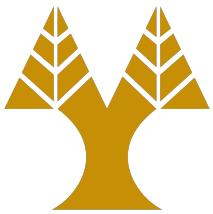
Thesis Dissertation

THE RETURN OF FORGOTTEN KNOWLEDGE: ON THE REVERSIBILITY OF APPROXIMATE MACHINE UNLEARNING

Charalambos Thoma

UNIVERSITY OF CYPRUS



COMPUTER SCIENCE DEPARTMENT

May 2026

UNIVERSITY OF CYPRUS
COMPUTER SCIENCE DEPARTMENT

The return of forgotten knowledge: On the reversibility of Approximate Machine Unlearning

Charalambos Thoma

Supervisor
Dr. Elias Athanasopoulos

Thesis submitted in partial fulfilment of the requirements for the award of degree of Bachelor in Computer Science at University of Cyprus

I declare that this dissertation and any accompanying software are my own original work, conducted in full compliance with the University of Cyprus Code of Conduct for Research (https://www.ucy.ac.cy/legislation/kodikas-deontologias-kai-politikes/), and with full accountability on my part for the accuracy, integrity, and validity of all content.

May 2026

Acknowledgments

Above all, I would like to express my heartfelt gratitude to my supervisor, Dr. Elias Athanasopoulos, for his invaluable guidance, support, and encouragement throughout this research journey. His expertise and intuition, along with constant feedback, helped me produce the best possible result for my thesis. I am truly grateful for the opportunity to work under his supervision, growing both academically and personally.

I would also like to thank Dr. Andreas Dionysiou for introducing me to the topic of machine unlearning and for his deep expertise in the field, which proved vital to the outcome of this thesis. I would further like to thank Panayiotis Charitonos for his patience in helping me understand his previous work, which served as the foundation upon which this thesis builds.

Finally, I would like to thank my family, my girlfriend Sophia, and my friends for their constant support and encouragement throughout this journey.

Abstract

Machine Unlearning (MU) has emerged as a practical mechanism for removing the influence of specific training data from deployed models, motivated largely by privacy regulations and the right to be forgotten. Approximate MU methods have gained considerable traction due to their computational efficiency, yet they operate under a fundamental trade-off, they trade completeness for scalability leaving room for residual information related to the forgotten data to persist within the model's parameters. This thesis investigates the extent to which such residual traces survive class-level approximate unlearning, and whether a post-hoc adversary can exploit them. Specifically, we examine whether limited fine-tuning after unlearning is sufficient to amplify latent class-level signals to the point where an unlearned model becomes distinguishable from one retrained from scratch without the target class. We evaluate this threat across three approximate MU methods, UNSIR, SalUn, and LTU, each representing a distinct approach to efficient, deployment-oriented unlearning. Our results confirm that residual class information survives unlearning across all three frameworks, and that targeted post-unlearning fine-tuning reliably surfaces it. Current MU evaluation protocols do not consider adversaries of this kind, and our findings suggest this is a meaningful gap. We argue that robustness to post-unlearning amplification warrants explicit consideration in the design and evaluation of future MU methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Privacy-aware machine learning has become an active area of concern as models trained on personal data are deployed at scale. Legal frameworks such as the GDPR grant individuals the right to request deletion of their data, yet fulfilling this right is far from trivial once a model has already been trained. Although retraining from scratch each time a deletion request arrives offers the strongest privacy guarantees, it is often infeasible in practice due to its computational cost, especially for larger models, making it impractical in most real-world deployments [3, 4].

This has given rise to the field of Machine Unlearning (MU), which studies how to modify a trained model to remove the influence of designated data without full retraining. The most principled approaches maintain auxiliary structures that support exact rollback of individual data contributions, but these too impose significant computational and storage costs [6, 14, 19, 32]. As a result, the majority of practical MU work has converged on *approximate* methods, which trade formal guarantees for efficiency by applying targeted, heuristic modifications to the model's weights to suppress the influence of the data to be forgotten [5, 15–17, 26, 35, 42, 44, 46, 49]. A particularly relevant subclass of these methods targets the removal of entire data classes rather than individual records, an operation common in face recognition systems where a single user's data constitutes a class [2, 8, 11, 29, 41, 46].

Regardless of which method is used, any unlearned model must satisfy a fundamental privacy requirement: an adversary who obtains the model after unlearning should be unable to determine whether the model was originally trained on specific data and later modified to remove it, or whether it was simply never trained on that data at all [3, 14, 39]. Current evaluation practice tests this requirement by checking whether the unlearned model misclassifies forgotten samples [3, 16], whether membership inference attacks fail

to distinguish them from non-members [9, 15], or whether the gradient footprint of the forgotten data has been sufficiently diminished [19,48]. Each of these checks examines the model at a single point in time, immediately after unlearning, and none of them consider what an adversary might be able to learn by subsequently updating the model.

This is a significant blind spot. Some approximate MU methods already incorporate a post-unlearning repair step in which the model is briefly fine-tuned on retained data to recover lost utility [46]. If fine-tuning on retained data can restore performance, it is natural to ask whether fine-tuning on the forgotten data itself can restore what was supposedly removed. This thesis addresses that question directly. We show that an adversary equipped with samples from a forgotten class and white-box access to an unlearned model can, through a small number of fine-tuning steps, reliably determine whether that class was ever part of the model's training data. We refer to this process as *memory jogging* [1], and show that it enables *class-level membership inference* against models that pass all standard unlearning evaluations.

We focus on a threat model in which the adversary obtains access to the model only after unlearning has taken place and holds samples from the class in question. This is a realistic scenario in face recognition, where user images are typically not kept secret and model access can be obtained through auditing, regulatory inspection, or model extraction techniques [25, 40, 47]. The full specification of the threat model is given in Sec. 3.2.

To assess how broadly this vulnerability applies, we instantiate our analysis on three representative approximate MU frameworks. UNSIR [46] operates exclusively at the class level by generating synthetic anti-samples to disrupt class representations. SalUn [13] targets both individual samples and classes through gradient-based weight saliency. LTU [24] approaches unlearning as a meta-learning problem, using gradient harmonization to balance forgetting and utility preservation. Despite their different mechanisms, all three share the same underlying objective: increase the loss on the forgotten data while keeping performance on the retained data intact. This common structure makes them well-suited for a controlled study of whether residual class-level information is a property of approximate MU in general, or an artifact of any particular method.

Experiments conducted across five datasets, MNIST, SVHN, CIFAR-10, CIFAR-100, and VGGFace100, and two model families, ResNet and Vision Transformers (ViT), show that all three frameworks leave behind detectable residual signals. In every setting, a small number of fine-tuning steps on the forgotten class produces a measurable separation between a model that underwent unlearning and one that was never trained on that class. This separation persists across architectures and datasets, and remains detectable even when the adversary does not know which output node represented the forgotten class.

[1] Hereafter, we use the term *memory jogging* to denote the process of recovering an unlearned model's memory of forgotten data through fine-tuning.

1.2 Contributions

This thesis makes the following contributions:

- **A new attack vector against approximate MU.** We demonstrate that targeted post-unlearning fine-tuning can amplify latent class-level signals in models that pass standard unlearning evaluations, enabling an adversary to determine whether a given class was part of a model's original training data.
- **An evaluation gap in current MU practice.** We show that static, post-unlearning assessments are insufficient to characterize the privacy guarantees of approximate MU methods. Residual information that is invisible under standard checks can become detectable once the model is subjected to further updates.
- **A systematic empirical study.** We evaluate three state-of-the-art approximate MU frameworks, UNSIR, SalUn, and LTU, across five datasets and two model architectures, demonstrating consistent and reliable separability between unlearned and never-learned classes under memory jogging.
- **Reproducibility.** The source code for all experiments is released at our GitHub repository to facilitate further research on post-unlearning privacy risks.

Chapter 2

Background

2.1 Machine Unlearning and its Challenges

Given a trained ML model and a dataset $\mathcal{D}$, MU aims to revert the effects of a specific subset of data instances, denoted as $\mathcal{D}_f \subset \mathcal{D}$, referred to as the *forget set*. The goal is for the ML model to behave as if it had never been trained on $\mathcal{D}_f$, but rather only on a disjoint subset of $\mathcal{D}$, denoted as $\mathcal{D}_r$, referred to as the *retain set*, such that $\mathcal{D}_f \cap \mathcal{D}_r = \emptyset$.

Removing the influence of specific data records from a trained ML model is inherently challenging for several reasons. Modern deep networks operate over tens of millions of parameters, ResNet-18 alone has approximately 11 million, while transformer-based architectures push this into the hundreds of millions. Due to the black-box nature of deep learning, the precise way in which these parameters collectively encode any given piece of information is not well understood. Training proceeds incrementally, meaning any given weight update reflects the accumulated effect of all preceding updates, making it difficult to isolate and reverse the contribution of any specific subset of data [3]. Additionally, multiple models can generalize equally well on $\mathcal{D}_r$ due to stochasticity in training (e.g., random batch sampling) and the non-uniqueness of learned hypotheses that minimize an empirical notion of error. The highly non-convex loss landscape of deep ANNs further compounds this, making it difficult to model the precise influence of individual samples on the optimization trajectory and final parameters [3, 15].

As a consequence, approximate MU methods typically resort to suppressing class-level information rather than truly erasing it. Whether through synthetic anti-samples [46], saliency-guided weight perturbations [13], or meta-learned gradient updates [24], these methods apply constrained modifications to the model's weights that reduce the model's ability to recognize the forgotten class. However, given the scale and opacity of deep networks, such modifications cannot guarantee that all traces of a class are removed. What appears to happen in practice is that learned features are suppressed rather than

eliminated, leaving latent representations that may still be recoverable under the right conditions.

2.2 Limitations of Existing Evaluation Practices

The effectiveness of approximate MU methods is typically assessed using empirical criteria, including post-unlearning classification performance on retained data [3, 16], misclassification rates on forgotten data [3, 16], robustness against membership inference attacks [9, 15], gradient or influence-based measures of forgotten data contribution [19,48], and similarity metrics comparing unlearned models to retrained-from-scratch baselines [3, 16].

At first glance, these metrics paint a convincing picture: classification accuracy on the forgotten class drops to near zero, membership inference attacks fail to distinguish forgotten samples from non-members, and gradient norms suggest diminished influence. However, these results are unsurprising given that approximate MU methods are explicitly trained to suppress exactly these signals. Low accuracy on the forgotten class is a direct consequence of the unlearning objective, not independent evidence that the underlying representations have been removed.

A more informative diagnostic would examine how quickly a model can reacquire knowledge of the forgotten class under fine-tuning, what some works refer to as relearn time [46]. The intuition is straightforward: a model that was never trained on a class should require substantially more updates to learn it from scratch than a model that previously encoded it and was later suppressed. If an unlearned model recovers from near-zero accuracy to moderate performance in just one or two epochs, while a truly naive model remains near chance level over the same period, that asymmetry is a strong indicator that suppression rather than erasure has taken place. Existing evaluations, however, do not consistently apply this kind of dynamic assessment, and when they do, the threshold for what constitutes successful relearning is not always defined carefully enough to expose this gap.

More broadly, the dominant evaluation paradigm assumes *static* and *black-box* access to the unlearned model: the adversary can only query the model and observe its outputs, and the model's weights are not modified after unlearning. Under this assumption, approximate methods may appear adequate. The more relevant and underexplored threat, however, involves *white-box* access, where an adversary can directly inspect and modify the model's parameters. This setting exposes a fundamentally different class of vulnerabilities, as latent representations that are invisible through output queries alone may become detectable once the model is subject to targeted weight updates. This is precisely the gap our work addresses.

2.3 Overview of MU Frameworks

To ground our discussion, we briefly outline the landscape of MU frameworks, distinguishing between approaches for non-ANN models and those targeting deep ANNs. Our focus is on deep ANNs, as such models are among the most widely deployed in practice and incur the highest computational costs, making retraining-from-scratch-based unlearning impractical. Moreover, the majority of prior MU work evaluates image classification tasks, including object recognition on CIFAR-10 [2, 8, 11, 13, 24, 29, 33, 41, 46], CIFAR-100 [13, 29, 41, 46], Tiny ImageNet [13, 24], ImageNet [29], and Food101 [41]; digit recognition on MNIST [2, 11, 33, 41] and SVHN [11, 13]; and face recognition on AT&T Faces [2], VGGFace-100 [46], VGGFace2 [8], and CelebA [33], motivating our focus on this setting. Finally, we concentrate our empirical analysis on a representative set of state-of-the-art approximate MU frameworks, and motivate this choice at the end of Sec. 2.3.2.

2.3.1 Unlearning for Non-ANN Models

MU was originally introduced as *decremental learning* by Cauwenberghs & Poggio [5] in 2001, where the authors presented an online and provably reversible learning algorithm for Support Vector Machines (SVMs). Online learning updates a model's parameters on a per-sample basis, whereas ANNs are typically trained in batches, with weight updates driven by the accumulated error over a batch of samples. Training SVMs requires solving a quadratic programming (QP) problem in a number of coefficients equal to the number of training samples. For the large datasets typical of contemporary ML workloads, however, standard numerical techniques for QP become infeasible [5], making provable removal significantly more challenging for deep ANNs.

Jian et al. [26] proposed an online decremental learning algorithm for LS-SVM that does not require retaining all training data, though the method is heuristic and previously learned knowledge may not be preserved as new samples arrive. Lee et al. [32] later proposed a provable model updating approach for LS-SVM supporting both incremental and decremental learning without requiring access to all training data. Beyond SVMs, MU frameworks have been developed for clustering methods, linear models, and decision trees [7, 14, 19, 35, 44].

2.3.2 Unlearning for ANN Models

Cao et al. [4] showed that provable MU is possible when the learning algorithm can be converted to its equivalent in Statistical Query (SQ) learning [27] and queries are made in a predetermined order. In the case of adaptive queries, however, efficient SQ learning

methods for deep ANNs do not exist [3].

Golatkar et al. [16] proposed an MU framework for deep ANNs trained with SGD involving a shift in weight space and addition of noise to destroy information. The method carries notable limitations: (a) it assumes the model has been pre-trained on a generic backbone and that weights do not change significantly during fine-tuning, a strong assumption given that even small changes in training settings can lead to dramatically different solutions [1, 18]; and (b) its efficacy has not been demonstrated under optimizers such as Adam [28] or conjugate gradient [22]. More fundamentally, the framework is effective only when the loss function is quadratic, which is rarely the case in practice, and even then, practical optimization proceeds in discrete steps rather than as a gradient flow [16].

Peste et al. [42] proposed SSSE, an extension that also considers the non-diagonal terms of the Fisher Information Matrix (FIM), excluding the noise term. SSSE is however highly sensitive to the number of removed samples, which affects the quality of its Taylor approximation, and assumes negative log-likelihood loss for FIM computation.

In a follow-up, Golatkar et al. [17] quantify the maximum information an attacker can extract from input-output observations and propose procedures for removing it in one shot. The framework inherits the limitations of their prior work and imposes memory and time complexity of $O(c^2|\mathcal{D}_r|^2)$ and $O(|w|c^2|\mathcal{D}_r|^2)$ respectively, where c is the number of classes, $|\mathcal{D}_r|$ the number of retained instances, and $|w|$ the number of parameters. The authors themselves acknowledge that their framework is not yet scalable to production level.

Bourtoule et al. [3] proposed SISA, a provable MU framework that partitions training data into disjoint shards, trains separate models on each, and aggregates their predictions. For each unlearn request, only the affected shard's model is retrained. Despite its simplicity, SISA still imposes significant overhead when the forgotten class is distributed evenly across shards, as all models must then be retrained. Splitting data into shards can also degrade accuracy, and maintaining per-slice checkpoints introduces substantial storage costs [3].

Wu et al. [49] proposed DeltaGrad, which accelerates retraining by caching gradient and curvature information from earlier training. Its theoretical guarantees hold only for convex problems, and when applied to non-convex deep ANNs at scale, the computational overhead remains prohibitive [15]. The authors' own experiments on ResNet-152 freeze all but the final layer, effectively reducing the problem to convex logistic regression.

Golatkar et al. [15] introduced a framework that designates a protected core subset of training data not subject to forgetting, enabling efficient removal from the non-core subset. Deploying this framework requires substantial modifications to the model's architecture and training pipeline, making it impractical for general use. The approach also scales

quadratically with the number of training samples and faces the challenge that evolving privacy regulations may require changes to what constitutes the core subset over time.

Tarun et al. [46] proposed UNSIR, an approximate MU framework that removes class-level information by generating synthetic noise samples designed to disrupt representations associated with a forgotten class. For each target class y_f, UNSIR initializes noise matrices from a standard normal distribution and optimizes them to maximize the model's classification loss on y_f, yielding anti-samples that act as the polar opposite of the forgotten class in the model's latent space. The model is then fine-tuned on these synthetic samples, optionally alongside retained data, followed by a repair phase on the retain set to restore utility. UNSIR operates in a *zero-glance* setting, requiring no access to the forget set, and has seen widespread adoption in MU research due to its practicality and efficiency.

Liu et al. [34] were the first to demonstrate that model sparsification via weight pruning can improve unlearning performance. As later noted by Fan et al. [13], however, identifying an optimal sparse pattern for deep ANNs is far from trivial and can significantly harm utility on retained data. Motivated by this, Fan et al. [13] introduced *Saliency Unlearning* (SalUn), drawing from input saliency methods in model interpretability to guide approximate unlearning in both image classification and generation tasks.

Recently, Huang et al. [24] proposed *Learning-to-Unlearn* (LTU), a meta-learning framework that introduces gradient harmonization to jointly optimize forgetting and utility preservation. LTU fine-tunes on carefully constructed support and query subsets, using membership inference models to guide more thorough forgetting while retaining generalizable knowledge. Although LTU outperforms prior frameworks across multiple metrics in the random sample unlearning setting for which it was designed, two practical challenges arise. First, the computational cost of constructing support and query sets is not explicitly reported, making end-to-end efficiency difficult to assess. Second, no public implementation was available at the time of writing. We therefore developed our own implementation, validating it by reproducing the authors' reported results on random sample unlearning before applying the framework to the class-based unlearning setting examined in this thesis. Key hyperparameters, including the learning rate and number of membership inference models, were not specified in the paper and were determined empirically. It is worth noting that LTU was designed and evaluated exclusively for random sample unlearning: forgetting a small number of individual samples from a class has limited impact on overall classification accuracy, since the model's broader knowledge of that class remains largely intact. Whether the framework generalizes to whole-class removal, a fundamentally different setting, was not examined by the authors and represents an important open question that our work touches upon.

Taken together, UNSIR, SalUn, and LTU constitute a representative subset of state-of-the-art approximate MU frameworks that share a common unlearning strategy: in-

ducing forgetting by increasing the loss on the forgotten data through heuristic weight updates, while preserving utility on the retained data. They differ in how these updates are constructed, ranging from synthetic anti-samples to saliency-guided perturbations and meta-learned gradient alignment, but operate within the same approximate, weight-space modification regime. This shared structure allows us to isolate the effects of approximate MU itself, rather than confounding factors introduced by fundamentally different removal mechanisms. A broader analysis of other approximate MU frameworks operating under similar paradigms is left for future work.

Chapter 3

Architecture

3.1 Intuition

To understand why approximate MU is vulnerable to memory jogging, it helps to think about what training and unlearning actually do in the parameter space of a deep network. Training a model on a dataset is an optimization process over a loss landscape with tens of millions of dimensions. Through iterative gradient descent, the model's parameters converge toward a local minimum of this landscape, one where the model generalizes well across all training classes, including the class that will later be designated for forgetting.

Approximate MU methods attempt to undo this by applying a constrained set of weight updates designed to raise the loss on the forgotten class while keeping it low on everything else. The intuition is that this pushes the model away from the region of parameter space it occupied after training. However, given the scale and opacity of modern deep networks, these updates are necessarily imprecise. Rather than relocating the model to a fundamentally different region of the loss landscape, they introduce a small perturbation that nudges it away from its original minimum without truly escaping the basin of attraction surrounding it. The forgotten class is suppressed rather than erased by changing the model's outputs for that class, but the underlying weight configuration remains geometrically close to where it was.

This distinction has a concrete consequence. A model that has been nudged away from a local minimum but remains within its basin can be guided back toward that minimum through additional gradient-based updates. If an adversary fine-tunes the unlearned model on samples from the forgotten class, the gradient signal from those samples pulls the parameters back toward the region they previously occupied. Because the model was never truly displaced, this recovery happens quickly and with very few updates. In contrast, a model that was never trained on the forgotten class starts from a different region of the landscape entirely and has no latent structure to recover. Fine-tuning it on the same

samples produces a much weaker and slower response, since the model must learn these representations from scratch rather than rediscover ones it already encoded.

This asymmetry is the core signal our analysis exploits. It shows that a model that underwent approximate unlearning responds differently to targeted fine-tuning than one that was genuinely never trained on the class in question.

3.2 Threat Model

We study a single-class unlearning setting in which a target classification model is first trained at time t_0 on a set of classes $C = \{c_i \mid i \geq 2\}$, and at a later point t_1 an approximate MU method is applied to remove a subset $\mathcal{U} \subseteq C$ of those classes. The adversary enters the picture only after t_1, with no knowledge of or access to any earlier version of the model.

Adversary Goal. The adversary holds samples from a set of candidate classes $\mathcal{A} \subseteq C$ and wants to determine whether those classes were part of the model's training data at t_0 and later removed at t_1, following the inference framing of [23]. When the answer is yes, $\mathcal{A}$ overlaps with the unlearned set $\mathcal{U}$. The goal is strictly historical: the adversary is not trying to reconstruct individual training records, nor to probe what the model currently knows. Because the inference target is a data class rather than a specific training example, this threat is weaker than record-level membership inference.

Data Access. The adversary has access to a portion of the original training data $\mathcal{D}$, drawn from both retained and unlearned classes. For retained classes, a handful of representative samples per class is sufficient to probe the model and identify which output nodes still respond with high confidence to those inputs. In our analysis, we assume direct access to samples from the original training distribution, which is a stronger assumption than what is typically made in the membership inference literature [43, 45], where the adversary is assumed to possess only auxiliary data drawn from the same or a related distribution. Concrete sources for such auxiliary data include publicly available datasets, domain-specific corpora, or naturally occurring samples [10, 51]. Whether the attack remains effective under the weaker auxiliary-only setting is an interesting direction for future work, and falls outside the scope of this thesis.

Model Access. The adversary has white-box access to the post-unlearning model [20,36, 45], meaning full visibility into its architecture and learned parameters. This level of access is realistic whenever models are shared for auditing, regulatory review, or fine-tuning, and can also be achieved indirectly through model extraction or replication [25,40,47]. Beyond

simply querying the model, white-box access lets the adversary inspect internal activations and, critically, apply gradient-based updates to the weights.

Using retained-class samples, the adversary can probe the model to identify which output nodes are still actively mapped to known classes, and by elimination, identify the set of unmapped indices $\mathcal{I} \subseteq \{1, \ldots, |C|\}$ that may correspond to previously unlearned classes. These become the targets of the attack.

3.3 Experimental Setup

3.3.1 Datasets and Architectures

We evaluate across five benchmark datasets spanning different visual domains and levels of difficulty. MNIST [31] and SVHN [37] cover digit recognition, with SVHN presenting a more challenging natural-image setting. CIFAR-10 and CIFAR-100 [30] cover object recognition at two scales of class granularity. VGGFace100 [38] is a face recognition benchmark with 100 identities, making it the most directly relevant to the face recognition use case motivating our threat model.

For model architectures, we use ResNet-based convolutional neural networks [21] and the ViT-Base/16 vision transformer [12], covering both the dominant convolutional paradigm and the more recent attention-based architecture. This combination reflects the evaluation settings most commonly used in prior MU work, as discussed in Sec. 2.3.

3.3.2 MU Frameworks

We evaluate three state-of-the-art approximate MU frameworks: UNSIR [46], SalUn [13], and LTU [24]. These were selected because they represent distinct heuristic approaches to approximate unlearning while sharing a common objective structure, as discussed in Sec. 2.3. This makes them well-suited for isolating the effects of approximate MU itself from those of fundamentally different removal mechanisms.

3.3.3 Model Construction

For each target class $c^\star$, we train two models with identical architectures and output spaces. The first is trained on the full dataset and subsequently unlearned using one of the three MU frameworks. The second is trained from scratch on a version of the dataset with all samples of $c^\star$ removed, but retaining the same output dimensionality including the output node for $c^\star$, which remains present but untrained. For 10-class datasets we apply this procedure to every class in turn, while for 100-class datasets we randomly select 10 target

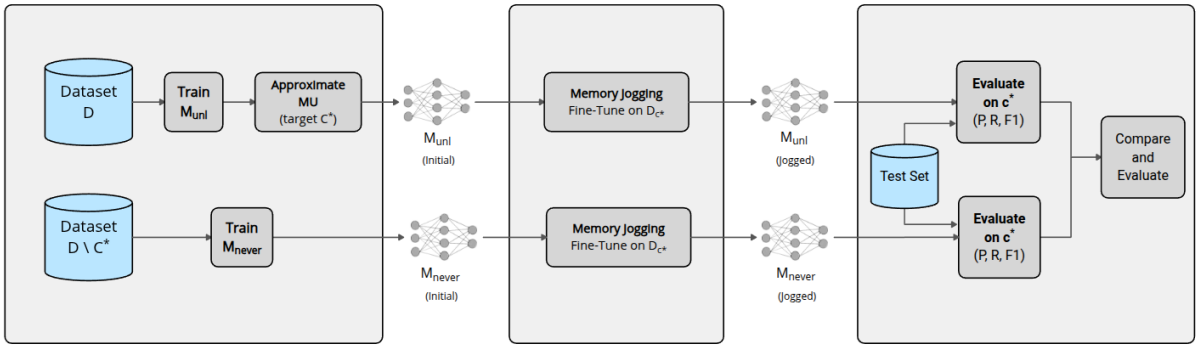


Figure 3.1: Pipeline of the proposed class-level membership inference attack based on memory jogging.

classes. Every experiment is repeated ten times with different random initializations and aggregate results are reported.

3.3.4 Computational Environment

All experiments ran on a high-performance computing cluster. Each run was allocated 5 CPU cores (Intel Xeon Gold 6242R @ 3.10 GHz), 30 GB of RAM, and one GPU, either an NVIDIA Tesla T4 with 16 GB memory or an NVIDIA Tesla V100 with 32 GB memory. The software environment consisted of Python 3.11.3, PyTorch 2.6.0+cu124, CUDA 12.4, and cuDNN 9.1.0, running on CentOS Linux 7 (Core).

3.4 Analysis Methodology

The central question driving our analysis is whether a model that has undergone approximate unlearning is distinguishable from one that was simply never trained on the forgotten class. Standard evaluation answers this with a no, both models misclassify the forgotten class and pass membership inference checks. We ask whether a white-box adversary who can modify the model's weights might get a different answer. The full attack workflow is summarized in Fig. 3.1.

3.4.1 Two-Model Construction

For a given target class $c^\star \in \mathcal{Y}$, where $\mathcal{Y} = \{1, \ldots, |C|\}$, we construct two classifiers over the same input space $\mathcal{X}$ and label space. The unlearned model $M_{\text{unl}} : \mathcal{X} \to \Delta^K$ is trained on the full dataset $\mathcal{D}$ and then subjected to an approximate MU procedure targeting $c^\star$. The never-trained model $M_{\text{never}} : \mathcal{X} \to \Delta^K$ is trained from scratch on $\mathcal{D}_{\setminus c^\star}$, the dataset with all $c^\star$ samples removed. Both models share the same architecture, and crucially, the

same output dimensionality, the node corresponding to $c^\star$ exists in M_{never} but receives no gradient signal during training. At evaluation time, both models are designed to produce near-zero performance on $c^\star$, making them indistinguishable under static assessment.

3.4.2 Memory Jogging Procedure

Following the threat model in Sec. 3.2, the adversary identifies the output index $i \in \mathcal{I}$ corresponding to $c^\star$ by probing the model with retained-class samples. With both the candidate model and samples from $c^\star$ in hand, the adversary performs memory jogging: fine-tuning the model on $\mathcal{D}_{c^\star}$ for a limited number of epochs using a small learning rate. The idea, grounded in the loss-landscape intuition of Sec. 3.1, is that a model which previously encoded $c^\star$ remains geometrically close to the minimum it occupied before unlearning, and even a few gradient steps on $c^\star$ samples will pull it back toward that region. A model that never learned $c^\star$ has no such structure to recover.

In practice, we use 1–2 jogging epochs for 10-class datasets and 5–10 epochs for 100-class datasets, with learning rates of 10^{-3}–2×10^{-3} for ResNet models and 10^{-6} for ViT models. These values were determined empirically to expose residual signals without destabilizing performance on the retained classes.

3.4.3 Evaluation Signal

After jogging, both models are evaluated on a held-out test set containing samples from $c^\star$ alongside all other classes. We record precision, recall, and F1-score on $c^\star$ for each model. The discriminative signal we exploit is the asymmetry in how the two models respond: M_{unl} is expected to show concurrent gains in both precision and recall, reflecting genuine knowledge recovery, while M_{never} is expected to exhibit either imbalanced responses or negligible improvement on at least one metric, consistent with a model attempting to learn a class from scratch rather than recovering suppressed representations. This asymmetry forms the basis of our class-level membership inference attack, formalized in Algorithm 1.

Algorithm 1 Single-Class MIA via Memory Jogging

Require: Model under assessment $\mathcal{M}_{\text{target}}$
Target class $c^\star$ and samples $\mathcal{D}_{c^\star}$
Jogging epochs E, learning rate η, thresholds τ_P, τ_R

Store pre-jogging model $\mathcal{M}_0 \leftarrow \mathcal{M}_{\text{target}}$
(post-unlearning, before any fine-tuning)
Initialize $\mathcal{M}_{\text{jog}} \leftarrow \mathcal{M}_0$
Fine-tune $\mathcal{M}_{\text{jog}}$ on $\mathcal{D}_{c^\star}$ for E epochs using learning rate η
Compute precision P and recall R of $\mathcal{M}_{\text{jog}}$ on $\mathcal{D}_{c^\star}$
if $P < \tau_P$ **or** $R < \tau_R$ **then**
    return "$\mathcal{M}_{\text{target}}$ was never trained on $c^\star$"
else
    return "$\mathcal{M}_{\text{target}}$ learned and subsequently unlearned $c^\star$"
end if

This concrete decision rule is used as the operational attack in our experimental evaluation, where we analyze how reliably it separates M_{unl} from M_{never} across frameworks and datasets.

Chapter 4

Implementation

4.1 Overview

The implementation consists of two distinct workflows, each driven by its own configuration file. The first handles model training and the application of MU frameworks, controlled by `config.yaml` and orchestrated through `workflow.py`. The second runs the memory jogging attack pipeline, controlled by `attack_config.yaml` and executed through `attack.py`. Both workflows are built in PyTorch and share common model loading and architecture utilities defined in `model.py`. Before launching the training workflow, `main.py` performs a dependency verification step by parsing all core module files using Python's AST module and confirming that every detected third-party import is installed in the current environment.

4.2 Model Training and Unlearning

4.2.1 Architecture Adaptations

ResNet-18 [21] was used as the primary convolutional architecture across all datasets. For the 32×32 datasets (MNIST, SVHN, CIFAR-10, and CIFAR-100), the standard ResNet-18 architecture was modified to better suit the smaller input resolution. Specifically, the first convolutional layer was changed from a 7×7 kernel with stride 2 to a 3×3 kernel with stride 1 and padding 1, and the subsequent max-pooling layer was replaced with an identity mapping. Without this modification, the early layers would downsample 32×32 inputs too aggressively, leaving very little spatial resolution for deeper layers to process. For VGGFace100, whose images are resized to 224×224, the standard architecture was used without modification. In both cases, the final fully connected layer was replaced with a new linear layer matching each dataset's number of output classes.

For ViT, a `ViTWrapper` class was implemented to provide a consistent interface with the rest of the codebase. The wrapper loads the ViT-Base/16 configuration from Hugging Face Transformers, with 12 attention heads, 12 hidden layers, and a hidden dimension of 768. The classification head is reset for each dataset's number of classes. A non-trivial engineering challenge arose with model cloning: PyTorch's standard `deepcopy` fails on ViTWrapper because the `_features` attribute caches the CLS token output from the last forward pass as a non-leaf computation graph node, which cannot be copied. A custom `_safe_deepcopy_model` function was implemented to handle this by temporarily setting all such non-leaf tensor attributes to `None` before copying, then restoring them on the original model. During forward passes, the CLS token output is stored in `model._features` to enable feature extraction without requiring external hooks, which are incompatible with the Hugging Face forward pass structure.

4.2.2 Experiment Orchestration

The training and unlearning workflow is managed by an `ExperimentManager` class that orchestrates the full multi-class, multi-run experimental loop. For each target class, it resets the model to its trained checkpoint, applies the configured MU framework, evaluates the result, and logs the output. For 10-class datasets the loop iterates over all classes; for 100-class datasets it samples 10 classes uniformly at random. Each configuration is repeated for multiple independent runs to support aggregate reporting. The `ExperimentManager` uses a profile-based configuration system, where each profile specifies all architecture, dataset, and hyperparameter settings for a given model-dataset combination, covering all 10 pairs used in the experiments.

4.2.3 Model Scale

Producing the full set of model checkpoints required for the experimental evaluation was a substantial undertaking. For each of the 5 datasets and 2 architectures, a base model was trained on all classes. For 10-class datasets, one M_{never} model was trained per class. For 100-class datasets, one M_{never} model was trained per each of the 10 randomly selected target classes. This yields 100 M_{never} checkpoints in total across all dataset-architecture combinations. Each base model was then subjected to all three MU frameworks, once per target class, producing $5 \times 2 \times 3 \times 10 = 300$ unlearned model checkpoints. Combined with base and untrained models, the total number of saved checkpoints across the full experimental setup exceeds 400.

4.2.4 UNSIR Implementation

UNSIR [46] proceeds in three sequential phases. In the noise generation phase, for each target class y_f, a learnable noise tensor is initialized from a standard normal distribution and optimized via gradient ascent to maximize the model's cross-entropy loss on y_f, with an ℓ_2 regularization term to prevent the noise from growing unboundedly. The loss used is $-\mathcal{L}_{CE}(\theta, x_{\text{noise}}, y_f) + 0.1 \cdot \|x_{\text{noise}}\|^2$, yielding anti-samples that act as adversarial inputs targeting the forgotten class's representation in the model's latent space.

In the impairment phase, the model is fine-tuned on these synthetic anti-samples. Crucially, retain samples are interleaved with the noise samples within the same training loop, rather than being processed separately, so that disrupting the forgotten class's representations does not collapse utility on the retained classes. In the repair phase, the impaired model is fine-tuned for one epoch on a subset of retained data using a separate optimizer to restore retained class performance. Both the impairment and repair phases support separate learning rates for the backbone and the classification head, configured independently via `impair_lr_head` and `repair_lr_head`. UNSIR requires no access to any samples from the forget set during noise generation, making it a zero-glance method.

4.2.5 SalUn Implementation

SalUn [13] operates in two stages. In the first stage, a saliency mask is constructed by accumulating the absolute gradients of all model parameters on the forget set under gradient ascent, using a negative cross-entropy loss to identify which weights are most responsible for the forgotten class's predictions. All gradient magnitudes are then pooled globally and the top $k\%$ of weights by magnitude are marked as salient, where k is controlled by the `saliency_threshold` parameter (set to 0.5 in our experiments).

In the second stage, the forget set samples are relabeled uniformly at random using a `RandomLabelDataset` wrapper, which replaces each sample's true label with a randomly drawn class index on-the-fly without storing any extra data. The model is then fine-tuned on an interleaved stream of these randomly relabeled forget samples and original retain samples using SGD with momentum and weight decay. The saliency mask is applied by zeroing the gradients of all non-salient weights after each backward pass and before the optimizer step, ensuring that only the weights most strongly associated with the forgotten class are modified.

4.2.6 LTU Implementation

LTU [24] was the most involved framework to implement, as no public code was available at the time of writing. The implementation was built entirely from scratch based on the

paper's description and validated by reproducing the authors' reported results on random sample unlearning before applying it to the class-based setting of this thesis. Several hyperparameters, including the inner and outer loop learning rates, the number of MI models, and the retain subset ratio, were not specified in the original paper and were determined empirically.

LTU proceeds in three stages. In the first stage, an ensemble of k membership inference models is pre-trained before the meta-loop begins. Each MI model is a three-layer MLP that takes the softmax output of the target model as input and predicts a binary membership label. The ensemble is trained on forget set samples as members and retain set samples as non-members using binary cross-entropy loss. Having multiple MI models with different random initializations produces diverse forgetting feedback signals, following the paper's insight that different MI models exploit different statistical patterns to detect membership.

In the second stage, support and query sets are constructed. The support set consists of forget class samples with randomly reassigned labels, where the random labels are re-drawn at the start of each meta-epoch to prevent the model from learning a static wrong mapping. Three types of query sets are constructed from a subset of the retain data: samples nearest to the forget class in feature space, samples nearest by label similarity, and randomly drawn samples. This diversity ensures that the meta-test feedback captures generalizable retention knowledge rather than memorizing specific examples.

In the third stage, the meta-optimization loop runs for the configured number of epochs. Each iteration begins with the meta-tune step, in which a temporary copy of the model is updated for one inner step on the support set, simulating aggressive forgetting. The meta-test step then evaluates this temporary model on all query sets, accumulating retain gradients that reflect how much generalizable knowledge was preserved. Separately, the MI models are queried on the forget set to produce a differentiable forgetting gradient. These two gradient signals are then combined in the gradient harmonization step: if the cosine similarity between the retain gradient g_r and the forget gradient g_f is negative, indicating conflict, the conflicting component of g_f is projected out by

$$g'_f = g_f - \frac{g_f \cdot g_r}{\|g_r\|^2} g_r \tag{4.1}$$

yielding a corrected forgetting gradient that is orthogonal to g_r. The final harmonized gradient $G = g_r + g'_f$ is then applied to the actual model parameters via the outer-loop update with gradient clipping.

4.3 Memory Jogging Attack Pipeline

The attack pipeline is implemented in `attack.py` and driven by `attack_config.yaml`. At the top level, the configuration specifies the active architecture, dataset, and MU framework, which together determine which model checkpoints and dataset paths are loaded.

4.3.1 Per-Class Fine-Tuning

For each target class $c^\star$, the pipeline loads a pair of checkpoints: the unlearned model M_{unl} produced by the selected MU framework, and the corresponding M_{never} model. A class-filtered data loader is constructed by extracting only the samples belonging to $c^\star$ from the training dataset using a configurable fraction of the available data. Both models are then independently fine-tuned on this subset for a fixed number of epochs using either AdamW or SGD depending on the configuration, other hyperparameters include gradient clipping and weight decay. The optimizer uses a small learning rate to help nudge the M_{unl} parameters in the direction of restoring the forgotten class's representation without allowing M_{never} to learn it from scratch, making the models distinguishable.

4.3.2 Metric Collection and Logging

Before and after fine-tuning, both models are evaluated on the full test set. Per-class precision, recall, and F1-score are computed using scikit-learn's per-class metric computation with `average=None`, returning an independent metric array for every class. Results are stored in two CSV files: a core file that records metrics only for the target class $c^\star$, and a detailed file that records metrics for all classes, enabling post-hoc analysis of whether memory jogging on $c^\star$ causes any collateral effect on retained classes.

Each experiment is repeated for a configurable number of independent runs, with fresh model copies loaded from disk for each run to ensure independence between trials. GPU memory is explicitly freed between runs via `gc.collect()` and `torch.cuda.empty_cache()` to prevent memory accumulation across the large number of model pairs evaluated.

4.3.3 Swapped Label Experiment

The pipeline also supports a swapped label experiment to test robustness to label-class misalignment. In this setting, two classes are forgotten simultaneously, and during memory jogging the samples from one forgotten class are assigned the label of the other. A derangement function ensures that no class is ever mapped to itself. This tests whether the

recovery signal is truly class-specific or whether it arises as an artifact of the output node mapping, and is discussed further in Sec. 5.

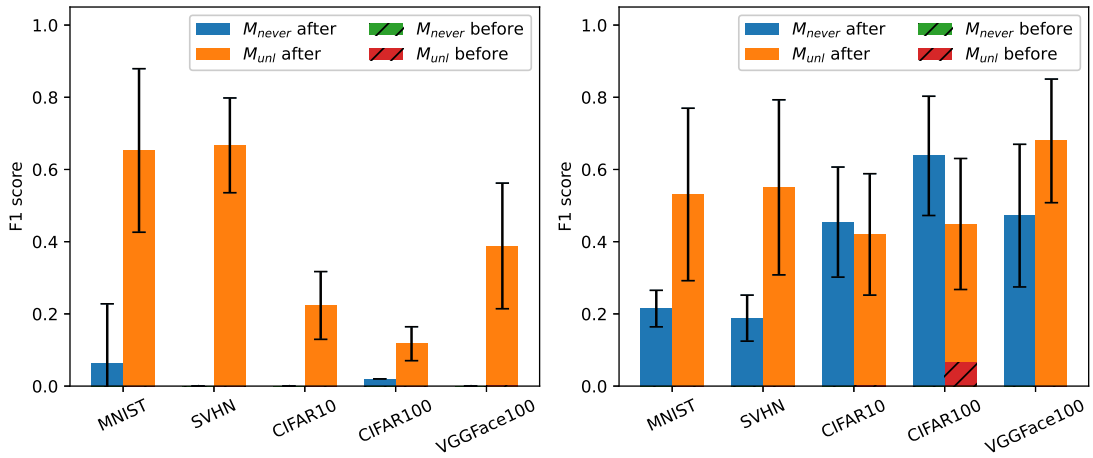
Chapter 5

Evaluation

5.1 Main Results

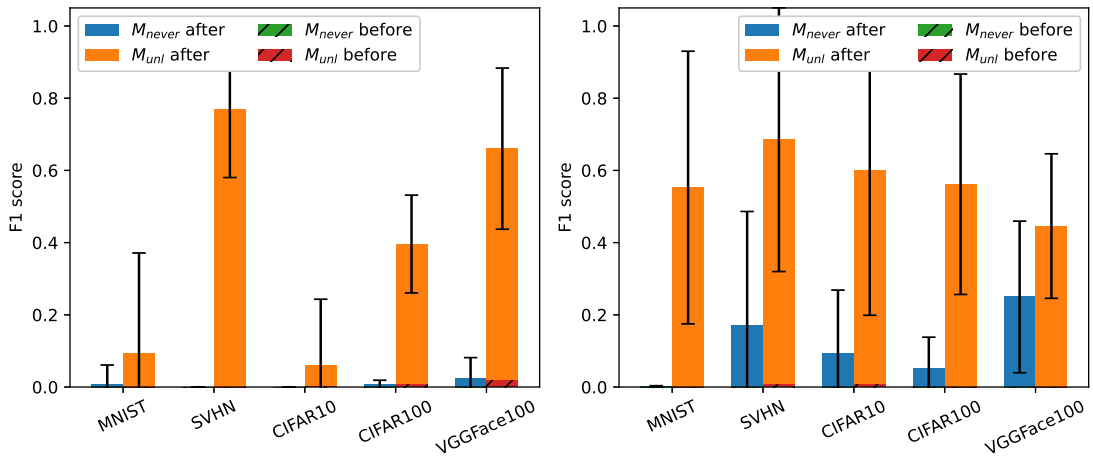
5.1.1 F1-Score Analysis

Fig. 5.1 reports the mean F1 score of both M_{never} and M_{unl} before and after memory jogging, averaged across all target classes, for each framework-architecture combination across all five datasets. Each bar group shows the pre-jogging baseline alongside the post-jogging result, allowing a direct comparison of how much each model type responds to the fine-tuning signal. Before jogging, both model states produce near-zero F1 on the target class by construction, confirming that standard evaluation would not distinguish them. After jogging, the picture changes. Across all three MU frameworks and both architectures, memory jogging produces a systematic separation between the two model states, with M_{unl} achieving higher post-jogging F1 in the majority of settings. This separation is the core signal our attack exploits: a model that previously encoded a class and was later subjected to approximate unlearning recovers measurably faster than one that was simply never trained on that class. The strength of this separation, however, varies considerably across frameworks, and understanding this variation is the focus of the analysis that follows.



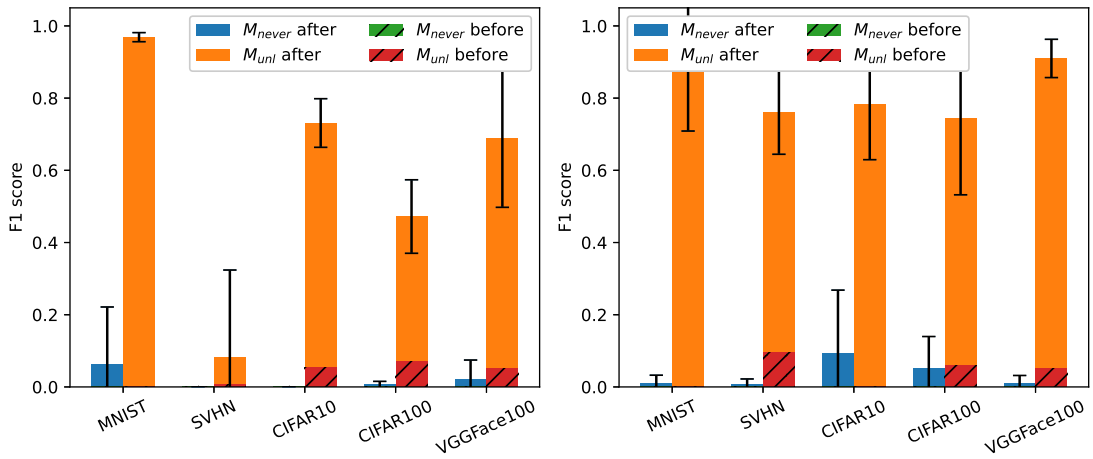
(a) UNSIR: ResNet-18

(b) UNSIR: ViT



(c) SalUn: ResNet-18

(d) SalUn: ViT



(e) LTU: ResNet-18

(f) LTU: ViT

Figure 5.1: Per-dataset F1 score before and after memory jogging for M_{never} and M_{unl}, averaged over all target classes, across ResNet-18 and ViT models. M_{unl} models were produced using UNSIR, SalUn, and LTU. Bars report the mean F1 score and standard deviation across runs.

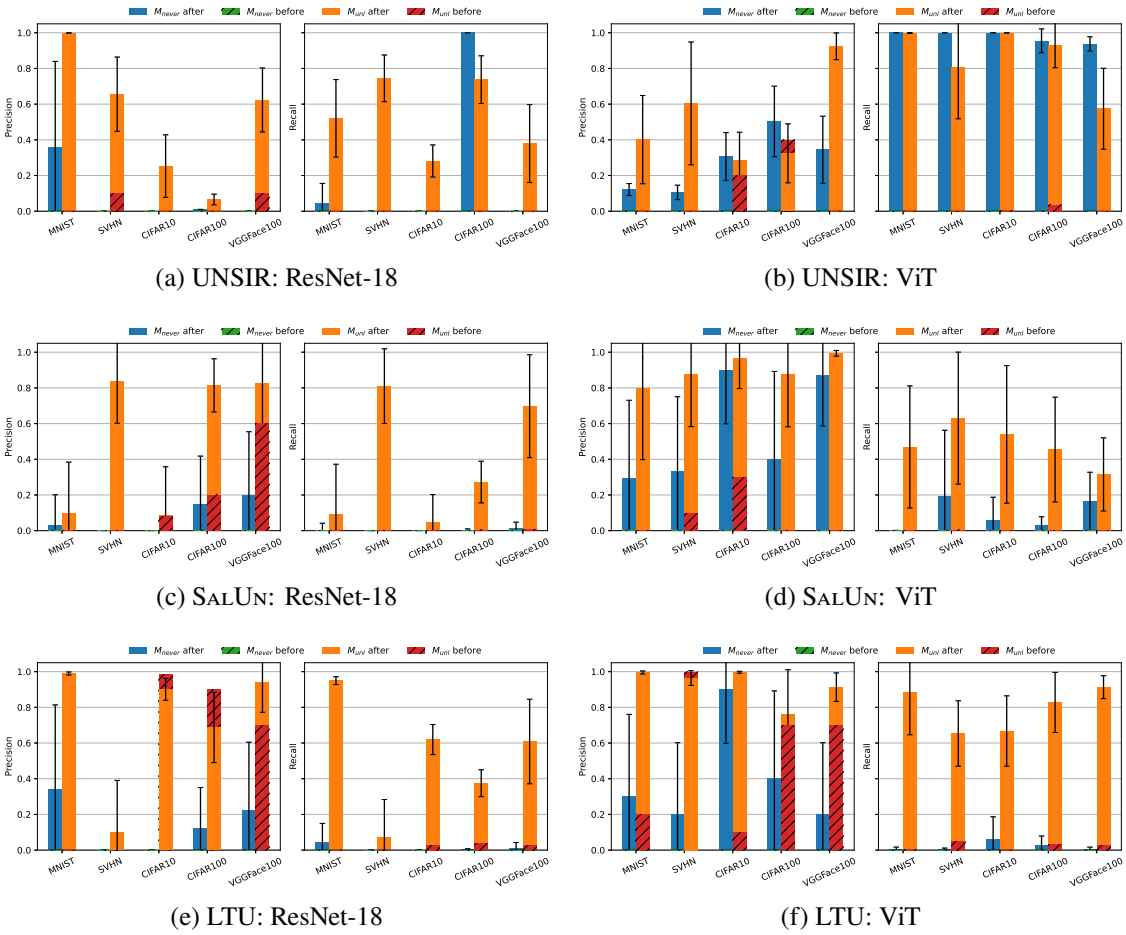
Under UNSIR, the picture is architecture-dependent. On ResNet-18, M_{never} shows virtually no change in F1 after jogging, while M_{unl} recovers measurably across all five datasets, producing clear separability. On ViT, the gap narrows: M_{never} exhibits modest post-jogging improvements on some datasets, making the two model states less consistently distinguishable. Despite this, differences remain observable across both architectures, and the relative ordering between M_{unl} and M_{never} is preserved in the large majority of cases.

SalUn produces substantially stronger and more consistent separation. For both ResNet-18 and ViT, M_{unl} attains markedly higher post-jogging F1 across all datasets, while M_{never} remains at low levels. Unlike UNSIR, there are no dataset-architecture combinations where M_{never} matches or exceeds M_{unl} after jogging. This suggests that the saliency-guided weight updates in SalUn suppress class-level representations more selectively than the anti-sample approach in UNSIR, but leave residual signals that are reliably amplified by fine-tuning.

LTU yields the strongest and most uniform separability of the three frameworks. Across all datasets and both architectures, M_{unl} reaches high post-jogging F1 while M_{never} stays near chance level, with minimal overlap between the two states. Paradoxically, this makes LTU the most vulnerable of the three frameworks to memory jogging, despite performing best on standard unlearning metrics. We return to this point in Sec. 5.2. Overall, the dominant factor governing the strength of the signal is the MU framework itself rather than the dataset or architecture: dataset complexity, measured by the number of classes or semantic difficulty, does not reliably predict resistance to memory jogging, and SalUn and LTU produce strong separability across all five datasets while UNSIR shows variability primarily explained by the choice of architecture.

5.1.2 Precision-Recall Decomposition

F1-score is an aggregate measure and can be driven by precision alone, recall alone, or both. To understand which pattern underlies the observed separability, Fig. 5.2 decomposes post-jogging performance into its constituent precision and recall components.



(a) UNSIR: ResNet-18 (b) UNSIR: ViT

(c) SalUn: ResNet-18 (d) SalUn: ViT

(e) LTU: ResNet-18 (f) LTU: ViT

Figure 5.2: Per-dataset precision and recall before and after memory jogging for M_{never} and M_{unl}, averaged over all target classes, for ResNet-18 and ViT. M_{unl} models were produced using UNSIR, SalUn, and LTU. Bars report the mean and standard deviation across runs.

This decomposition reveals a key asymmetry: M_{unl} consistently shows concurrent gains in both precision and recall after jogging, while M_{never} tends to show gains in only one of the two, or neither. Concurrent improvement in both metrics is the signature of genuine knowledge recovery, whereas single-axis improvement is more consistent with overfitting or spurious sensitivity.

Under UNSIR on ResNet-18, the asymmetry is particularly clear. The few non-zero F1 cases for M_{never} arise from highly imbalanced precision-recall behavior: on MNIST, recall is near zero while precision is moderate, meaning the model rarely predicts the forgotten class but is occasionally correct when it does. On CIFAR-100, the opposite holds, with recall near one and precision low, indicating indiscriminate prediction of the forgotten class for nearly all inputs, a pattern characteristic of overfitting during jogging. In contrast, M_{unl} recovers with balanced precision and recall gains across all datasets. On ViT under UNSIR, the picture is less consistent: on MNIST and SVHN, M_{never} achieves high recall but precision near the random-guessing baseline of approximately 0.1, while

on CIFAR-10 and CIFAR-100 both models show similar profiles, and on VGGFace100 the behaviors diverge again with M_{never} trading precision for recall while M_{unl} achieves the opposite balance.

Under SalUn, the behavior is markedly more stable. Across all datasets and both architectures, M_{unl} shows balanced improvements in precision and recall, while M_{never} may reach moderate precision on some configurations but without a corresponding recall gain. Unlike UNSIR, there are no recall-dominated outliers for M_{never} under SalUn, suggesting that the amplified residual signal reflects genuine class recovery rather than spurious sensitivity.

LTU exhibits the strongest and most uniform precision-recall separation. Across all datasets and architectures, M_{unl} reaches near-saturated precision alongside substantial recall, while M_{never} stays at low recall without balanced improvement along either axis. The absence of recall-dominated behavior for M_{never} under LTU indicates that memory jogging does not push models that never encoded the class toward overprediction while improvement is concentrated almost exclusively in M_{unl}. Taken together, the precision-recall decomposition confirms that the F1 separability observed in the previous section is driven by genuine knowledge recovery in M_{unl} rather than an artifact of how the aggregate metric is computed.

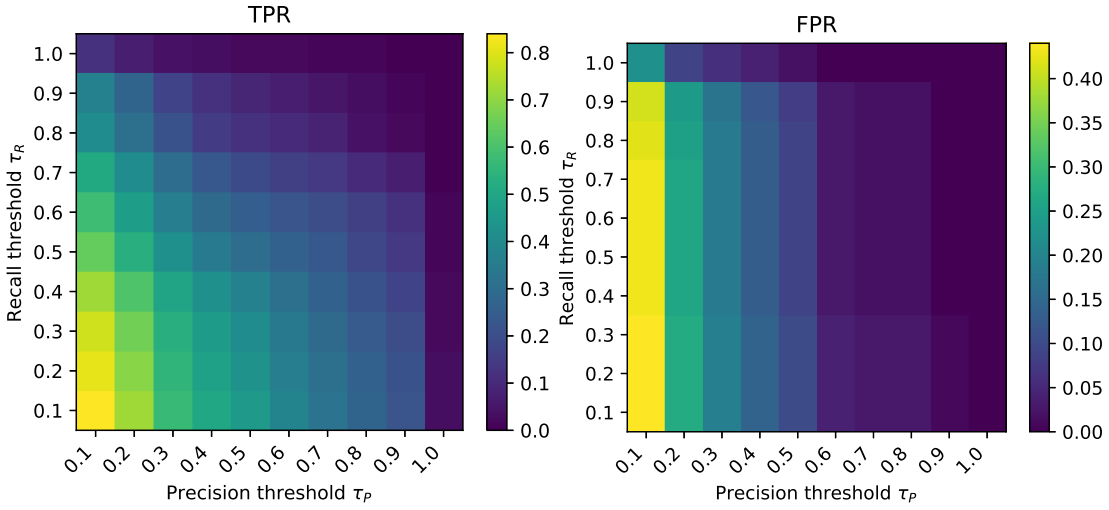
5.1.3 Threshold-Based Inference

The precision-recall results establish that M_{unl} and M_{never} respond differently to memory jogging. We now ask how an adversary can translate this difference into a reliable decision. The approach is straightforward: after jogging, the adversary applies thresholds (τ_P, τ_R) to the post-jogging precision and recall of the target model. Table 5.1 defines the confusion outcomes used throughout this section.

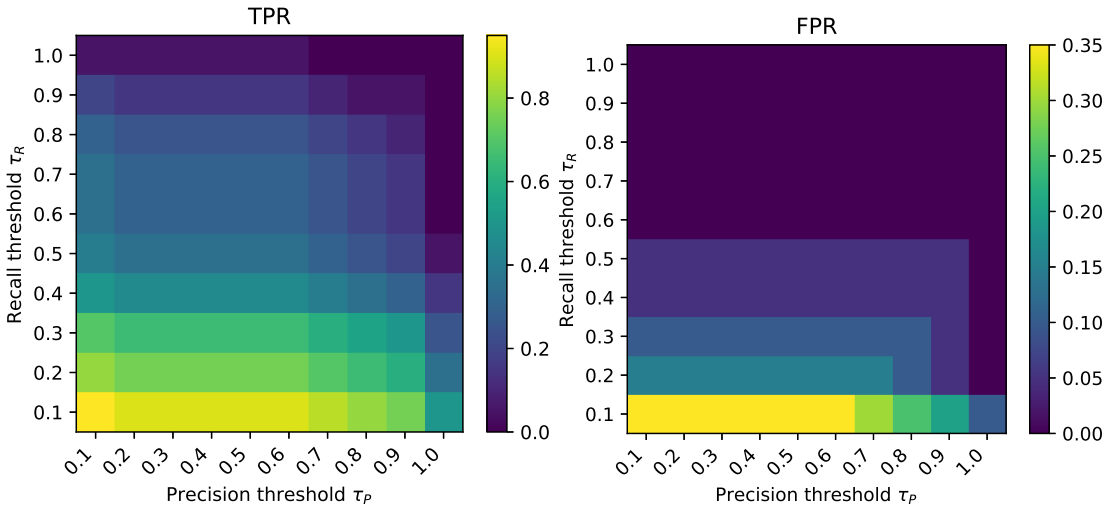
Table 5.1: Outcome definitions for class-level distinguishability between M_{never} and M_{unl}.

Model State	**Diagnostic Outcome**	**Result**
M_{unl}	Class predicted present	True Positive (TP)
M_{unl}	Class predicted absent	False Negative (FN)
M_{never}	Class predicted present	False Positive (FP)
M_{never}	Class predicted absent	True Negative (TN)

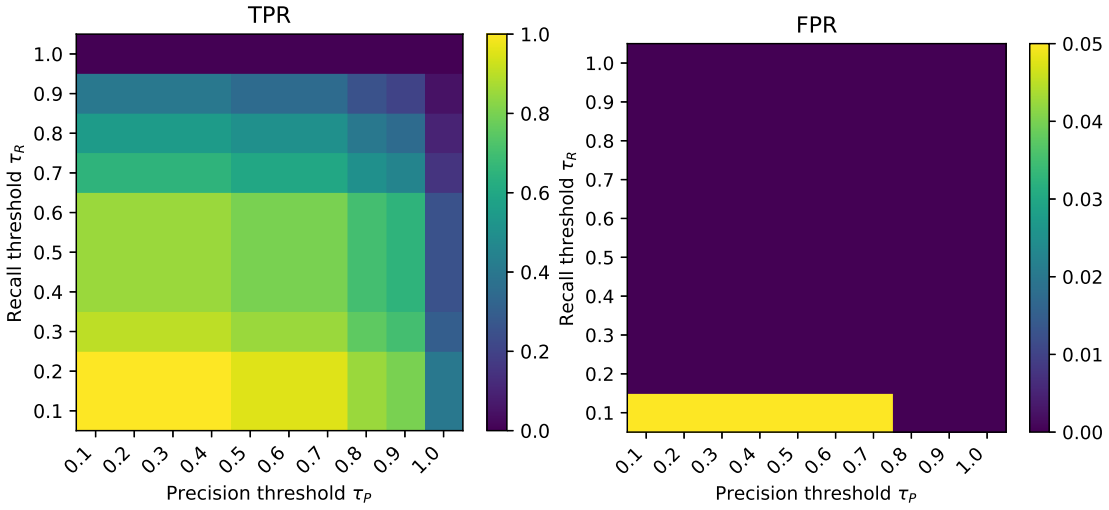
If both post-jogging precision and recall exceed their respective thresholds, the model is classified as M_{unl}; otherwise as M_{never}. Varying these thresholds traces out a space of operating points with different trade-offs between true positive rate (TPR) and false positive rate (FPR), as shown in Fig. 5.3.



(a) UNSIR



(b) SALUN



(c) LTU

Figure 5.3: Confusion behavior of the class-level decision rule across all architectures and datasets for UNSIR, SALUN, and LTU. Each heatmap reports the aggregated TPR and FPR obtained by applying precision and recall thresholds (τ_P, τ_R) to post-jogging metrics. Counts are summed across all model-dataset combinations and normalized to produce rates.

For UNSIR, TPR decays gradually as the thresholds tighten, while FPR remains non-negligible across a broad range of operating points. Useful operating regions exist at moderate thresholds, approximately $\tau_P, \tau_R \in [0.3, 0.5]$, where TPR reaches 0.5–0.7 while FPR is partially suppressed but typically remains above 0.1. Outside this range, stricter thresholds degrade TPR rapidly while more permissive ones push FPR up.

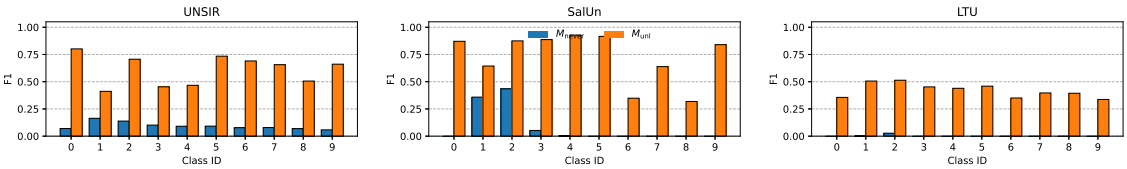
For SalUn, the heatmaps reveal a substantially larger and more stable operating region. Across a broad range of thresholds, approximately $\tau_P, \tau_R \in [0.3, 0.6]$, TPR exceeds 0.7 while FPR drops below 0.1. This extended region means that an adversary targeting SalUn-unlearned models does not need to tune the thresholds precisely — the residual signal is consistently amplified enough that a wide range of decision criteria all yield strong discriminative power.

LTU offers the most exploitable operating space. TPR remains high even at strict settings with $\tau_P, \tau_R \geq 0.5$, while FPR is near zero for almost all threshold combinations except the most permissive recall settings. This yields a large, well-separated operating region where the adversary can reliably distinguish the two model states under conservative thresholding. Per-architecture heatmaps for ResNet-18 and ViT are reported in the supplementary material. Taken together, the heatmap analysis confirms that the precision-recall asymmetry identified in the previous section translates directly into a practical and robust single-model diagnostic, with the size and reliability of the exploitable operating region increasing from UNSIR to SalUn to LTU.

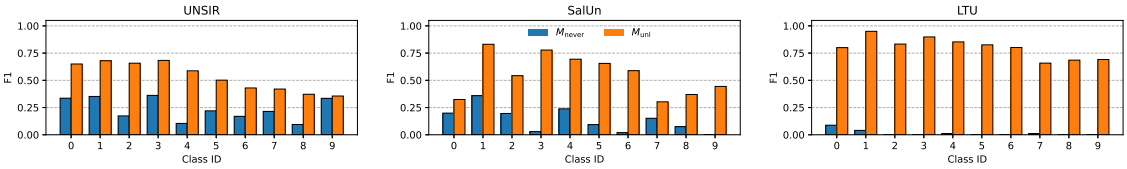
5.2 Additional Findings

5.2.1 Per-Class Vulnerability

The results in the previous section aggregate performance across all target classes, which can mask meaningful variation within a dataset. Fig. 5.4 breaks down post-jogging F1 by class for SVHN and VGGFace100, with analogous results for MNIST, CIFAR-10, and CIFAR-100 deferred to the appendix.



(a) SVHN



(b) VGGFace100

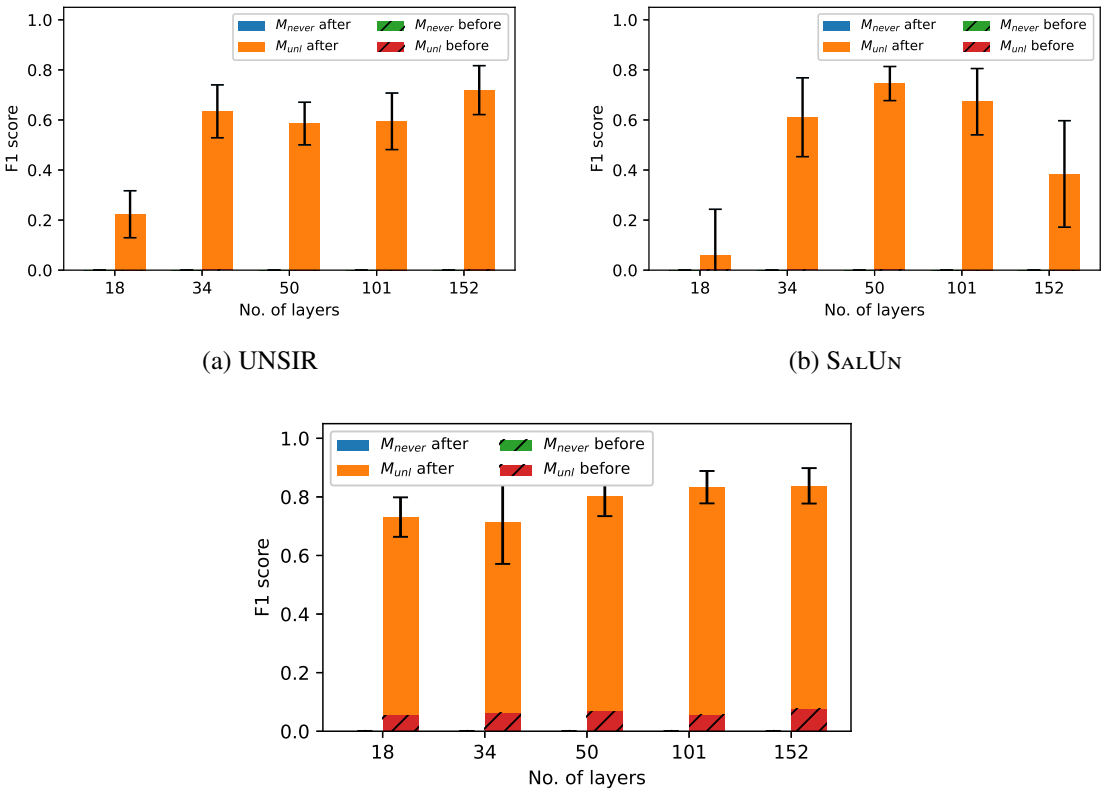
Figure 5.4: Per-class F1 scores after memory jogging for M_{never} and M_{unl} under UNSIR, SalUn, and LTU for SVHN and VGGFace100. Additional datasets are reported in the appendix.

The heterogeneity is substantial. Within a single dataset and framework, some classes consistently yield high post-jogging F1 for M_{unl} while others show only modest or inconsistent gains. This suggests that approximate unlearning does not suppress residual class information uniformly across the label space. Some classes appear more deeply encoded in the model's weights, either because they are harder to learn or because their representations are more entangled with those of retained classes, and therefore more of their latent structure survives the heuristic weight updates applied during unlearning.

This heterogeneity is most pronounced under UNSIR and SalUn. Under LTU, per-class behavior is more uniform: M_{unl} consistently achieves high F1 across most classes, which is what drives LTU's strong aggregate results but also means that an adversary has a high probability of success regardless of which class they target. The class-level variability persists independently of dataset size — even in CIFAR-100 and VGGFace100, a consistent subset of classes yields distinctly stronger post-jogging signals than others. Post-unlearning vulnerability is therefore class-dependent, shaped by how individual classes are represented within the model rather than by dataset scale alone.

5.2.2 Network Depth Variability

A natural question is whether network depth, as a proxy for model capacity, systematically affects post-jogging vulnerability. Fig. 5.5 addresses this by reporting F1 scores for M_{never} and M_{unl} across ResNet variants of increasing depth trained on CIFAR-10.



(a) UNSIR

(b) SalUn

(c) LTU

Figure 5.5: Post-jogging F1 scores for M_{never} and M_{unl} as a function of network depth for ResNet variants trained on CIFAR-10 under each MU framework.

Under UNSIR and SalUn, post-jogging F1 for M_{unl} fluctuates across depths without a clear trend. Neither shallower nor deeper models are consistently more or less vulnerable, the relative ordering between M_{unl} and M_{never} is preserved at all depths, but the magnitude of the gap varies non-monotonically. Under LTU, M_{unl} maintains consistently high post-jogging F1 regardless of depth, while M_{never} remains negligible across all configurations. The robustness of this signal to architectural variation further confirms that LTU's vulnerability to memory jogging is a property of the unlearning algorithm itself rather than an artifact of a specific model capacity. Network depth, in summary, does not exhibit a clear or monotonic relationship with vulnerability, and the dominant factor remains the choice of MU framework.

5.3 Beyond Single-Class Unlearning

The analyses above assume a single-class unlearning setting in which the adversary knows which output node corresponds to the forgotten class. In practice, when multiple classes are unlearned simultaneously, the adversary may not know which output node maps to

which forgotten class. We investigate whether class-level membership inference remains effective under this label-class misalignment.

We consider a setting in which two classes are forgotten simultaneously. During memory jogging, the adversary uses samples from one forgotten class but assigns them the label of the other, creating an intentional mismatch between the data and the output node being targeted. This captures the realistic scenario where the adversary knows which classes were unlearned but not their original correspondence to output indices.

Table 5.2: Average post-jogging F1 score under correct versus swapped label-class mappings, reported as mean ± std across ten independent runs on SVHN with ResNet-18.

MU Framework	**F1 (correct)**	**F1 (swapped)**
UNSIR	0.686 ± 0.074	0.609 ± 0.101
SalUn	0.745 ± 0.079	0.604 ± 0.065
LTU	0.926 ± 0.015	0.508 ± 0.055

Table 5.2 reports post-jogging F1 under both correct and swapped mappings across all three frameworks. Despite the intentionally incorrect label assignment, M_{unl} recovers substantial classification performance in all cases. The swapped mapping consistently reduces F1 relative to the correct mapping, but a meaningful class-level signal remains for UNSIR and SalUn. For LTU, the drop is larger, from 0.926 to 0.508, suggesting that the stronger residual signal in LTU-unlearned models is more sensitive to precise label alignment. Nevertheless, even at 0.508, the signal remains well above what would be expected from a model that never learned the class. Class-level membership inference therefore degrades gracefully under misalignment, remaining effective across all three frameworks even without knowledge of the original label-to-output mapping.

Chapter 6

Related Work

Chen et al. [9] proposed an MIA in the MU setting, where the adversary has access to two versions of a model: an original model trained on a dataset containing a target sample, and an unlearned model from which the influence of that specific sample has been removed. Their goal is to determine whether that target sample was part of the original training set, using access to both model versions simultaneously. Our setting is fundamentally different: the adversary observes only a single post-unlearning model instance that already satisfies existing MU evaluation criteria, with no access to any prior version. Given two candidate models that should not represent a target class, one never trained on it and one that underwent approximate unlearning, we ask whether residual class-level information can still be detected and amplified to distinguish between them.

Hu et al. [23] investigate the reversibility of MU in the context of large language models (LLMs). Using a benchmark of memorized factual knowledge, they show that even after unlearning, LLMs can often regenerate forgotten content when prompted strategically, suggesting that alternative activation pathways may suffice to surface suppressed information. Our work shares the core intuition that approximate unlearning suppresses rather than erases, but studies it in image classification models under a different threat model that permits direct weight modification through fine-tuning, rather than inference-time prompting.

Most closely related to our work is the concurrent and independent study by Xiao et al. [50], who propose the Reminiscence Attack (ReA), published at ICCV 2025. ReA reaches the same fundamental conclusion as our work: approximate MU methods leave residuals in the model's loss landscape that can be exploited through targeted fine-tuning to infer membership of unlearned data. Both attacks are version-independent and both demonstrate that standard evaluation metrics fail to capture this vulnerability. The convergence of two independent lines of work on the same finding strengthens the case that residual-based privacy leakage is a genuine and systematic property of approximate MU rather than an artifact of any particular experimental setup.

The two approaches differ in scope and methodology in ways that are worth noting. ReA targets both sample-wise and class-wise membership inference and additionally addresses concept unlearning in generative models, while our work focuses exclusively on class-level inference in image classification. On the practical side, Xiao et al. propose a learning rate search strategy based on averaging across candidate values to address the sensitivity of fine-tuning to the choice of learning rate, a challenge we also encountered. Their paper also introduces OUR (Orthogonal Unlearning and Replay), a new approximate MU framework designed specifically to resist ReA by enforcing orthogonality between the hidden representations of unlearned data and their original values, followed by a convergence stabilization phase. OUR has not yet been evaluated against our attack. Based on the mechanistic similarity between ReA and our memory jogging procedure, and given that both exploit the same class of loss-landscape residuals, it is plausible that OUR would also reduce vulnerability to our attack, though this remains an open empirical question and an interesting direction for future work.

Chapter 7

Conclusion

Approximate machine unlearning has emerged as a practical response to the computational infeasibility of full retraining, and current methods do succeed in suppressing a model's immediate behavior on forgotten data. However, suppression is not erasure. This thesis provides systematic empirical evidence that the residual traces left behind by approximate MU methods are not merely theoretical concerns, they are consistently detectable and exploitable across diverse datasets, architectures, and unlearning frameworks.

The central finding is that a small number of targeted fine-tuning steps on the forgotten class is sufficient to produce a measurable separation between a model that underwent unlearning and one that was never trained on that class in the first place. This result is not surprising in hindsight. Approximate MU methods apply heuristic weight-space updates for a limited number of epochs, without formal guarantees that the underlying representations of the forgotten data have been removed. The reduction in accuracy on the forgotten class, which these methods reliably achieve, reflects a change in the model's output behavior rather than a genuine elimination of the learned structure. When that structure is given even a modest gradient signal pointing back toward it, the model recovers measurably faster than one that was never exposed to the data in the first place.

A particularly instructive finding is that standard unlearning performance and resistance to memory jogging are effectively decoupled. A framework that scores well on accuracy-based evaluation does not necessarily leave fewer recoverable traces in its weight space, and the reverse also holds. This independence reveals the core of the evaluation gap: passing existing checks does not imply robustness against an adversary who can apply targeted weight updates after unlearning. Current practice evaluates the model in isolation at a fixed point in time, but the more relevant question is what the model reveals when pushed. Even when the adversary lacks precise knowledge of the original class-to-output mapping, the attack degrades gracefully, remaining effective across all three frameworks under label misalignment. The residual signal is robust enough to survive imprecision in how the attack is mounted, which makes the threat practical rather than merely theoretical.

Taken together, these findings suggest that the field's current definition of successful unlearning is insufficient. A model that shows no accuracy on the forgotten class under standard evaluation is not the same as a model that has truly forgotten it. Closing this gap calls for evaluation protocols that explicitly account for post-unlearning robustness, testing not just what a model outputs in isolation but how it responds when subjected to further updates.

One direction this thesis did not pursue, primarily due to the scale of empirical work required, is the design of an unlearning framework built with this threat in mind. Developing and validating such a method is a natural and important next step, and the findings here provide a clear target: a robust unlearning procedure must not only suppress output behavior but ensure that the forgotten class leaves no recoverable structure in the model's weight space.

Bibliography

[1] A. Achille, M. Rovere, and S. Soatto. Critical learning periods in deep networks. In *ICLR*, 2019.

[2] T. Baumhauer, P. Schöttle, and M. Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *Machine Learning*, 111(9):3203–3226, 2022.

[3] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. Machine unlearning. In *S&P*, pages 319–337, 2021.

[4] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *S&P*, pages 463–480, 2015.

[5] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2001.

[6] G. C. Cawley and N. L. Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural Networks*, 17(10):1467–1475, 2004.

[7] G. C. Cawley and N. L. Talbot. Efficient approximate leave-one-out cross-validation for kernel logistic regression. *Machine Learning*, 71(2):243–264, 2008.

[8] M. Chen, W. Gao, G. Liu, K. Peng, and C. Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *CVPR*, pages 7766–7775, 2023.

[9] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang. When machine unlearning jeopardizes privacy. In *CCS*, pages 896–911, 2021.

[10] C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot. Label-only membership inference attacks. In *ICML*, pages 1964–1974, 2021.

[11] V. S. Chundawat, A. K. Tarun, M. Mandal, and M. Kankanhalli. Zero-shot machine unlearning. *IEEE Trans. Inf. Forensics Secur.*, 18:2345–2354, 2023.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] C. Fan, J. Liu, Y. Zhang, E. Wong, D. Wei, and S. Liu. SalUn: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *ICLR*, 2024.

[14] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou. Making AI forget you: Data deletion in machine learning. *NeurIPS*, 32, 2019.

[15] A. Golatkar, A. Achille, A. Ravichandran, M. Polito, and S. Soatto. Mixed-privacy forgetting in deep networks. In *CVPR*, pages 792–801, 2021.

[16] A. Golatkar, A. Achille, and S. Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *CVPR*, pages 9304–9312, 2020.

[17] A. Golatkar, A. Achille, and S. Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *ECCV*, pages 383–398, 2020.

[18] A. S. Golatkar, A. Achille, and S. Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. *NeurIPS*, 32, 2019.

[19] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten. Certified data removal from machine learning models. In *ICML*, pages 3832–3842, 2020.

[20] J. Hayes, L. Melis, G. Danezis, and E. De Cristofaro. Logan: Membership inference attacks against generative models. In *PETS*, volume 2019, pages 133–152, 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[22] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[23] S. Hu, Y. Fu, S. Wu, and V. Smith. Unlearning or obfuscating? jogging the memory of unlearned LLMs via benign relearning. In *ICLR*, 2025.

[24] M. H. Huang, L. G. Foo, and J. Liu. Learning to unlearn for robust machine unlearning. In *ECCV*, pages 202–219. Springer, 2025.

[25] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot. High accuracy and high fidelity extraction of neural networks. In *USENIX Security*, pages 1345–1362, 2020.

[26] L. Jian, S. Shen, J. Li, X. Liang, and L. Li. Budget online learning algorithm for least squares SVM. *IEEE Transactions on Neural Networks and Learning Systems*, 28(9):2076–2087, 2017.

[27] M. Kearns. Efficient noise-tolerant learning from statistical queries. *JACM*, 45(6):983–1006, 1998.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] S. Kodge, G. Saha, and K. Roy. Deep unlearning: Fast and efficient gradient-free class forgetting. *Transactions on Machine Learning Research*, 2024.

[30] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[32] W.-H. Lee, B. J. Ko, S. Wang, C. Liu, and K. K. Leung. Exact incremental and decremental learning for ls-svm. In *ICIP*, pages 2334–2338, 2019.

[33] Y. Li, J. Zhang, Y. Liu, and C. Chen. Class-wise federated unlearning: Harnessing active forgetting with teacher–student memory generation. *Knowledge-Based Systems*, 316:113353, 2025.

[34] J. Liu, P. Ram, Y. Yao, G. Liu, Y. Liu, P. SHARMA, S. Liu, et al. Model sparsity can simplify machine unlearning. *NeurIPS*, 36, 2024.

[35] B. Mirzasoleiman, A. Karbasi, and A. Krause. Deletion-robust submodular maximization: Data summarization with "the right to be forgotten". In *ICML*, pages 2449–2458, 2017.

[36] M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *S&P*, pages 739–753, 2019.

[37] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, volume 2011, page 7, 2011.

[38] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, pages 343–347, 2014.

[39] T. T. Nguyen, T. T. Huynh, Z. Ren, P. L. Nguyen, A. W.-C. Liew, H. Yin, and Q. V. H. Nguyen. A survey of machine unlearning. *ACM Transactions on Intelligent Systems and Technology*, 16(5), sep 2025.

[40] T. Orekondy, B. Schiele, and M. Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR*, 2019.

[41] S. Panda, S. Sourav, et al. Partially blinded unlearning: Class unlearning for deep networks from bayesian perspective. In *AAAI*, volume 39, pages 6372–6380, 2025.

[42] A. Peste, D. Alistarh, and C. H. Lampert. SSSE: Efficiently erasing samples from trained machine learning models. In *PRIML (Poster)*, 2021.

[43] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *NDSS*, 2019.

[44] S. Schelter, S. Grafberger, and T. Dunning. Hedgecut: Maintaining randomised trees for low-latency machine unlearning. In *MOD*, pages 1545–1557, 2021.

[45] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *S&P*, pages 3–18, 2017.

[46] A. K. Tarun, V. S. Chundawat, M. Mandal, and M. Kankanhalli. Fast yet effective machine unlearning. *IEEE Trans. Neural Netw. Learn. Syst.*, 2023.

[47] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart. Stealing machine learning models via prediction APIs. In *USENIX Security*, pages 601–618, 2016.

[48] E. Ullah, T. Mai, A. Rao, R. A. Rossi, and R. Arora. Machine unlearning via algorithmic stability. In *COLT*, pages 4126–4142, 2021.

[49] Y. Wu, E. Dobriban, and S. Davidson. Deltagrad: Rapid retraining of machine learning models. In *ICML*, pages 10355–10366, 2020.

[50] Y. Xiao, Q. Ye, L. Hu, H. Zheng, H. Hu, Z. Liang, H. Li, and Y. Jiao. Reminiscence attack on residuals: Exploiting approximate machine unlearning for privacy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3058–3068, 2025.

[51] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*, pages 268–282, 2018.