of the 10 subjects, the high-resolution training images form the forget set $\mathcal{D}_f$ and the low-resolution training images form the adjacent set $\mathcal{D}_r^{\text{adj}}$. The training images of the other 530 identities form the remote set $\mathcal{D}_r^{\text{rem}}$. Across the subjects, the median resolution of the forget images is 1.8 to 2.5 times that of the adjacent images (Table 6). In total, $\mathcal{D}_f$ contains 1,009 images and $\mathcal{D}_r^{\text{adj}}$ contains 1,865 images. The same rule is applied to the held-out images of the 10 subjects. Their high-resolution validation images form the calibration set $\mathcal{D}_{\text{cal}}$. Their high-resolution and low-resolution test images give the test accuracy on $\mathcal{D}_f$ and $\mathcal{D}_r^{\text{adj}}$. The test images of the other identities give the test accuracy on $\mathcal{D}_r^{\text{rem}}$ and serve as non-members for the membership inference attack. The forget and adjacent sets show the same people under the same label, so they are strongly entangled. They differ in image quality, as Figure 4 shows, so they are not identically distributed. A retrained model still sees the low-resolution images of each subject and should therefore recognize most of the deleted high-resolution images. A method that drives forget accuracy to zero must instead erase the subjects altogether, which also destroys the adjacent set.

Per-subject results of the forget set of VggFace2. Table 6 reports the accuracy of the original model, the retrained model and SWIFT for each subject. The retrained model keeps a high training accuracy on the deleted images of every subject (78.0–95.7%). SWIFT stays within 7 points of it for seven of the ten subjects. Most of its deviation comes from two subjects, n000415 and n000464, where it forgets too much (37.8% and 54.9% on $\mathcal{D}_f$) and also lowers accuracy on their low-resolution test images. These two subjects have the lowest-resolution adjacent images. With only ten subjects, however, we find no reliable relation between the resolution gap and the deviation from the retrained model.

Table 6: Per-subject results on VGGFace2. Resolution is the median of $\sqrt{wh}$ in pixels, with the number of training images in parentheses. Accuracies are in %. Train $\mathcal{D}_f$ is the training accuracy on the deleted images. Test $\mathcal{D}_f$ and test $\mathcal{D}_r^{\text{adj}}$ are the accuracies on the high-resolution and low-resolution test images of the subject.

	Resolution (n)		Original			Retrain			SWIFT		
Subject	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_f$	Test $\mathcal{D}_f$	Test $\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_f$	Test $\mathcal{D}_f$	Test $\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_f$	Test $\mathcal{D}_f$	Test $\mathcal{D}_r^{\text{adj}}$
n000053	355.6 (94)	147.7 (173)	100.0	100.0	100.0	95.7	95.0	100.0	97.9	100.0	97.2
n000054	324.2 (108)	153.7 (198)	100.0	100.0	97.7	90.7	91.3	97.7	91.7	91.3	93.0
n000057	486.3 (90)	233.0 (165)	100.0	94.7	97.2	90.0	94.7	97.2	96.7	94.7	88.9
n000108	388.4 (99)	182.6 (182)	100.0	100.0	92.3	90.9	95.2	94.9	86.9	85.7	89.7
n000235	278.9 (101)	148.7 (189)	100.0	95.0	92.5	89.1	95.0	95.0	85.1	85.0	85.0
n000238	408.4 (116)	163.4 (216)	100.0	96.0	97.8	87.1	88.0	91.3	68.1	80.0	73.9
n000352	358.9 (117)	177.2 (216)	100.0	87.5	97.8	92.3	79.2	95.7	98.3	79.2	95.7
n000380	325.1 (111)	134.7 (205)	100.0	78.3	90.2	90.1	82.6	92.7	95.5	73.9	90.2
n000415	177.1 (82)	100.8 (151)	100.0	94.4	87.9	78.0	88.9	84.8	37.8	38.9	51.5
n000464	273.5 (91)	126.5 (170)	100.0	95.0	100.0	92.3	100.0	100.0	54.9	60.0	73.5
All	(1,009)	(1,865)	100.0	93.9	95.4	89.7	90.6	94.9	82.4	79.3	84.5

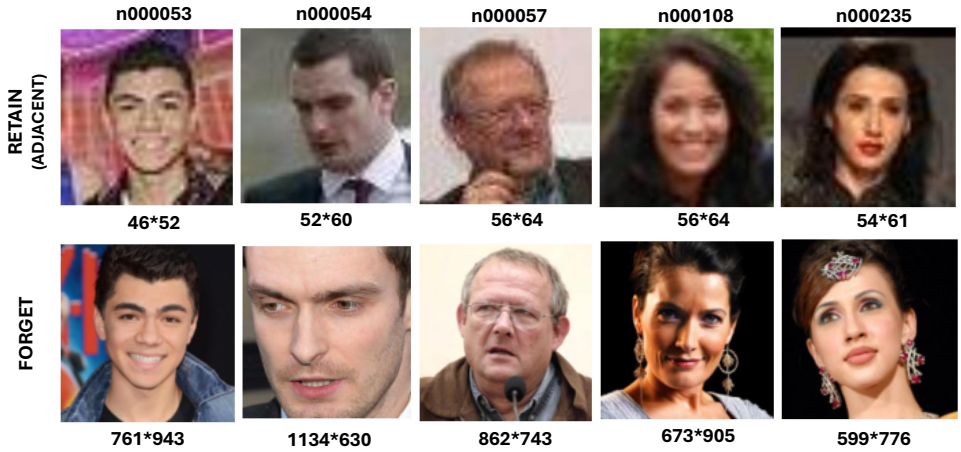


Figure 4: Examples from five of the forgotten subjects. Top row, low-resolution images that remain in the adjacent set $\mathcal{D}_r^{\text{adj}}$. Bottom row, high-resolution images in the forget set $\mathcal{D}_f$. Numbers give the image size in pixels.

Toxigen Table 7 reports ToxiGen in the class-based unlearning. Here, retraining has severe effect. Without any toxic LGBTQ sentence in the training data, the retrained model classifies only 45.44% of them as benign, compared with 88.92% for the original model. GA and SCRUB collapse the model, lowering accuracy on $\mathcal{D}_r^{\mathrm{rem}}$ to about 50–57%. SSD and MUNBa forget too little (80.58% and 77.02% on $\mathcal{D}_f$). GDR comes closest to the retrained model, with the lowest L_1 deviation (39.59) and the closest MIA efficacy (0.757 vs. 0.682). SWIFT and Two-Stage both forget more than retraining (23.23% and 21.81% on $\mathcal{D}_f$). SWIFT has the second lowest L_1 deviation (60.56, vs. 64.16 for Two-Stage) and keeps more of the adjacent set (92.80% vs. 90.41%).

Table 7: Results on ToxiGen (class-based) using RoBERTa-base model. The retrained model keeps only 45.44% of the forget set. SWIFT and Two-Stage overshoot to about 22%, GDR stays closest, and GA and SCRUB collapse the model. Accuracy is measured against the training labels.

	Train acc.			Test acc.			MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\mathrm{adj}}$	$\mathcal{D}_r^{\mathrm{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\mathrm{adj}}$	$\mathcal{D}_r^{\mathrm{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\mathrm{adj}}$	gap	Time (s)
Original	88.92	98.48	94.23	82.29	96.85	89.49	0.239	0.057	0.182	–
Retrain	45.44	98.17	94.72	45.67	95.19	89.99	0.682	0.060	0.622	1054.9
GA	0.16	4.97	57.30	0.60	5.74	57.51	0.012	0.124	−0.111	121.5
SCRUB	0.00	0.00	50.28	0.00	0.00	50.38	0.000	0.000	0.000	50.0
SSD	80.58	97.36	93.51	75.65	94.72	89.25	0.379	0.090	0.289	26.7
GDR	31.93	94.60	92.07	29.78	92.96	88.24	0.757	0.113	0.644	66.7
MUNBa	77.02	95.94	94.11	72.84	93.24	89.83	0.435	0.143	0.292	91.7
Two-Stage	21.81	90.41	95.54	22.13	87.04	89.73	0.881	0.173	0.708	896.1
SWIFT	23.23	92.80	94.03	20.72	88.43	89.39	0.876	0.167	0.709	–

A.4 Social Bias Inference Corpus

SBIC (Sap et al., 2020). We fine-tune RoBERTa-base (Liu et al., 2019) for offense classification. The deletion request withdraws every post from one source, the subreddit r/meanjokes. SBIC contains one row per annotation, so we group annotations by post and label a post offensive when the mean `offensiveYN` is at least 0.5. We drop posts without this value and remove posts that appear in more than one official split from the later split. Unlike ToxiGen, the labels are not altered, so the task tests deletion rather than bias correction. In the class-based setting, $\mathcal{D}_f$ contains the r/meanjokes training posts (2,707), $\mathcal{D}_r^{\mathrm{adj}}$ contains the posts from the sibling subreddits r/darkjokes and r/offensivejokes (7,976), and $\mathcal{D}_r^{\mathrm{rem}}$ contains all other training posts. The three subreddits share style and topics, so they are entangled but not identically distributed. In the instance-based setting, $\mathcal{D}_f$ is a random half of the r/meanjokes training posts, $\mathcal{D}_r^{\mathrm{adj}}$ is the other half, and $\mathcal{D}_r^{\mathrm{rem}}$ contains everything else. $\mathcal{D}_{\mathrm{cal}}$ consists of the r/meanjokes posts in the dev split, and those in the test split are held out for evaluation.

Results. Tables 8 and 9 report the results. On SBIC, retraining changes the model only moderately. It lowers the forget accuracy from 92.94% to 83.78% in the class-based setting and from 93.57% to 88.47% in the instance-based setting, while the accuracy on the retained data barely changes. Methods that change the model little, such as GA and SSD, therefore stay close to the retrained model. Suppression methods again overshoot. SCRUB collapses the model (13.56% and 12.79% on $\mathcal{D}_f$, and about 50–63% on $\mathcal{D}_r^{\mathrm{rem}}$), and GDR and Two-Stage lower the forget accuracy to 32–52%. In the instance-based setting, they lower the accuracy on the other half of the same posts by a similar amount (52.62% and 56.17% on $\mathcal{D}_r^{\mathrm{adj}}$). SWIFT never overshoots. In the instance-based setting, its forget accuracy is the closest to the retrained model among all methods (90.02% vs. 88.47%). In the class-based setting, it forgets too little (91.94%), although its MIA efficacy moves towards the retrained model (0.297 vs. 0.238). SWIFT runs in under 40 seconds, about $5\times$ faster than retraining and 8–$30\times$ faster than Two-Stage.

A.5 Varying the forget class on CIFAR-100

The main experiments on CIFAR-100 delete *aquarium fish*. We now test whether the behavior of SWIFT holds for other forget classes. We repeat both settings with three more subclasses from different superclasses, *beaver* (*aquatic mammals*), *bee* (*insects*) and *bicycle* (*vehicles 1*). In the class-based setting, $\mathcal{D}_f$ is the whole subclass and $\mathcal{D}_r^{\mathrm{adj}}$ contains the other four subclasses of its superclass. In the instance-based setting, $\mathcal{D}_f$ is a random half of the subclass and $\mathcal{D}_r^{\mathrm{adj}}$ is the other half. All other settings follow Section 4. Table 10 reports the results.

Table 8: Results on SBIC (class-based) using RoBERTa-base model. SWIFT never overshoots the retrained model, while SCRUB, GDR and Two-Stage lower the forget accuracy to 14–51%. Accuracy is measured against the true offense labels.

	Train acc.			Test acc.			MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	gap	Time (s)
Original	92.94	94.47	91.37	90.08	93.22	82.76	0.110	0.094	0.017	–
Retrain	83.78	94.43	91.44	87.05	93.54	83.43	0.238	0.095	0.143	201.0
GA	92.57	94.37	91.11	89.81	93.62	82.72	0.110	0.086	0.023	72.9
SCRUB	13.56	10.26	62.70	9.92	7.34	61.01	0.864	0.897	−0.033	38.8
SSD	88.25	94.32	91.22	88.15	93.22	82.45	0.191	0.098	0.093	48.9
GDR	31.97	92.57	89.30	24.24	92.19	82.59	0.693	0.114	0.579	52.7
MUNBa	91.35	92.26	90.05	87.60	90.51	82.96	0.213	0.174	0.039	254.6
Two-Stage	51.22	89.82	93.25	47.93	86.60	83.10	0.811	0.490	0.321	1183.8
SWIFT	91.94	94.54	92.06	88.98	93.22	82.86	0.297	0.127	0.170	39.1

Table 9: Results on SBIC (instance-based) using RoBERTa-base model. SWIFT keeps the forget accuracy closest to the retrained model (90.02% vs. 88.47%), while GDR and Two-Stage lower it to about 51% and unlearn the other half of the same posts as well. Test $\mathcal{D}_f/\mathcal{D}_r^{\text{adj}}$ are the r/meanjokes test posts.

	Train acc.			Test acc.		MIA			
Method	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f/\mathcal{D}_r^{\text{adj}}$	$\mathcal{D}_r^{\text{rem}}$	$\mathcal{D}_f$	$\mathcal{D}_r^{\text{adj}}$	gap	Time (s)
Original	93.57	92.31	92.13	90.08	85.88	0.106	0.127	−0.021	–
Retrain	88.47	92.61	92.09	91.18	86.04	0.172	0.123	0.050	217.7
GA	93.27	92.31	92.06	90.08	85.97	0.105	0.128	−0.023	32.5
SCRUB	12.79	14.34	49.94	9.92	45.01	0.872	0.857	0.016	18.5
SSD	92.24	91.94	91.96	89.53	85.73	0.142	0.143	−0.001	40.5
GDR	51.07	52.62	89.99	47.66	85.19	0.586	0.568	0.018	21.3
MUNBa	92.83	90.69	90.72	87.88	85.19	0.210	0.223	−0.013	203.7
Two-Stage	51.88	56.17	94.56	47.11	85.50	0.758	0.727	0.032	307.6
SWIFT	90.02	91.94	93.19	87.60	85.88	0.259	0.300	−0.041	38.5

Retraining does not forget completely. Even when the whole subclass is deleted, the retrained model keeps 27.40–49.20% accuracy on it. The other subclasses of the superclass still teach it the shared label. When only half of the subclass is deleted, it keeps much more (58.80–90.80%). So, the correct amount of forgetting depends on the forget class or set.

SWIFT follows this amount without knowing the setting. In the instance-based setting, its forget accuracy stays within 7 points of the retrained model for all four classes. In the class-based setting, it forgets somewhat more than retraining, by 1.2 points on *bicycle* and up to 16.6 points on *beaver*. SWIFT is never told which setting it is in. The difference comes from the calibration reference, whose held-out images are unfamiliar to the model when the whole subclass is deleted and familiar when only half is deleted. Its MIA efficacy on $\mathcal{D}_f$ follows the same pattern, within 0.06 of the retrained model in the class-based setting and within 0.13 in the instance-based setting.

Retained accuracy is slightly lower than after retraining. Accuracy on $\mathcal{D}_r^{\text{rem}}$ stays within 2.5 points of the retrained model. Accuracy on $\mathcal{D}_r^{\text{adj}}$ drops further, by up to 8.4 points in the class-based setting and 15.2 points on *beaver* in the instance-based setting. The instance-based MIA gap between $\mathcal{D}_f$ and $\mathcal{D}_r^{\text{adj}}$ is also smaller than after retraining (0.080–0.120, compared with 0.164–0.364), which indicates that part of the forgetting spreads to the retained half of the subclass. This matches the held-out gap observed in the main results.

A.6 Hyperparameters

Original and retrained models. For VGGFace2, we train a ResNet-50 from scratch on 128×128 face crops of all 540 identities for 30 epochs (SGD with Nesterov momentum 0.9, learning rate 0.2, batch size 512, weight decay 5×10^{-4}, 2 warm-up epochs and cosine decay), with standard augmentations. For ToxiGen and SBIC, we fine-tune RoBERTa-base for 3 epochs (AdamW, learning rate 2×10^{-5}, batch size 64, maximum length 128 tokens, 10%

Table 10: Original, retrained and SWIFT models on four forget classes of CIFAR-100. C denotes the class-based and I the instance-based setting. Accuracies are measured on training data. MIA is the fraction of $\mathcal{D}_f$ labeled as non-member, and gap is the difference between MIA efficacy on $\mathcal{D}_f$ and on $\mathcal{D}_r^{\mathrm{adj}}$. For comparison, the Two-Stage row reports the class-based results of Cheng et al. (2026, Table 1), obtained with their own training recipe.

Class		Model	$\mathcal{D}_f$	$\mathcal{D}_r^{\mathrm{adj}}$	$\mathcal{D}_r^{\mathrm{rem}}$	MIA	gap
aquarium fish	C	Original	99.60	98.00	97.49	0.066	−0.056
(*fish*)	C	Retrain	32.80	99.35	99.01	0.856	0.726
	C	Two-Stage	0.00	98.17	98.44	–	–
	C	SWIFT	24.80	90.95	95.55	0.890	0.600
	I	Original	99.60	99.60	97.52	0.072	0.012
	I	Retrain	79.60	98.80	99.00	0.432	0.348
	I	SWIFT	80.00	98.40	96.74	0.308	0.120
beaver	C	Original	93.80	95.95	97.64	0.328	0.087
(*aquatic mammals*)	C	Retrain	27.40	98.15	99.11	0.924	0.697
	C	SWIFT	10.80	89.75	96.68	0.974	0.622
	I	Original	93.60	94.00	97.57	0.324	−0.016
	I	Retrain	58.80	97.20	98.96	0.728	0.364
	I	SWIFT	53.20	82.00	96.76	0.776	0.116
bee	C	Original	98.80	97.60	97.52	0.124	0.002
(*insects*)	C	Retrain	49.20	98.55	99.04	0.802	0.659
	C	SWIFT	38.40	93.00	96.74	0.834	0.581
	I	Original	98.40	99.20	97.52	0.144	0.052
	I	Retrain	76.00	99.60	98.93	0.404	0.336
	I	SWIFT	82.80	93.60	96.75	0.368	0.080
bicycle	C	Original	99.60	95.70	97.59	0.060	−0.145
(*vehicles 1*)	C	Retrain	39.80	99.10	98.95	0.854	0.726
	C	SWIFT	38.60	96.50	97.04	0.910	0.685
	I	Original	100.00	99.20	97.52	0.056	−0.008
	I	Retrain	90.80	100.00	98.94	0.224	0.164
	I	SWIFT	84.40	96.40	96.72	0.332	0.096

warm-up and linear decay). The original and retrained models use the same recipe and seed. They differ only in whether $\mathcal{D}_f$ is part of the training data. For CIFAR-100, we fine-tune an ImageNet-pretrained ResNet-18 on the 20 superclass labels for 10 epochs (Adam, learning rate 2×10^{-5}, batch size 128, images resized to 224×224) and keep the checkpoint with the lowest validation loss.

SWIFT. We use the same settings on every dataset unless stated otherwise in Table 11. SWIFT runs for one epoch over $\mathcal{D}_r$, with retain and forget batch sizes of 128 and 64. Both distances are computed on 256 quantiles. The calibration reference is pooled over 256 samples and refreshed every 10 steps. The multiplier starts at $\lambda_0 = 1$ and is updated with $\rho = 2$ and $\beta = 1.2$, checked every 100 steps and capped at $\lambda_{\max} = 100$. The tolerance is $\kappa = 1.5$ times the sampling floor of the retain term. Gradients are clipped at norm 5. Then, batch-normalization statistics stay frozen during unlearning. Otherwise, a forget batch drawn from a single class overwrites them and harms $\mathcal{D}_r^{\mathrm{rem}}$. The checkpoint selection uses fixed subsets of 512 samples from $\mathcal{D}_f$, $\mathcal{D}_{\mathrm{cal}}$ and $\mathcal{D}_r^{\mathrm{probe}}$.

Table 11: Dataset-specific settings of SWIFT.

	CIFAR-100	VGGFace2	ToxiGen, SBIC
Optimizer	AdamW, 10^{-4}	SGD, 3×10^{-3}, momentum 0.9	AdamW, 2×10^{-5}
Retain reference	probe	original model	probe
Probe training	head + `layer4`	–	head + last encoder layer
Probe learning rate	10^{-3}	–	10^{-3} (head), 5×10^{-5} (layer)
Probe steps, batch	300, 128	–	300, 64
$\lvert\mathcal{D}_r^{\mathrm{probe}}\rvert$	5,000	4,860	5,000
Selection interval	50 steps	50 steps	100 steps

On VGGFace2, we use SGD instead of AdamW. The ResNet-50 was trained with SGD and has small weights, so the fixed step size of Adam changes them too much. In this case, we skip the probe on VGGFace2 and use the original model as the retain reference (Section 3.3).

Baselines. Every baseline starts from the original model. Table 12 lists the main settings. For GA and MUNBa on VGGFace2, and for GA on CIFAR-100, the default learning rate either collapses the model or leaves it unchanged. For these, we try a few learning rates and apply the same rule to all of them.

Table 12: Main settings of the baselines (optimizer, learning rate, epochs). †Selected from a small learning-rate sweep, as described in the text.

Method	CIFAR-100	VGGFace2	ToxiGen, SBIC
GA	SGD, XXX†, 7	AdamW, 5×10^{-7}†, 9	SGD, 2.5×10^{-6}, 10
SCRUB	Adam, 5×10^{-5}, 5	SGD, 8×10^{-5}, 6	Adam, 10^{-5}, 5
SSD	$\lambda = 1$, $\alpha = 10$	$\lambda = 1$, $\alpha = 10$	$\lambda = 1$, $\alpha = 50$
GDR	AdamW, 10^{-4}, 3	SGD, 10^{-4}, 10	AdamW, 2×10^{-5}, 3
MUNBa	SGD, 0.03, 10	AdamW, 3×10^{-6}†, 5	SGD, 10^{-5}, 3
Two-Stage	Adam, 2.5×10^{-6} / SGD, 2×10^{-5}, 6	same as CIFAR-100	Adam, 2.5×10^{-6} / SGD, 10^{-5}, 2

A.7 Hardware configurations

All experiments run on a server with two NVIDIA H100 NVL GPUs (94 GB each), two Intel Xeon Gold 6526Y CPUs and 251 GB of RAM. Each run uses a single GPU. We use PyTorch 2.2.2 with CUDA 12.1.

A.8 Same loss distribution implies same accuracy

For a model θ and a sample (x, y), let $p_y(\theta; x)$ be the softmax probability of the true class, so that $\ell(\theta; x, y) = -\log p_y(\theta; x)$. The prediction is correct when p_y is larger than the probability of every other class. For a finite set $\mathcal{S}$ and a threshold z, let $F^{<}_{\mathcal{S},\theta}(z)$ and $F^{\leq}_{\mathcal{S},\theta}(z)$ be the fractions of samples in $\mathcal{S}$ whose loss under θ is below z and at most z, respectively. Both fractions are determined by the loss distribution $P_\ell(\mathcal{S}; \theta)$.

Proposition 2. *For every model θ and finite set $\mathcal{S}$,*

$$F^{<}_{\mathcal{S},\theta}(\log 2) \;\leq\; \mathrm{Acc}(\theta; \mathcal{S}) \;\leq\; F^{\leq}_{\mathcal{S},\theta}(\log n_c). \tag{10}$$

For binary classification ($n_c = 2$), $\mathrm{Acc}(\theta; \mathcal{S}) = F^{<}_{\mathcal{S},\theta}(\log 2)$. Two models with the same loss distribution on $\mathcal{S}$ therefore have the same accuracy on $\mathcal{S}$.

Proof. Consider one sample. If its prediction is correct, p_y is the largest of n_c probabilities that sum to 1, so $p_y \geq 1/n_c$. Conversely, if $p_y > 1/2$, the other classes share less than $1/2$, so each of them has probability below p_y and the prediction is correct. Since the loss $\ell = -\log p_y$ decreases as p_y increases, these two facts read

$$\ell < \log 2 \implies \text{correct} \implies \ell \leq \log n_c. \tag{11}$$

The fraction of samples in $\mathcal{S}$ satisfying each of the three conditions therefore increases from left to right, which is equation 10. For $n_c = 2$, the prediction is correct exactly when $p_y > 1 - p_y$, that is, when $p_y > 1/2$ and hence $\ell < \log 2$. Accuracy then equals $F^{<}_{\mathcal{S},\theta}(\log 2)$, which depends on θ only through its loss distribution. □

For $n_c > 2$, a sample with $1/n_c \leq p_y \leq 1/2$ may be correct or not, depending on how the remaining probability is spread over the other classes. Matching loss distributions still fixes both bounds in equation 10, whereas matching mean losses fixes neither.

A.9 Utility of the calibration set

Section 3.2 replaces the forget reference of the retrained model by the loss distribution on a calibration set $\mathcal{D}_{\mathrm{cal}}$. We show that this replacement is exact in expectation and quantify its error for finite sets. Every sample in $\mathcal{D}_f$ and $\mathcal{D}_{\mathrm{cal}}$ carries the label y_0, so the loss of a sample depends only on its input x. We use the fraction $F^{\leq}_{\mathcal{S},\theta}(z)$ from Appendix A.8.

Assumptions. (i) The inputs of $\mathcal{D}_f$ and $\mathcal{D}_{\text{cal}}$ are independent draws from the same distribution μ_f, and both sets carry the label y_0. (ii) The retrained model $\theta^\star = A(\mathcal{D}_r)$ is trained without $\mathcal{D}_f$ and $\mathcal{D}_{\text{cal}}$. Assumption (i) holds by construction, since $\mathcal{D}_{\text{cal}}$ is drawn from held-out samples of the same subgroup as $\mathcal{D}_f$ and, on ToxiGen, is relabeled in the same way. Assumption (ii) holds by the definition of $\theta^\star$.

Fix the retrained model $\theta^\star$. Let $P^\star$ be the distribution of the loss $\ell(\theta^\star; x, y_0)$ when x is drawn from μ_f, and let $F^\star(z)$ be the probability that this loss is at most z. Let $\hat{P}_f = P_\ell(\mathcal{D}_f; \theta^\star)$ and $\hat{P}_{\text{cal}} = P_\ell(\mathcal{D}_{\text{cal}}; \theta^\star)$ be the two empirical loss distributions. All expectations and probabilities below are over the random draw of $\mathcal{D}_f$ and $\mathcal{D}_{\text{cal}}$, with $\theta^\star$ held fixed.

Proposition 3. *Under (i) and (ii), for every threshold z,*

$$\mathbb{E}\big[F^{\leq}_{\mathcal{D}_f,\theta^\star}(z)\big] = \mathbb{E}\big[F^{\leq}_{\mathcal{D}_{\text{cal}},\theta^\star}(z)\big] = F^\star(z). \tag{12}$$

Moreover, for any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\max_z \big|F^{\leq}_{\mathcal{D}_f,\theta^\star}(z) - F^{\leq}_{\mathcal{D}_{\text{cal}},\theta^\star}(z)\big| \;\leq\; \varepsilon(|\mathcal{D}_f|) + \varepsilon(|\mathcal{D}_{\text{cal}}|), \qquad \varepsilon(n) = \sqrt{\frac{\log(4/\delta)}{2n}}. \tag{13}$$

Proof. By (ii), $\theta^\star$ does not depend on $\mathcal{D}_f$ or $\mathcal{D}_{\text{cal}}$. By (i), the losses of the samples in both sets are then independent draws from $P^\star$. Each sample has loss at most z with probability $F^\star(z)$, so the expected fraction of such samples is $F^\star(z)$ in both sets. This gives equation 12. For equation 13, the Dvoretzky–Kiefer–Wolfowitz inequality (Massart, 1990) states that for a set of n independent samples, the fraction $F^{\leq}_{S,\theta^\star}(z)$ stays within $\varepsilon(n)$ of $F^\star(z)$ for all z at once, with probability at least $1-\delta/2$. Applying it to both sets and adding the two deviations gives the result. □

Under the retrained model, $\mathcal{D}_f$ and $\mathcal{D}_{\text{cal}}$ thus have the same loss distribution up to sampling error. Matching the two therefore aims at the behavior of $\theta^\star$ on $\mathcal{D}_f$, without training $\theta^\star$. By Proposition 2, the same holds for accuracy in binary classification. With probability at least $1-\delta$, $|\text{Acc}(\theta^\star; \mathcal{D}_f) - \text{Acc}(\theta^\star; \mathcal{D}_{\text{cal}})| \leq \varepsilon(|\mathcal{D}_f|) + \varepsilon(|\mathcal{D}_{\text{cal}}|)$. The next result states the same for the forget term itself.

Corollary 1. *Under (i) and (ii), the forget term vanishes at $\theta^\star$ when both sets are replaced by their common distribution $P^\star$, since $W_2^2(P^\star, P^\star) = 0$. If, in addition, all losses lie in $[0, B]$, the forget term on the finite sets satisfies*

$$\mathbb{E}\Big[W_2^2\big(\hat{P}_f,\, \hat{P}_{\text{cal}}\big)\Big] \;\leq\; B^2 \left(\frac{1}{\sqrt{|\mathcal{D}_f|}} + \frac{1}{\sqrt{|\mathcal{D}_{\text{cal}}|}}\right). \tag{14}$$

Proof. We use the 1-Wasserstein distance $W_1(P,Q) = \int_0^1 |F_P^{-1}(u) - F_Q^{-1}(u)|\, du$, the analogue of equation 3 without the square. We bound equation 14 in three steps. First, W_2 is a distance, so $W_2(\hat{P}_f, \hat{P}_{\text{cal}}) \leq W_2(\hat{P}_f, P^\star) + W_2(P^\star, \hat{P}_{\text{cal}})$. Squaring and using $(a+b)^2 \leq 2a^2 + 2b^2$ gives $W_2^2(\hat{P}_f, \hat{P}_{\text{cal}}) \leq 2W_2^2(\hat{P}_f, P^\star) + 2W_2^2(P^\star, \hat{P}_{\text{cal}})$. Second, all quantiles lie in $[0, B]$, so each difference in equation 3 satisfies $|a-b|^2 \leq B|a-b|$, and hence $W_2^2 \leq B\, W_1$. Third, for n independent samples on the real line, the expected W_1 distance between the empirical distribution and $P^\star$ is at most $B/(2\sqrt{n})$ (Bobkov & Ledoux, 2019). Combining the three steps gives $\mathbb{E}[W_2^2(\hat{P}_f, \hat{P}_{\text{cal}})] \leq 2B \cdot \frac{B}{2}\big(|\mathcal{D}_f|^{-1/2} + |\mathcal{D}_{\text{cal}}|^{-1/2}\big)$, which is equation 14. □

Remarks. The results do not hold for the original model. θ_0 was trained on $\mathcal{D}_f$, so assumption (ii) fails, and its losses on $\mathcal{D}_f$ are lower than on $\mathcal{D}_{\text{cal}}$. This gap is the memorization that the forget term removes. The results also concern $\theta^\star$, whereas SWIFT evaluates the reference under the current model θ_t. The calibration set is therefore a proxy during optimization, not an exact target, and the retain term is needed to keep θ_t from degrading on both sets at once (Figure 3). Finally, both error bounds shrink as $1/\sqrt{|\mathcal{D}_{\text{cal}}|}$. A small calibration set gives a noisy reference, which is why we pool calibration losses across batches, as we showed in optimization.