Machine Unlearning in Low-Dimensional Feature Subspace

Kun Fang[1*] **Qinghua Tao**[2*] **Junxu Liu**[1†] **Yaxin Xiao**[1†]
Yulin Jin[1] **Qingqing Ye**[1] **Jian Sun**[2†] **Haibo Hu**[1]

[1]The Hong Kong Polytechnic University [2]Beijing Institute of Technology

{kun.fang,junxu.liu,samin-yaxin.xiao,qqing.ye,haibo.hu}@polyu.edu.hk
{qinghua.tao,sunjian}@bit.edu.cn yulin.jin@connect.polyu.hk

Abstract

Machine Unlearning (MU) aims at removing the influence of specific data from a pretrained model while preserving performance on the remaining data. In this work, a novel perspective for MU is presented upon low-dimensional feature subspaces, which gives rise to the potentials of separating the remaining and forgetting data herein. This separability motivates our LOFT, a method that proceeds unlearning in a LOw-dimensional FeaTure subspace from the pretrained model through principal projections, which are optimized to maximally capture the information of the remaining data and meanwhile diminish that of the forgetting data. During unlearning, LOFT simply optimizes a small-size projection matrix that can be flexibly inserted into the pretrained model, requiring just one-shot feature fetching from the pretrained model. LOFT demonstrates that effective and efficient MU can be achieved in low-dimensional feature subspaces even without repeated raw-data access or updates to the entire pretrained model. Extensive experiments validate the significantly lower computational overhead and superior unlearning performance of LOFT across diverse models, datasets, tasks, and applications. Code is publicly available.[1]

1 Introduction

Deep Neural Networks (DNNs) pretrained on large-scale data, e.g., web-crawled data, greatly advocate various applications [1, 2]. However, it also raises significant privacy concerns due to its heavy reliance on massive data. Modern data regulatory frameworks, such as GDPR [3], emphasize to preserve *the right to be forgotten* when requesting to delete particular data, e.g., private user data. To address this challenge, Machine Unlearning (MU) [4] has emerged and attracted increasing interests in recent years. MU targets dual goals: erasing the knowledge from the deleted (forgotten) data and meanwhile preserving the model performance. A gold-standard approach to MU is to retrain the model from scratch based on the remaining data only, i.e., the entire training data excluding the forgetting data [5], which is called *the exact MU*. Such a retraining process is computationally expensive, especially for modern DNNs which are commonly large-scale models trained on massive data. Hence, a surge of research on *approximate MU* [6] has been developed to approach the ideal performance of the retrained model.

Existing approximate MU methods generally update the parameters of the original model pretrained on the entire training data $\mathcal{D}$, so as to align its outputs with that of the ideal model retrained in the exact MU. The core idea here is to perform parameter updates with an optimization objective that

*Equal contribution. †Corresponding authors.

[1]https://github.com/fanghenshaometeor/unlearning-feat-subspace

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

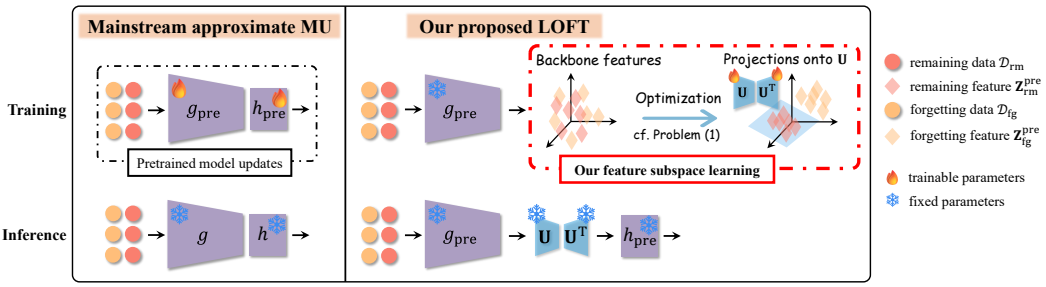


Figure 1: Overview of our proposed LOFT and mainstream approximate MU methods. After unlearning, the trained projection matrix of LOFT is incorporated into the model's forward propagation during inference, with details in Section 4.2.

jointly penalizes the performance on the forgetting data $\mathcal{D}_{\text{fg}}$ and maintains that on the remaining data $\mathcal{D}_{\text{rm}}$, with $\mathcal{D}_{\text{fg}} \bigcup \mathcal{D}_{\text{rm}} = \mathcal{D}$ and $\mathcal{D}_{\text{fg}} \bigcap \mathcal{D}_{\text{rm}} = \emptyset$. This optimization objective leads to different ways to modify the pretrained model, including fine-tuning only on $\mathcal{D}_{\text{rm}}$ [7], assigning random labels to $\mathcal{D}_{\text{fg}}$ [8], conducting gradient ascent on $\mathcal{D}_{\text{fg}}$ [9], masking those parameters sensitive to $\mathcal{D}_{\text{fg}}$ [10], distilling an additional network [11, 12], etc. Detailed explanations and discussions on these MU methods are provided in Appendix A. While enabling good performance, we note their two critical drawbacks far underexplored:

• **Frequent revisits to the massive training data.** Existing MU methods generally request to visit training samples in both $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ during the iterative parameter updates to the pretrained model. Such massive data fetching not only incurs expensive computational overhead, but also poses privacy threats during the frequent access to (user) data.

• **Iterative modifications to the pretrained model.** Whenever a particular dataset $\mathcal{D}_{\text{fg}}$ is requested to be forgotten, parameters of the pretrained model have to be updated, yielding a new model tailored to $\mathcal{D}_{\text{fg}}$. This can be computationally infeasible in practice, as each unlearning request demands a new model modified from the pretrained one.

In this work, we aim to address the above challenges in approximate MU methods. *(i)* Firstly, a new perspective to study MU in *low-dimensional feature subspaces* is introduced. We reveal the potential separability in feature subspaces between $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ on the retrained model when comparing to the pretrained model. *(ii)* Then, the observed separability inspires us to *learn a feature subspace* from the pretrained model to achieve unlearning. To this end, the feature subspace is optimized to differentiate $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ by capturing maximal information of the features w.r.t. $\mathcal{D}_{\text{rm}}$ and meanwhile describing minimal information w.r.t. $\mathcal{D}_{\text{fg}}$, thus fulfilling the objective of unlearning. This optimization is implemented following the spirits of Principal Component Analysis (PCA) [13]: A projection matrix is trained based on only two feature covariance matrices of $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ and then is inserted into the pretrained model for inference. Our unlearning method is thereby named **LOFT** for learning **LO**w-dimensional **F**ea**T**ure subspaces.

Figure 1 provides an overview of our LOFT and mainstream approximate MU methods. LOFT offers efficiency for computation and storage, flexibility in practice, and reduced raw-data exposure during unlearning, towards closing the gaps in adapting to real-world MU scenarios:

• **One-shot feature fetching.** LOFT conducts MU in the feature space by fetching the features from the pretrained model *once*. After this one-shot fetching, two covariance matrices of $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ are computed and remain fixed, and our PCA-based optimization then updates only the projection directions to learn a subspace that distinguishes $\mathcal{D}_{\text{rm}}$ from $\mathcal{D}_{\text{fg}}$. Unlike other approximate MU methods, LOFT avoids direct, repeated, iterative raw training data access during unlearning, improving computational efficiency.

• **Plug-in module implementation.** LOFT optimizes a projection matrix to eliminate information of $\mathcal{D}_{\text{fg}}$ and preserve that of $\mathcal{D}_{\text{rm}}$, thereby enabling a plug-and-play implementation: this projection matrix is inserted into the pretrained model, without retraining or fine-tuning the pretrained parameters. This merit is significant for efficiency and real-world practicality. For instance, when handling multiple unlearning requests from different users, LOFT only requires a separate projection module in a rather small size for each request, instead of a new model modified from the pretrained DNN.

The key contributions of this work are summarized below:

- We present a novel feature-subspace perspective for approximate MU and, to the best of our knowledge, are the first to show that forgetting and remaining data can exhibit separability in low-dimensional feature subspaces.
- A new MU method named LOFT is proposed by learning a feature subspace from pretrained features via PCA-based techniques to capture information from the remaining data while suppressing information from the forgetting data.
- LOFT performs *one-shot feature fetching* and *optimizes only a projection matrix* using two fixed feature covariance matrices. This plug-in design avoids repeated raw-data access and full-model updates, reducing optimized parameters, computation, and per-request storage.
- Extensive experiments verify our superior unlearning accuracy and efficiency on varied datasets, networks, unlearning tasks, and real-world applications.

2 Preliminaries

Machine unlearning. MU aims at maintaining prediction performance of a well-trained model $f_{\rm pre}$ when removing (or forgetting) specific data [4]. In MU, the model $f_{\rm pre}$ pretrained on the full dataset $\mathcal{D}$ is given. A particular subset $\mathcal{D}_{\rm fg}$ is requested to be forgotten, and we have the remaining data denoted as $\mathcal{D}_{\rm rm} := \mathcal{D} \backslash \mathcal{D}_{\rm fg}$. MU seeks to erase the knowledge of $\mathcal{D}_{\rm fg}$ from $f_{\rm pre}$ and meanwhile to preserve the performance on $\mathcal{D}_{\rm rm}$. The golden standard in MU is to attain a model $f_{\rm exact}$ retrained from scratch on $\mathcal{D}_{\rm rm}$, i.e., the exact MU [5]. In contrast, fine-tuning the pretrained model $f_{\rm pre}$ to approximate the ideal performance of $f_{\rm exact}$ has been a central research focus for flexibility and efficiency, namely the approximate MU [6]. We provide more discussions on related work in Appendix A.

Principal component analysis. PCA has long been a fundamental tool in machine learning and beyond [13]. PCA seeks orthogonal directions, i.e., principal components capturing the highest projection variance, so as to keep maximal data information. These directions can be obtained by taking the eigenvectors of the input covariance matrix. PCA can explore the intrinsic patterns residing in low-dimension subspaces spanned by those principal components, and has been studied for learning with modern DNNs, e.g., training algorithms [14] and anomaly detection applications [15, 16]. In this work, we follow the spirits of PCA for MU by *learning* the projection directions associated with the low-dimensional feature subspace from $f_{\rm pre}$.

Notations. Let $f_{\rm pre} : \mathcal{X} \to \mathbb{R}^C$ be a DNN pretrained on $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$, with samples $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^D$ and their labels $y_i \in \mathcal{Y} = \{1, 2, \cdots, C\}$. The model $f(\cdot)$ takes $\boldsymbol{x}$ as the input and outputs C-dimensional logits $f(\boldsymbol{x}) \in \mathbb{R}^C$ for classification prediction. More specifically, f can be structured with a backbone $g : \mathcal{X} \to \mathbb{R}^d$ and a linear layer $h : \mathbb{R}^d \to \mathbb{R}^C$ in sequence, where $g(\cdot)$ learns the (penultimate-layer) feature $\boldsymbol{z} = g(\boldsymbol{x}) \in \mathbb{R}^d$ and then the linear layer $h(\cdot)$ is applied to the feature $\boldsymbol{z}$ for the output logits, i.e., $f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$.

3 A Feature Subspace Perspective for Approximate MU

In this section, we present an empirical study that motivates our feature subspace perspective for approximate MU. Particularly, we study whether $\mathcal{D}_{\rm fg}$ and $\mathcal{D}_{\rm rm}$ can be separated in low-dimensional feature subspaces under $f_{\rm pre}$ and $f_{\rm exact}$. Here, "separability" does not refer to linear classification separability, but to a reconstruction property in low-dimensional feature subspaces, e.g., a subspace fitted to $\mathcal{D}_{\rm rm}$ reconstructs $\mathcal{D}_{\rm rm}$ significantly better than $\mathcal{D}_{\rm fg}$.

The pretrained $f_{\rm pre}$ is learned with the entire training dataset $\mathcal{D} = \mathcal{D}_{\rm fg} \bigcup \mathcal{D}_{\rm rm}$, whereas the ideal model $f_{\rm exact}$ from the exact MU is trained exclusively on $\mathcal{D}_{\rm rm}$ and never sees $\mathcal{D}_{\rm fg}$. This fundamental distinction in their training paradigms naturally raises the following question:
After exact retraining, do the features of $\mathcal{D}_{\rm fg}$ and $\mathcal{D}_{\rm rm}$ exhibit different low-dimensional structures, and if so, can this low-dimensional property inspire approximate MU from the pretrained model?

Setups. In our analysis, we consider the penultimate-layer features $\boldsymbol{z} \in \mathbb{R}^d$ from the network backbone. Let $\mathbf{Z}_{\rm fg}^{\rm pre}, \mathbf{Z}_{\rm rm}^{\rm pre}, \mathbf{Z}_{\rm fg}^{\rm exact}, \mathbf{Z}_{\rm rm}^{\rm exact}$ denote the features of $\mathcal{D}_{\rm fg}$ and $\mathcal{D}_{\rm rm}$ from the pretrained $g_{\rm pre}$ and the retrained $g_{\rm exact}$, respectively. Their corresponding covariance matrices are denoted by

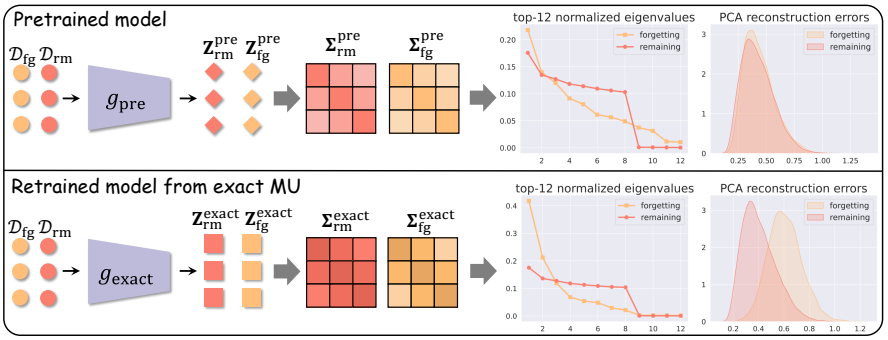


Figure 2: Spectrum and reconstruction analyses on features of $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ w.r.t. the pretrained f_{pre} and the retrained f_{exact}, with results on the top-12 normalized eigenvalues and reconstruction errors. Detailed setups and more empirical results for this analysis can be found in Appendix B.

$\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}, \mathbf{\Sigma}_{\text{rm}}^{\text{pre}}, \mathbf{\Sigma}_{\text{fg}}^{\text{exact}}, \mathbf{\Sigma}_{\text{rm}}^{\text{exact}} \in \mathbb{R}^{d\times d}$. With eigen-decomposition to those covariance matrices, we investigate their eigenvalues and reconstruction performance, demonstrated in Figure 2.

Spectrum analysis. Figure 2 visualizes the eigen-decay, i.e., the distribution of variance across principal components, for the aforementioned covariance matrices. In the pretrained f_{pre}, the eigenvalue decays of $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$ exhibit similar trends. In contrast, a marked divergence appears in the retrained f_{exact}: $\mathbf{\Sigma}_{\text{fg}}^{\text{exact}}$ shows a distinctly sharper eigenvalue decay than $\mathbf{\Sigma}_{\text{rm}}^{\text{exact}}$. Intuitively, since f_{exact} is trained exclusively on $\mathcal{D}_{\text{rm}}$, its feature space is optimized solely to capture the variations of $\mathcal{D}_{\text{rm}}$, lacking dedicated directions to encode the unique characteristics of the unseen $\mathcal{D}_{\text{fg}}$. Consequently, features of unseen $\mathcal{D}_{\text{fg}}$ tend to collapse into fewer dominant directions, leading to the sharper decay, while $\mathcal{D}_{\text{rm}}$ retains a more dispersed structure.

Reconstruction analysis. The PCA reconstruction error quantitatively measures the captured information along projection directions. We apply PCA to the remaining features $\mathbf{Z}_{\text{rm}}$ and obtain $\mathbf{U}_{\text{rm}} \in \mathbb{R}^{d\times s}$ projecting features onto the s-dimensional subspace. The reconstruction error for a centered feature $\bar{\boldsymbol{z}}$ is calculated as $e = \|\mathbf{U}_{\text{rm}}\mathbf{U}_{\text{rm}}^{\top}\bar{\boldsymbol{z}} - \bar{\boldsymbol{z}}\|_2$. In Figure 2, we compute such errors on both $\mathbf{Z}_{\text{rm}}$ and $\mathbf{Z}_{\text{fg}}$ under f_{pre} and f_{exact}, respectively. The reconstruction errors for $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ and $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ of the pretrained f_{pre} are very similar, while those for $\mathbf{Z}_{\text{fg}}^{\text{exact}}$ and $\mathbf{Z}_{\text{rm}}^{\text{exact}}$ of the retrained f_{exact} appear distinctively different. We provide more empirical results for this analysis in Appendix B.

Discussion. This analysis suggests an answer to the question raised earlier: under exact retraining, a feature subspace aligned with $\mathcal{D}_{\text{rm}}$ captures its dominant structure well but describes $\mathcal{D}_{\text{fg}}$ far less effectively, whereas such separability is considerably weaker in the pretrained f_{pre}. Inspired by this observation, we seek to *learn* a feature subspace from f_{pre}. Note that our goal is not to recover the ideal subspace from f_{exact}, but rather to execute subspace learning based on f_{pre} toward a similar objective: a feature subspace that preserves dominant variation of $\mathcal{D}_{\text{rm}}$, while suppressing information associated with $\mathcal{D}_{\text{fg}}$, thereby achieving effective unlearning on f_{pre}.

Remark. We emphasize that our analysis here is intended primarily for the class-wise forgetting. In this setting, $\mathcal{D}_{\text{fg}}$ includes samples from one or multiple classes, and is thereby semantically more coherent, making its feature structure more likely to deviate from that of $\mathcal{D}_{\text{rm}}$ after exact retraining. In contrast, under the more challenging instance-wise forgetting, $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ may substantially overlap in semantics and class membership, leading to weaker subspace separability, as demonstrated in Appendix B. Accordingly, the purpose of this analysis is not to claim universal separability across all unlearning scenarios, but to identify a practically useful feature subspace phenomenon that motivates our method. Moreover, our principle of subspace learning is also potentially beneficial for instance-wise unlearning, where the desired separability is weaker and must be induced rather than expected to emerge directly, as validated in our experiments later in Section 5.

4 Approximate MU with Subspace Learning

Grounded on the feature subspace perspective and the analysis in Section 3, we now focus on how to attain a preferable feature subspace for f_{pre}, such that the features $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ and $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ can be well distinguished in this subspace, i.e., diminishing the knowledge of $\mathcal{D}_{\text{fg}}$ and preserving that of $\mathcal{D}_{\text{rm}}$ from f_{pre}. Accordingly, we name this proposed method as **LOFT**, learning **LO**w-dimensional **F**ea**T**ure subspaces. Apart from approaching the outputs or parameters of f_{exact}, LOFT leads to an interesting and promising direction of formulating unlearning models that pertain similar properties in feature (sub)spaces, which we hope could bring new perspectives and opportunities in MU.

4.1 Optimization

LOFT aims to find an s-dimensional subspace $\texttt{range}(\mathbf{U})$ for the d-dimensional features of the pretrained model f_{pre}. This linear subspace is given by the span of the orthonormal columns of the $d \times s$ matrix $\mathbf{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_s]$, the set of which defines the well-known Stiefel manifold $\text{St}(d, s)$ with $d \geq s$. Towards the feature separability for MU, LOFT follows the spirits of PCA to seek such a subspace (to optimize $\mathbf{U}$), in which the information of $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ is maximally captured and meanwhile that of $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ gets minimally described. We start from Proposition 4.1 [17] below.

Proposition 4.1 ([17]). *Let $\mathbf{M}$ be an $l \times l$ symmetric matrix. Let $\gamma_1, \ldots, \gamma_m$ be its m smallest eigenvalues, possibly including multiplicities, with associated orthonormal eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. Let $\mathbf{V}$ be a matrix whose columns are these eigenvectors. Then, the optimization problem $\min_{\mathbf{U} \in \text{St}(l,m)} \text{Tr}(\mathbf{U}^\top \mathbf{M} \mathbf{U})$ has a minimizer at $\mathbf{U}_\star = \mathbf{V}$ and we have $\mathbf{U}_\star^\top \mathbf{M} \mathbf{U}_\star = \text{diag}(\boldsymbol{\Gamma})$ with $\boldsymbol{\Gamma} = [\gamma_1, \ldots, \gamma_m]^\top$, where $\text{Tr}(\cdot)$ denotes the matrix trace.*

Proposition 4.1 [17] formalizes the optimization problem for obtaining the projection directions capturing minimal information in PCA with the Stiefel manifold, where $\mathbf{M}$ denotes the covariance matrix. This correspondingly leads to the optimization problem for the projection directions capturing maximal information: the subspace spanned by the eigenvectors of $\mathbf{M}$ with the m largest eigenvalues is obtained by solving $\min_{\mathbf{U} \in \text{St}(l,m)} \text{Tr}(\mathbf{M} - \mathbf{U}\mathbf{U}^\top \mathbf{M} \mathbf{U}\mathbf{U}^\top)$.

While Proposition 4.1 [17] identifies the optimal subspace for a single criterion (either minimizing or maximizing variance), it motivates the optimization objective of LOFT to address the dual requirements in MU of suppressing $\mathcal{D}_{\text{fg}}$ (J_{fg}) and preserving $\mathcal{D}_{\text{rm}}$ (J_{rm}) as follows.

$$\min_{\mathbf{U} \in \text{St}(d,s)} J(\mathbf{U}) = \underbrace{\Big(\frac{\text{Tr}(\mathbf{U}^\top \boldsymbol{\Sigma}_{\text{fg}}^{\text{pre}} \mathbf{U})}{\text{Tr}(\boldsymbol{\Sigma}_{\text{fg}}^{\text{pre}})} \Big)^2}_{J_{\text{fg}}} + \underbrace{\Big(\frac{\text{Tr}\left(\boldsymbol{\Sigma}_{\text{rm}}^{\text{pre}} - \mathbf{U}\mathbf{U}^\top \boldsymbol{\Sigma}_{\text{rm}}^{\text{pre}} \mathbf{U}\mathbf{U}^\top\right)}{\text{Tr}\left(\boldsymbol{\Sigma}_{\text{rm}}^{\text{pre}}\right)} \Big)^2}_{J_{\text{rm}}}, \tag{1}$$

where numerators play key roles and denominators balance normalization. The joint optimization of J_{rm} and J_{fg} advocates the separability between $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ and $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ in the subspace projected via $\mathbf{U}$:

- J_{fg} quantifies the projection variance of the forgetting features $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ onto the subspace spanned by $\mathbf{U}$, as explained in Proposition 4.1 [17]. Minimizing J_{fg} encourages $\mathbf{U}$ to capture minimal information of $\mathbf{Z}_{\text{fg}}^{\text{pre}}$, aiming to forget the knowledge of $\mathcal{D}_{\text{fg}}$.
- J_{rm} measures the projection variance of the remaining features $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ captured within the complement subspace w.r.t. $\mathbf{U}$, namely the reconstruction error. Minimizing J_{rm} forces $\mathbf{U}$ to preserve maximal information of $\mathbf{Z}_{\text{rm}}^{\text{pre}}$, so as to maintain model performance on $\mathcal{D}_{\text{rm}}$.

The Riemannian Adam optimizer [18, 19] is deployed to solve (1) with $\mathbf{U}$ optimized on the Stiefel manifold $\text{St}(d, s)$. We show that $J(\mathbf{U})$ is L-smooth, and thereby the convergence rate of Riemannian Adam on minimizing $J(\mathbf{U})$ is guaranteed in Lemma C.3 of Appendix C.1.

4.2 Implementation and Discussion

LOFT simply requires to optimize a projection matrix $\mathbf{U}$ applied to features $\mathbf{Z}_{\text{fg}}^{\text{pre}}$ and $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ from the pretrained f_{pre}. During inference, the resulting model from LOFT for unlearning is given by $f_{\mathbf{U}}(\cdot) \triangleq h_{\text{pre}}(\mathbf{U}\mathbf{U}^\top g_{\text{pre}}(\cdot))$, in contrast to the pretrained $f_{\text{pre}} = h_{\text{pre}}(g_{\text{pre}}(\cdot))$, as shown in Figure 1. Algorithm 1 provides an intuitive overview of a practical implementation of LOFT.

Algorithm 1 LOFT for Approximate Machine Unlearning

Offline one-shot feature fetching

Require: Pretrained model $f_{\mathrm{pre}}(\cdot) = h_{\mathrm{pre}}(g_{\mathrm{pre}}(\cdot))$ and training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$

Ensure: Indexed training feature set $\mathcal{Z}$

Set f_{pre} to evaluation mode and freeze all its parameters

Extract $\boldsymbol{z}_i \leftarrow g_{\mathrm{pre}}(\boldsymbol{x}_i)$ for every $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$ through one inference pass

Store the indexed training feature set $\mathcal{Z} \leftarrow \{(i, \boldsymbol{z}_i)\}_{i=1}^{N}$

On-demand unlearning

Require: Pretrained model f_{pre}, training feature set $\mathcal{Z}$, forgetting request $\mathcal{F} \subseteq \{1, \ldots, N\}$, subspace dimension s, and optimization steps T

Ensure: Request-specific projection $\mathbf{U}$ and unlearned model $f_{\mathbf{U}}$

Retrieve forgetting features $\mathbf{Z}_{\mathrm{fg}}^{\mathrm{pre}} \leftarrow \{\boldsymbol{z}_i \mid i \in \mathcal{F}\}$ from $\mathcal{Z}$

Retrieve remaining features $\mathbf{Z}_{\mathrm{rm}}^{\mathrm{pre}} \leftarrow \{\boldsymbol{z}_i \mid i \in \{1, \ldots, N\} \setminus \mathcal{F}\}$ from $\mathcal{Z}$

Compute covariances $\mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}} \leftarrow \mathrm{Cov}(\mathbf{Z}_{\mathrm{fg}}^{\mathrm{pre}})$ and $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}} \leftarrow \mathrm{Cov}(\mathbf{Z}_{\mathrm{rm}}^{\mathrm{pre}})$

Initialize $\mathbf{U} \in \mathrm{St}(d, s)$

for $t = 1, \ldots, T$ **do**

  Evaluate $J(\mathbf{U}) = J_{\mathrm{fg}} + J_{\mathrm{rm}}$ using $\mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}}$ and $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}}$ as defined in (1)

  Update $\mathbf{U}$ using Riemannian Adam on $\mathrm{St}(d, s)$ ▷ No iterative raw-data access

end for

return $\mathbf{U}$ and $f_{\mathbf{U}}(\cdot) \triangleq h_{\mathrm{pre}}(\mathbf{U}\mathbf{U}^{\top} g_{\mathrm{pre}}(\cdot))$

One-shot feature fetching. The computation to $J(\mathbf{U})$ relies on the two covariance matrices $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}}$ and $\mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}}$ w.r.t. features from f_{pre}. Once $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}}, \mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}}$ are computed, LOFT proceeds to optimize the $d \times s$ projection matrix $\mathbf{U}$ and thus only conducts one-shot feature fetching. This differs fundamentally from existing approximate MU methods, which must repeatedly access the original training data at each optimization iteration. By eliminating the need for such direct and massive visits to data samples, LOFT significantly reduces unlearning-time computation and raw-data exposure.

Plug-in module implementation. Compared to fine-tuning or retraining all parameters in f_{pre}, LOFT optimizes a $d \times s$ matrix as in (1), the parameter number of which is thereby reduced by orders of magnitude. Thus, LOFT takes significantly less computation overhead than the exact MU and mainstream approximate MU methods, supported by numerical results in Section 5.2. In implementation, LOFT flexibly serves as a plug-in module into f_{pre}, without modifying parameters of f_{pre}. This merit is of great practicality in real-world deployment: in cases of handling multiple unlearning requests from different users, each request can be addressed by efficiently optimizing a small-size projection matrix, instead of maintaining multiple models refined from f_{pre}.

Theoretical analysis. LOFT offers a novel feature subspace perspective and an empirical method for MU. We position LOFT as orthogonal and complementary to the established research on certified unlearning [20]. Certified methods provide rigorous guarantees, often via differential privacy [21] by bounding the parameter difference between approximate and exact unlearned models. In contrast, LOFT operates on a fundamentally different principle. Instead of updating parameters of f_{pre}, LOFT applies a learnable projection matrix $\mathbf{U}$ on features from f_{pre} to achieve unlearning in a feature subspace sense. As a result, existing certification frameworks based on parameter-space proximity are not directly applicable to LOFT.

Nevertheless, we provide a preliminary theoretical discussion for LOFT, as formalized in Proposition 4.2 below. It shows that, under bounded optimization errors and proximity assumptions between f_{pre} and f_{exact}, the output discrepancy between our $f_{\mathbf{U}}$ and f_{exact} can be controlled. Proposition 4.2 should therefore be interpreted only as a conditional discrepancy analysis that offers intuition for the approximation behavior of LOFT, rather than a certified unlearning guarantee. Due to space limitation, a detailed version of Proposition 4.2 is put in Appendix C.2.

Proposition 4.2. *Suppose the learned projection* $\hat{\mathbf{U}}$ *satisfies* $\mathrm{Tr}(\hat{\mathbf{U}}^{\top} \mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}} \hat{\mathbf{U}}) \leq \epsilon_{\mathrm{fg}}$ *and* $\mathrm{Tr}((\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^{\top}) \mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}} (\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^{\top})) \leq \epsilon_{\mathrm{rm}}$. *Then, under explicit boundedness and proximity assumptions*

stated in Appendix C.2, the expected output discrepancy between $f_{\mathbf{U}}$ and f_{exact}, i.e., $\mathbb{E}_{\boldsymbol{x}}\|f_{\hat{\mathbf{U}}}(\boldsymbol{x}) - f_{\text{exact}}(\boldsymbol{x})\|_2$, is bounded by terms that scale with ϵ_{rm} for $\boldsymbol{x} \in \mathcal{D}_{\text{rm}}$ and with ϵ_{fg} for $\boldsymbol{x} \in \mathcal{D}_{\text{fg}}$.

5 Experiments

In this section, extensive experiments are conducted to evaluate LOFT, showing its potentials of feature subspace learning for MU. In experiments, we primarily focus on the *class-centric* unlearning [12], widely studied in real-world scenarios, as data from different users can represent varied classes. Another significant *instance-wise* unlearning is also investigated. More empirical results, details and discussions are provided in Appendices D-I, including detailed efficiency analysis, more unlearning scenarios, real-world applications and generation tasks.

5.1 Setups

Datasets and models. Experiments are executed on Tiny-ImageNet [22] (200 classes, $64 \times 64 \times 3$) and ImageNet-1K [23] (1,000 classes, $224 \times 224 \times 3$). We skip comparisons on other easier datasets with smaller scales and distinctively fewer classes, e.g., CIFAR [24] and SVHN [25], which have been extensively evaluated and yet show almost incremental improvements with limited challenges for class-centric unlearning. Swin-T [26] and ResNet50 [27] are evaluated on Tiny-ImageNet and ImageNet-1K, respectively. Main comparison experiments are repeated 3 times, with mean and standard deviations reported.

Baselines and restricted data access. We compare LOFT with a series of strong baselines: Gradient Ascent (GA) [9], Random Labeling (RL) [8], SalUn [10], Learn to UnLearn (L2UL) [28], COUN [29] and DELETE [12]. Results of the pretrained f_{pre} and the retrained f_{exact} are also included. See Appendices A and E for baseline outlines and implementation details.

We consider a practically relevant MU protocol where repeated access to raw training samples is *restricted* during unlearning, as iteratively revisiting raw $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ can be costly and undesirable from a privacy perspective. Existing methods are typically evaluated either with full access to both datasets or in a more restrictive setting using only raw $\mathcal{D}_{\text{fg}}$ [28, 12]. In contrast, LOFT does not revisit raw $\mathcal{D}_{\text{fg}}$ or $\mathcal{D}_{\text{rm}}$ during unlearning, but uses their feature covariances, $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$, precomputed from a one-shot forward pass of f_{pre}. Thus, LOFT still leverages information from $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$, but under a summarized-data access protocol rather than raw-data iterative optimization. Accordingly, we report restricted-access baselines using only raw $\mathcal{D}_{\text{fg}}$. For reference, we include baselines with full access to both $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ (marked in gray), which are not directly matched to LOFT in access modality but help contextualize the gap between restricted raw-data access and full-data optimization.

Metrics. Four metrics on learning accuracy are leveraged [12]: $\text{Acc}_{\text{fg}}^{\text{tr}}$ on the forgetting $\mathcal{D}_{\text{fg}}$ and $\text{Acc}_{\text{rm}}^{\text{tr}}$ on the remaining $\mathcal{D}_{\text{rm}}$ of the training data, $\text{Acc}_{\text{fg}}^{\text{te}}$ on the forgetting $\mathcal{D}_{\text{fg}}^{\text{te}}$ and $\text{Acc}_{\text{rm}}^{\text{te}}$ on the remaining $\mathcal{D}_{\text{rm}}^{\text{te}}$ of the test data. In addition, the Membership Inference Attack (MIA) success rate [30] is adopted, measuring the proportion of samples in $\mathcal{D}_{\text{fg}}$ that gets memorized by the unlearned model. Results against more MIAs, LiRA[31] and RMAI [32], are provided in Appendix H.

Note that the performance gap to the retrained f_{exact} essentially reflects the unlearning efficacy of approximate MU methods. This is the key evaluation criterion for MU and is marked in blue within parentheses. Hence, the overall performances are indicated by the average of the accuracy and MIA gap to f_{exact} (Avg.G.). We mark the best results in bold font.

5.2 Main Comparisons

Multi/Single-class unlearning. Tables 1 and 2 present comparisons on the unlearning results and computational costs with Swin-T on Tiny-ImageNet, where 4 of 200 classes (2%) are randomly selected as $\mathcal{D}_{\text{fg}}$. Table 1 shows that unlearning without access to $\mathcal{D}_{\text{rm}}$ leads to substantial performance drops for baselines such as RL and SalUn, showing their strong reliance to $\mathcal{D}_{\text{rm}}$. Our LOFT leverages the two feature covariance matrices and achieves highly effective unlearning performance with a smallest average gap of 0.85 to the retrained model. On efficiency, Table 2 shows that the parameter number and the Run Time Efficiency (RTE) in our LOFT are less by orders of magnitude than that of other baselines, as LOFT simply optimizes a projection matrix operating on two covariance matrices

Table 1: Comparison results of Swin-T on Tiny-ImageNet. The ✓ and ✗ indicate whether $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ are involved in optimization. Avg.G. is average accuracy and MIA gap to the retrained model.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\textbf{Acc}^{\text{tr}}_{\text{rm}}$	$\textbf{Acc}^{\text{tr}}_{\text{fg}}$	$\textbf{Acc}^{\text{te}}_{\text{rm}}$	$\textbf{Acc}^{\text{te}}_{\text{fg}}$	MIA	Avg.G.↓
f_{pre}	✓	✓	$99.63_{\pm 0.01}$	$99.75_{\pm 0.20}$	$74.54_{\pm 0.16}$	$74.83_{\pm 8.14}$	$96.63_{\pm 1.02}$	-
f_{exact}	✓	✗	$99.67_{\pm 0.03}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$75.44_{\pm 0.19}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	0.00
GA	✗	✓	$88.40_{\pm 5.51}$ (11.27)	$23.12_{\pm 12.14}$ (23.12)	$65.27_{\pm 3.56}$ (10.17)	$18.82_{\pm 7.52}$ (18.82)	$19.95_{\pm 10.24}$ (19.95)	16.67
GA	✓	✓	$99.49_{\pm 0.03}$ (0.18)	$0.02_{\pm 0.03}$ (0.02)	$74.57_{\pm 0.21}$ (0.87)	$0.00_{\pm 0.00}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	0.21
RL	✗	✓	$92.94_{\pm 1.91}$ (6.73)	$9.98_{\pm 7.44}$ (9.98)	$67.17_{\pm 1.46}$ (8.17)	$7.50_{\pm 7.57}$ (7.50)	$1.83_{\pm 0.54}$ (1.83)	6.85
RL	✓	✓	$99.91_{\pm 0.01}$ (0.24)	$1.52_{\pm 1.17}$ (1.52)	$74.43_{\pm 0.18}$ (1.01)	$0.33_{\pm 0.58}$ (0.33)	$0.00_{\pm 0.00}$ (0.00)	0.62
SalUn	✗	✓	$93.16_{\pm 1.90}$ (6.51)	$13.55_{\pm 10.30}$ (13.55)	$67.49_{\pm 1.56}$ (7.95)	$10.00_{\pm 9.26}$ (10.00)	$3.45_{\pm 0.87}$ (3.45)	8.29
SalUn	✓	✓	$99.88_{\pm 0.01}$ (0.21)	$2.57_{\pm 1.09}$ (2.57)	$74.71_{\pm 0.19}$ (0.73)	$1.00_{\pm 0.87}$ (1.00)	$0.00_{\pm 0.00}$ (0.00)	0.90
COUN+RL	✗	✓	$92.56_{\pm 0.75}$ (7.11)	$10.07_{\pm 3.85}$ (10.07)	$67.85_{\pm 0.51}$ (7.59)	$5.67_{\pm 2.57}$ (5.67)	$2.28_{\pm 0.46}$ (2.28)	6.54
COUN+RL	✓	✓	$99.94_{\pm 0.01}$ (0.27)	$0.73_{\pm 0.16}$ (0.73)	$72.89_{\pm 0.18}$ (1.36)	$0.67_{\pm 0.58}$ (0.67)	$0.00_{\pm 0.00}$ (0.00)	0.61
L2UL	✗	✓	$92.69_{\pm 3.43}$ (6.98)	$4.43_{\pm 0.15}$ (4.43)	$67.69_{\pm 2.58}$ (7.75)	$2.00_{\pm 0.87}$ (2.00)	$2.57_{\pm 0.92}$ (2.57)	4.75
DELETE	✗	✓	$99.13_{\pm 0.12}$ (**0.54**)	$13.32_{\pm 4.17}$ (13.32)	$73.31_{\pm 0.28}$ (2.13)	$6.50_{\pm 2.18}$ (6.50)	$1.85_{\pm 0.48}$ (1.85)	4.88
LOFT	only $\boldsymbol{\Sigma}^{\text{pre}}_{\text{rm}}$	only $\boldsymbol{\Sigma}^{\text{pre}}_{\text{fg}}$	$98.59_{\pm 0.07}$ (1.08)	$1.48_{\pm 1.27}$ (**1.48**)	$74.27_{\pm 0.18}$ (**1.17**)	$0.50_{\pm 0.50}$ (**0.50**)	$0.00_{\pm 0.00}$ (**0.00**)	**0.85**

Table 2: Comparison on computational costs w.r.t. the results in Table 1. The detailed counts and percentages of optimized parameters are listed. RTE is the run time efficiency in seconds on single NVIDIA GeForce RTX 4090 GPU. For LOFT, RTE measures only the optimization time of (1), as training features can be pre-calculated in practice and covariances are readily available for on-demand use. Refer to Table S1 in Appendix D for a detailed analysis on the full pipeline of LOFT.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	# optimized param. (%)	RTE (seconds) ↓	additional requirements
f_{pre}	✓	✓	27,673,154 (100%)	882.05	-
f_{exact}	✓	✗	27,673,154 (100%)	858.14	-
GA	✗	✓	27,673,154 (100%)	13.49	-
GA	✓	✓	27,673,154 (100%)	218.37	-
RL	✗	✓	27,673,154 (100%)	13.19	-
RL	✓	✓	27,673,154 (100%)	438.48	-
SalUn	✗	✓	13,836,577 (50%)	15.00	a parameter saliency mask
SalUn	✓	✓	13,836,577 (50%)	450.23	a parameter saliency mask
COUN+RL	✗	✓	27,673,154 (100%)	23.45	-
COUN+RL	✓	✓	27,673,154 (100%)	704.87	-
L2UL	✗	✓	27,673,154 (100%)	15225.05	a set of adversarial examples; a parameter importance mask
DELETE	✗	✓	27,673,154 (100%)	22.14	a copy of the pretrained model
LOFT	only $\boldsymbol{\Sigma}^{\text{pre}}_{\text{rm}}$	only $\boldsymbol{\Sigma}^{\text{pre}}_{\text{fg}}$	**192,000 (0.69%)**	**0.24**	a projection matrix $\mathbf{U} \in \text{St}(768, 250)$

and others generally update the pretrained model with the original training data. A detailed discussion on the time and memory cost through the entire pipeline of LOFT is presented in Appendix D, where LOFT maintains its efficiency advantage.

For single-class unlearning, we consider the much more challenging ImageNet-1K dataset with ResNet50, where 1 of 1,000 classes is randomly selected as $\mathcal{D}_{\text{fg}}$ with 3 repeated runs. Given the extremely large scale of this dataset ($\mathcal{D}_{\text{rm}}$ contains over 1.28M samples), we therefore compare with methods that only utilize $\mathcal{D}_{\text{fg}}$. The results in Table 3 demonstrates that LOFT outperforms other baselines with the smallest average gap, together with the shortest running time (less than 1 second) and the fewest number of parameters (reduced by approximately 96%).

Instance unlearning. LOFT is also applicable to instance-wise unlearning with results shown in Table 4, where 1,000 of 100,000 training samples (1%) in Tiny-ImageNet are randomly selected as $\mathcal{D}_{\text{fg}}$. Accuracy on the whole test dataset is evaluated as Acc^{te}. Under this random forgetting setting, $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ contain samples from all classes, which is more challenging to seek a feature subspace well differentiating $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$. Nevertheless, in Table 4, the low MIA gap of LOFT implies that its learned subspace can effectively remove information related to $\mathcal{D}_{\text{fg}}$. Furthermore, LOFT attains superior performance and efficiency on the overall Avg.G. metric across all baselines by optimizing only <1% parameters w.r.t. the given model in 0.5 seconds.

More unlearning setups, applications and generation tasks. Due to space limitation, we discuss two practical unlearning scenarios in Appendix F: *extreme unlearning*, which forgets a very high ratio of the training samples, and *continual unlearning*, which handles multiple unlearning requests in sequence. We further consider two real-world applications in Appendix G: *face recognition*, where the model is required to forget a particular identity, and *emotion recognition*, where it is required to forget a particular emotion category. Our LOFT maintains superior unlearning performance

Table 3: Comparison results under single-class unlearning with ResNet50 on ImageNet-1K.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{te}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{te}}$	Avg.G.↓	RTE ↓	# param. (%)
f_{pre}	✓	✓	$77.92_{\pm 0.01}$	$83.46_{\pm 7.64}$	$76.15_{\pm 0.01}$	$81.33_{\pm 10.26}$	-	-	25,557,032 (100%)
f_{exact}	✓	✘	$80.25_{\pm 0.16}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$75.72_{\pm 0.04}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	0.00	-	25,557,032 (100%)
GA	✘	✓	$71.87_{\pm 1.80}$ (8.38)	$0.00_{\pm 0.00}$ (0.00)	$71.42_{\pm 1.89}$ (4.30)	$0.00_{\pm 0.00}$ (0.00)	3.17	9.32	25,557,032 (100%)
RL	✘	✓	$72.11_{\pm 1.42}$ (8.14)	$1.87_{\pm 0.69}$ (1.87)	$71.53_{\pm 1.44}$ (4.19)	$5.33_{\pm 4.16}$ (5.33)	4.88	14.40	25,557,032 (100%)
SalUn	✘	✓	$71.56_{\pm 0.68}$ (8.69)	$2.67_{\pm 0.77}$ (2.67)	$70.84_{\pm 0.60}$ (4.88)	$4.67_{\pm 3.06}$ (4.67)	5.23	27.73	12,778,516 (50%)
COUN+RL	✘	✓	$70.04_{\pm 2.01}$ (10.21)	$1.36_{\pm 0.35}$ (1.36)	$69.72_{\pm 1.82}$ (6.00)	$4.67_{\pm 3.06}$ (4.67)	5.56	15.24	25,557,032 (100%)
DELETE	✘	✓	$72.46_{\pm 1.30}$ (7.79)	$0.95_{\pm 0.78}$ (0.95)	$71.90_{\pm 1.23}$ (**3.82**)	$2.00_{\pm 3.46}$ (2.00)	3.64	17.82	25,557,032 (100%)
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	$79.96_{\pm 0.38}$ (**0.29**)	$0.00_{\pm 0.00}$ (**0.00**)	$69.40_{\pm 0.35}$ (6.32)	$0.00_{\pm 0.00}$ (**0.00**)	**1.65**	**0.94**	**1,024,000 (4.01%)**

Table 4: Comparison results under instance unlearning with Swin-T on Tiny-ImageNet.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{tr}}$	$\mathbf{Acc}^{\mathbf{te}}$	**MIA**	Avg.G.↓	RTE ↓	# param. (%)
f_{pre}	✓	✓	$99.63_{\pm 0.00}$	$99.80_{\pm 0.01}$	$74.55_{\pm 0.00}$	$96.20_{\pm 0.44}$	-	882.05	27,673,154 (100%)
f_{exact}	✓	✘	$99.67_{\pm 0.02}$ (0.00)	$75.97_{\pm 1.00}$ (0.00)	$75.03_{\pm 0.10}$ (0.00)	$64.47_{\pm 1.25}$ (0.00)	0.00	857.21	27,673,154 (100%)
GA	✘	✓	$99.63_{\pm 0.00}$	$99.80_{\pm 0.01}$	$74.55_{\pm 0.00}$	$96.20_{\pm 0.44}$	-	882.05	27,673,154 (100%)
RL	✘	✓	$97.56_{\pm 0.62}$ (2.11)	$95.00_{\pm 1.64}$ (19.03)	$70.57_{\pm 0.50}$ (4.46)	$88.83_{\pm 0.99}$ (24.36)	12.49	7.61	27,673,154 (100%)
SalUn	✘	✓	$98.47_{\pm 0.34}$ (**1.20**)	$97.23_{\pm 1.01}$ (21.26)	$71.86_{\pm 0.24}$ (3.17)	$89.60_{\pm 1.08}$ (25.13)	12.69	12.79	13,836,577 (50%)
L2UL	✘	✓	$94.77_{\pm 0.51}$ (4.90)	$74.90_{\pm 0.10}$ (**1.07**)	$68.69_{\pm 0.27}$ (6.34)	$73.90_{\pm 0.46}$ (9.43)	5.44	7919.11	27,673,154 (100%)
COUN+RL	✘	✓	$98.30_{\pm 0.32}$ (1.37)	$96.47_{\pm 0.21}$ (20.50)	$71.45_{\pm 0.26}$ (3.58)	$89.02_{\pm 0.82}$ (24.55)	12.50	11.45	27,673,154 (100%)
DELETE	✘	✓	$91.84_{\pm 2.12}$ (7.83)	$77.57_{\pm 2.39}$ (1.60)	$65.44_{\pm 1.43}$ (9.59)	$68.67_{\pm 1.50}$ (4.20)	5.81	8.42	27,673,154 (100%)
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	$92.65_{\pm 0.54}$ (7.02)	$82.97_{\pm 1.17}$ (7.00)	$68.14_{\pm 0.42}$ (6.89)	$64.47_{\pm 2.36}$ (**0.00**)	**5.23**	**0.50**	**230,400 (0.83%)**

across these unlearning setups and applications. Furthermore, Appendix I provides a preliminary exploration of leveraging the low-dimensional feature subspace for unlearning in generation tasks, a still underexplored area, and our initial results demonstrate the potential of LOFT in this setting while offering beneficial insights for future research. Collectively, these results further demonstrate the broad generalizability of our feature-subspace learning approach for machine unlearning across diverse scenarios, real-world applications, and task paradigms.

5.3 Ablation Study and Sensitivity Analysis

Objective terms J_{rm} and J_{fg}. We explore the roles of the two loss terms in the optimization objective $J(\mathbf{U})$ of (1): J_{rm} preserves performance on $\mathcal{D}_{\text{rm}}$ and J_{fg} guarantees forgetting on $\mathcal{D}_{\text{fg}}$. By respectively removing J_{rm} and J_{fg} from $J(\mathbf{U})$, we report the results in Table 5. It shows that optimizing $\mathbf{U}$ only with J_{fg} suppresses Acc_{fg} and MIA, while failing to preserve Acc_{rm}. Employing only J_{rm} manages to achieve high Acc_{rm}, but leaves $\mathcal{D}_{\text{fg}}$ hardly forgotten with high Acc_{fg} and MIA. Hence, a joint optimization with J_{rm} and J_{fg} is crucial for effective MU. Besides, a sensitivity analysis of a weighted sum for J_{rm} and J_{fg} is provided in Appendix E.4.

Table 5: Ablation study on J_{rm} and J_{fg}.

objective	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{te}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{te}}$	**MIA**
None	97.11	98.10	71.61	65.00	86.10
J_{fg} only	89.51	11.50	65.40	4.00	0.00
J_{rm} only	99.19	87.80	75.07	53.00	58.35
$J_{\text{rm}} + J_{\text{fg}}$	98.66	1.10	74.27	0.00	0.00

The "J_{fg}-only" results in Table 5 show a clear degradation in Acc_{rm}, indicating that the current LOFT does not fully address the strict setting where no information from $\mathcal{D}_{\text{rm}}$ is available. Nevertheless, LOFT substantially relaxes the remaining-data access requirement: a one-time, gradient-free pass can be performed before unlearning to extract and cache features. Once these features are available, LOFT computes the covariances and optimizes $\mathbf{U}$ without revisiting the raw remaining samples, unlike methods that repeatedly access $\mathcal{D}_{\text{rm}}$ during iterative optimization.

Subspace dimensions s. The projection matrix $\mathbf{U}$ is confined on the Stiefel manifold $\text{St}(d, s)$, consisting of s orthonormal bases in columns. Here, the dimension s determines the percentage of captured feature information in the subspace. A sensitivity analysis w.r.t. varied values of s on the unlearning performance is provided in Figure 3 (left). As illustrated, there is a clear trade-off: a larger s preserves higher Acc_{rm} but sacrifices the ability to forget $\mathcal{D}_{\text{fg}}$ and incurs higher computation, whereas a smaller s effectively forgets $\mathcal{D}_{\text{fg}}$ but compromises Acc_{rm}. From our evaluations, we recommend initializing s with preserving 95% of the explained variance in $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$, as this choice readily provides good empirical performance for different settings and tasks evaluated in this work.

Implementation position for subspace learning. The subspace learning in LOFT is primarily applied to the penultimate-layer features $\boldsymbol{z}$, while its implementation can be quite flexible to be inserted into different positions of f_{pre}. As an example, we consider a ResNet18 model that is structured with a preceding module followed by 4 `layer` modules (`layer1`, `layer2`, `layer3`, and

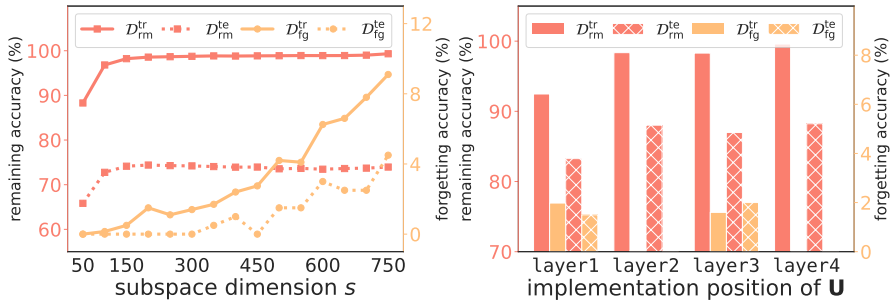


Figure 3: A sensitivity analysis on *(i)* subspace dimensions s (**left**) and *(ii)* different positions of the projection matrix **U** (**right**).

`layer4`) and a linear module[2]. By default, LOFT is applied to the penultimate-layer features, i.e., features after the `layer4` module. In experiments, we further apply LOFT to earlier `layer` modules (`layer1`, `layer2`, and `layer3`) with results shown in Figure 3 (right).

As illustrated, LOFT maintains competitive performance when applied in earlier positions, compared to its default setting of `layer4`. It further indicates that the feature covariance matrices in earlier layers contain sufficient information to achieve effective MU from a low-dimensional feature subspace perspective. However, it can cost more computations, due to the larger feature dimensions and more optimization steps for early-layer implementation. We provide in Table 6 the corresponding feature dimension d, subspace dimension s, and the optimization steps when applying LOFT to different `layer` modules. Considering the trade-off between performance and efficiency, it is recommended to implement LOFT in the penultimate layer, as done in main evaluations.

Table 6: Hyper-parameters in different positions for LOFT.

position	d	s	steps
`layer1`	4,096	500	500
`layer2`	2,048	350	200
`layer3`	1,024	9	200
`layer4`	512	9	50

6 Conclusion and Discussion

This work presents a low-dimensional feature-subspace perspective for MU, a direction that remains largely underexplored yet offers a favorable effectiveness and efficiency trade-off. Our key insight is that $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ can become separable in an appropriate feature subspace, and we aim to learn such a subspace for the pretrained model f_{pre}. Based on this idea, we propose LOFT, which leverages PCA-inspired techniques to learn a projection matrix that preserves the knowledge of $\mathcal{D}_{\text{rm}}$ while suppressing that of $\mathcal{D}_{\text{fg}}$. Its optimization requires only one-shot feature fetching to compute the covariances, avoiding direct and iterative access to raw training data during unlearning. Moreover, LOFT updates only a small projection matrix and can be implemented as a plug-in module to f_{pre} without modifying the full model parameters. Across extensive evaluations, LOFT reduces both the number of updated parameters and the running time by orders of magnitude during unlearning, while maintaining highly competitive unlearning performance.

In addition, since LOFT does not update the parameters of f_{pre}, its extension to generation tasks is less direct; in such cases, the projection matrix needs to be carefully tailored to the specific generative model and the particular unlearning scenario. We provide an initial exploration of this direction in Appendix I, which suggests the potential of feature-subspace learning for MU in generative tasks. We hope our explorations could inspire more sophisticated future study.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No: 62502416, 62372122, 62503051, and U25A20430), and the Research Grants Council (Grant No:15210023 and 15224124), Hong Kong SAR, China, and the Beijing Natural Science Foundation (L2607017).

[2]https://docs.pytorch.org/vision/main/_modules/torchvision/models/resnet.html#resnet18

References

[1] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:25278–25294, 2022.

[2] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. Data-centric artificial intelligence: A survey. *ACM Computing Surveys*, 57(5):1–42, 2025.

[3] Chris Jay Hoofnagle, Bart Van Der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[4] 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 (SP)*, pages 141–159. IEEE, 2021.

[5] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *USENIX Security Symposium*, pages 4007–4022, 2022.

[6] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2008–2016. PMLR, 2021.

[7] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.

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

[9] 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)*, pages 303–319. IEEE, 2022.

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

[11] 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 *AAAI Conference on Artificial Intelligence*, volume 37, pages 7210–7217, 2023.

[12] 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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20350–20359, 2025.

[13] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[14] Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional trajectory hypothesis is true: DNNs can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3411–3420, 2023.

[15] Kun Fang, Qinghua Tao, Kexin Lv, Mingzhen He, Xiaolin Huang, and JIE YANG. Kernel pca for out-of-distribution detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 134317–134344, 2024.

[16] Kun Fang, Qinghua Tao, Mingzhen He, Kexin Lv, Runze Yang, Haibo Hu, Xiaolin Huang, Jie Yang, and Longbing Cao. Kernel pca for out-of-distribution detection: Non-linear kernel selection and approximation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(10):13220–13235, 2026.

[17] Arun Pandey, Michaël Fanuel, Joachim Schreurs, and Johan A. K. Suykens. Disentangled representation learning and generation with manifold optimization. *Neural Computation*, 34(10):2009–2036, 09 2022.

[18] Gary Becigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations (ICLR)*, 2019.

[19] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch. In *Proceedings of the ICML Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

[20] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3832–3842. PMLR, 13–18 Jul 2020.

[21] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[22] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[24] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[25] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[27] 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)*, pages 770–778, 2016.

[28] Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In *AAAI conference on artificial intelligence*, volume 38, pages 11186–11194, 2024.

[29] Yasser H Khalil, Mehdi Setayesh, and Hongliang Li. Coun: Empowering machine unlearning via contrastive learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[30] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[31] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE symposium on security and privacy (SP)*, pages 1897–1914. IEEE, 2022.

[32] Sajjad Zarifzadeh, Philippe Liu, and Reza Shokri. Low-cost high-power membership inference attacks. In *International Conference on Machine Learning*, pages 58244–58282. PMLR, 2024.

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

[34] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[35] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[36] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 18:2345–2354, 2023.

[37] Junyaup Kim and Simon S Woo. Efficient two-stage model retraining for machine unlearning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4361–4369, 2022.

[38] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.

[39] 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 (NeurIPS)*, 36:51584–51605, 2023.

[40] Zhehao Huang, Xinwen Cheng, JingHao Zheng, Haoran Wang, Zhengbao He, Tao Li, and Xiaolin Huang. Unified gradient-based machine unlearning with remain geometry enhancement. *Advances in Neural Information Processing Systems (NeurIPS)*, 37:26377–26414, 2024.

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[42] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[43] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

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

[45] Guanghao Li, Li Shen, Yan Sun, Yue Hu, Han Hu, and Dacheng Tao. Subspace based federated unlearning. *Transactions on Machine Learning Research*, 2026.

[46] Chaohao Fu, Weijia Jia, and Na Ruan. Client-free federated unlearning via training reconstruction with anchor subspace calibration. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 9281–9285. IEEE, 2024.

[47] Tyler Lizzo and Larry Heck. Unlearn efficient removal of knowledge in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7257–7268, 2025.

[48] Marcin Sendera, Łukasz Struski, Kamil Książek, Kryspin Musiol, Jacek Tabor, and Dawid Rymarczyk. Semu: Singular value decomposition for efficient machine unlearning. *International Conference on Machine Learning (ICML)*, 2025.

[49] Yaxin Xiao, Qingqing Ye, Li Hu, Huadi Zheng, Haibo Hu, Zi Liang, Haoyang Li, and Yijie 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.

[50] Junxu Liu, Mingsheng Xue, Jian Lou, Xiaoyu Zhang, Li Xiong, and Zhan Qin. Muter: Machine unlearning on adversarially trained models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4869–4879. IEEE, 2023.

[51] Zitong Li, Qingqing Ye, and Haibo Hu. Funu: Boosting machine unlearning efficiency by filtering unnecessary unlearning. In *ACM on Web Conference*, pages 3366–3376, 2025.

[52] Yang Liu, Mingyuan Fan, Cen Chen, Ximeng Liu, Zhuo Ma, Li Wang, and Jianfeng Ma. Backdoor defense with machine unlearning. In *IEEE INFOCOM*, pages 280–289, 2022.

[53] Zihao Liu, Tianhao Wang, Mengdi Huai, and Chenglin Miao. Backdoor attacks via machine unlearning. In *AAAI Conference on Artificial Intelligence*, volume 38, pages 14115–14123, 2024.

[54] Xiaoyu Xu, Minxin Du, Qingqing Ye, and Haibo Hu. Obliviate: Robust and practical machine unlearning for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3696–3715, 2025.

[55] Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, et al. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, pages 1–14, 2025.

[56] Xiaoyu Xu, Xiang Yue, Yang Liu, Qingqing Ye, Huadi Zheng, Peizhao Hu, Minxin Du, and Haibo Hu. Unlearning isn't deletion: Investigating reversibility of machine unlearning in LLMs. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 142657–142704. PMLR, 06–11 Jul 2026.

[57] Xiaoyu Xu, Minxin Du, Zi Liang, Zhibiao Guo, Zhang Shiyu, Peizhao Hu, Qingqing Ye, Haibo Hu, et al. From domains to instances: Dual-granularity data synthesis for llm unlearning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 28540–28556, 2026.

[58] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[59] Bohan Wang, Yushun Zhang, Huishuai Zhang, Qi Meng, Ruoyu Sun, Zhi-Ming Ma, Tie-Yan Liu, Zhi-Quan Luo, and Wei Chen. Provable adaptivity of adam under non-uniform smoothness. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2960–2969, 2024.

[60] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[61] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer, 2012.

[62] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. VGGFACE2: A dataset for recognising faces across pose and age. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pages 67–74. IEEE, 2018.

[63] Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2584–2593. IEEE, 2017.

[64] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[65] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[66] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023.

[67] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Appendices are organized as follows:

• Appendix A provides an overview of related work in MU and introduces the baseline approximated MU methods included in our empirical comparisons.

• Appendix B presents more empirical evidences for the empirical study in Section 3 to motivate our feature subspace learning.

• Appendix C contains theoretical discussions on the convergence of Problem (1) and the unlearning performance with complete details of Proposition 4.2 in the main text.

• Appendix D provides a comprehensive analysis into the time and memory cost of LOFT throughout its entire pipeline.

• Appendix E presents detailed setups of involved MU methods in experiments of *multi-class*, *single-class* and *instance-wise* unlearning in the main text, together with results on the weighted sum of J_{fg} and J_{rm}.

• Appendix F supplements results under two practical unlearning setups of *extreme unlearning* and *continual unlearning*.

• Appendix G supplements results on two real-world MU applications of *face recognition* and *emotion recognition*.

• Appendix H discusses the privacy scope of LOFT and potential threat models.

• Appendix I gives a preliminary exploration on the low-dimensional feature subspace learning for MU in generation tasks.

A Related Work on Machine Unlearning

Machine unlearning aims at maintaining prediction performance of a well-trained model f_{pre} when removing (forgetting) specific data [33, 34, 4]. In general, such forgetting data $\mathcal{D}_{\mathrm{fg}}$ can mainly be categorized into *class-wise* and *instance-wise* types. The former indicates that $\mathcal{D}_{\mathrm{fg}}$ includes all the samples from one or multiple classes in $\mathcal{Y}$ [35, 36, 12] while the latter implies that $\mathcal{D}_{\mathrm{fg}}$ can be a random subset of the entire training data $\mathcal{D}$ with mixed classes and even all classes [37, 10, 28].

A golden standard of MU is to retrain the model from scratch on the remaining data $\mathcal{D}_{\mathrm{rm}}$ only, known as *the exact MU* [4, 5]. The resulting retrained model f_{exact} never sees the forgetting data $\mathcal{D}_{\mathrm{fg}}$ and thereby is an oracle in evaluating MU performance. However, the model retraining on $\mathcal{D}_{\mathrm{rm}}$ can suffer from heavy computational overload, which motivates a series of researches optimizing the pretrained model f_{pre} by approaching to the ideal performance of the exact retrained model f_{exact}, namely *the approximate MU*. Existing approximate MU methods generally take different techniques to update f_{pre}, usually requiring visiting the forgetting data $\mathcal{D}_{\mathrm{fg}}$ and/or the remaining data $\mathcal{D}_{\mathrm{rm}}$ from the original training dataset. In the following, the involved MU methods in our comparisons are outlined, covering mainstream unlearning techniques and marking their reliance to $\mathcal{D}_{\mathrm{fg}}$ and $\mathcal{D}_{\mathrm{rm}}$.

GA [38, 9] fine-tunes f_{pre} by applying gradient ascent only on the forgetting data $\mathcal{D}_{\mathrm{fg}}$, implemented by maximizing the cross entropy loss. GA with $\mathcal{D}_{\mathrm{fg}}$ only is highly sensitive to the learning rate and usually leads to a substantial drop on the performance of $\mathcal{D}_{\mathrm{rm}}$. In our experiments, GA is enhanced with an additional step of executing gradient descent on the remaining data $\mathcal{D}_{\mathrm{rm}}$ [7, 8], and demonstrates nearly perfect performance as the retrained model f_{exact}.

RL [8] fine-tunes f_{pre} by replacing the labels of $\mathcal{D}_{\mathrm{fg}}$ with new different random labels. RL is implemented with both $\mathcal{D}_{\mathrm{fg}}$ and $\mathcal{D}_{\mathrm{rm}}$ involved and can easily approach the performance of the ideal retrained model f_{exact}. Nevertheless, existing research has highlighted the importance of restricted data access for approximate MU, particularly with only $\mathcal{D}_{\mathrm{fg}}$ accessible [28, 12]. Under this more practical setting with limited data access, RL excluding $\mathcal{D}_{\mathrm{rm}}$ in optimization shows unsatisfactory accuracy on $\mathcal{D}_{\mathrm{rm}}$ and $\mathcal{D}_{\mathrm{fg}}$, verified by our empirical results in this work.

SalUn [10] proposes to mask those parameters in f_{pre} that are sensitive to $\mathcal{D}_{\mathrm{fg}}$. In implementation, a parameter saliency mask is derived based on $\mathcal{D}_{\mathrm{fg}}$ only, and then is applied to the fine-tuning of f_{pre}. The fine-tuning of SalUn follows RL and similarly faces the aforementioned challenge as RL. SalUn excluding $\mathcal{D}_{\mathrm{rm}}$ in the fine-tuning shows uncompetitive unlearning results and the saliency mask does not bring significant performance gains to RL when $\mathcal{D}_{\mathrm{fg}}$ is only available. Another technique

with a similar idea of leveraging parameter sparsity is applying an additional ℓ_1 regularization on the parameters of f_{pre} based on RL [39]. Besides SalUn, SFR-on [40] further introduces gradient information via Hessians into the parameter mask and proposes a unified optimization framework with a fast-slow weight update. SFR-on also heavily relies on $\mathcal{D}_{\text{rm}}$ in computing the Fisher information matrix and the loss function.

L2UL [28] introduces adversarial examples (AEs) [41] and weight importance [42] to promote MU given access only to $\mathcal{D}_{\text{fg}}$. As claimed in L2UL, AEs w.r.t. $\mathcal{D}_{\text{fg}}$ can mimic the remaining data $\mathcal{D}_{\text{rm}}$, so that minimizing the cross-entropy loss on those AEs can benefit the performance on $\mathcal{D}_{\text{rm}}$. A parameter importance mask is further applied, where changes of weights that are less important in classifying $\mathcal{D}_{\text{fg}}$ get penalized. L2UL is particularly superior in instance-wise unlearning. However, the computational demand of L2UL is prohibitively heavy. Generating AEs with its default configuration (100 iterations, 200 AEs per image) for a dataset as large as ImageNet-1K (over 1.28M training images) renders the approach impractical.

COUN [29] leverages the contrastive learning [43] for MU. COUN basically fine-tunes f_{pre} by supervised training only on the remaining data $\mathcal{D}_{\text{rm}}$, and adds an additional contrastive learning loss on $\mathcal{D}_{\text{rm}}$. The rationale behind is to implicitly push forgetting representations towards those remaining representations with the highest semantic similarity by regularizing the augmented views of $\mathcal{D}_{\text{rm}}$ via the contrastive loss. As originally reported, COUN is trained solely on $\mathcal{D}_{\text{rm}}$ with improved results. To explore whether its contrastive learning paradigm can consistently enhance performance when only $\mathcal{D}_{\text{fg}}$ is available, we adapt COUN to operate with the RL method using only $\mathcal{D}_{\text{fg}}$, denoted as "COUN+RL" in our comparisons. Under this restricted data access, the contrastive learning implemented in COUN brings limited performance gains, as demonstrated in our results.

DELETE [12] is a distillation-based method relying only on $\mathcal{D}_{\text{fg}}$. In DELETE, a copy of the pretrained network is utilized as the teacher model, and the key is to apply a mask to the outputs from the teacher. This mask aims at altering the predictive probabilities of $\mathcal{D}_{\text{fg}}$ to approximate the ideal case and is implemented in a simple way: positions in the mask w.r.t. the ground-truth labels of $\mathcal{D}_{\text{fg}}$ are set to negative infinity and others are set to zero. There also exist other distillation-based methods, such as bad-teaching [11] and SCRUB [44], which adopt distinct designs to preserve $\mathcal{D}_{\text{rm}}$ and suppress $\mathcal{D}_{\text{fg}}$ during distillation. Among these, we include the most advanced DELETE into comparisons.

Beyond the aforementioned baselines, we further review several subspace-based MU methods that are more closely relevant with our LOFT. These methods typically define their subspaces in a *non-learnable* way based on network *parameters*. Gradient subspaces in convolution neural networks are identified in [45, 46] through eigen-decomposition. In [47], the subspace in large language models is constructed by applying the Gram-Schmidt process to the singular vectors of parameters. A latest unlearning method SEMU [48] performs SVD disentanglement on the gradients of $\mathcal{D}_{\text{fg}}$ to identify and update parameters mostly important for unlearning. However, our empirical results show that SEMU yields poor unlearning performance on SwinT. This limitation possibly stems from the fact that its decomposition, applied to parameters of 2d convolution and linear layers, may not adequately adapt to the attention structure in transformers. Thereby we omit comparisons with SEMU. In short, our LOFT fundamentally differs from these methods by leveraging a *learnable feature subspace* for approximate MU.

Researches in MU have been extended to a wide range of fields with different tasks. The class-wise and concept forgetting in image generation is explored in [10, 40]. Privacy attacks against MU methods and corresponding defenses are discussed in [49]. Unlearning from adversarially trained models is exploited to achieve superior unlearning performance and also strong adversarial robustness [50]. The concept of unnecessary unlearning is formalized in [51] with an algorithm to filter those unnecessary requests. MU can be an effective tool for defending the backdoor attacks [52, 53] through forgetting the backdoor triggers hidden in data. Moreover, MU in large language models has also received significant attention with extensive researches in recent years [54, 55, 56, 57].

B More Evidences for the Empirical Study in Section 3

The empirical study in Section 3 on the spectrum and reconstruction analyses motivates our low-dimensional feature subspace perspective for approximate MU. In the eigenvalue and reconstruction error results presented in Figure 2, we adopt a class-wise forgetting setting, where images correspond-

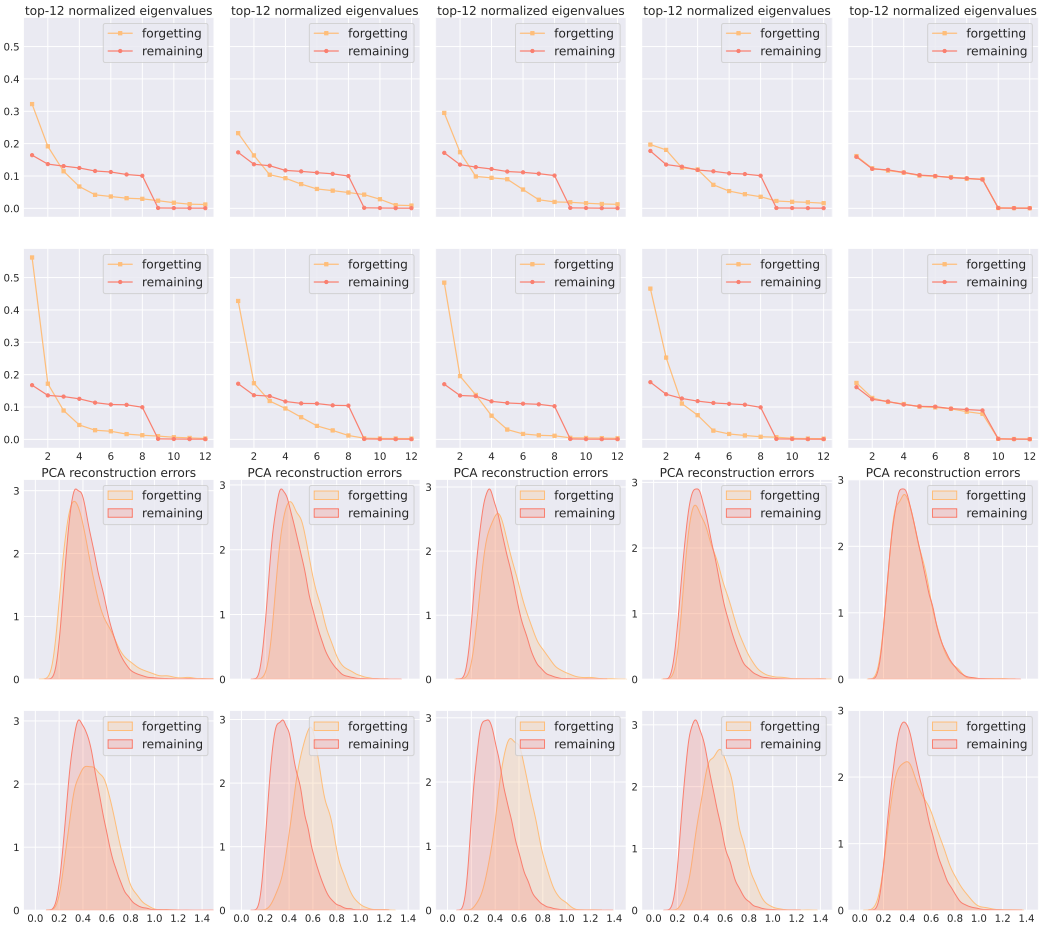


Figure S1: Spectrum and reconstruction analyses on features of $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ using the pretrained f_{pre} and the retrained f_{exact}. The results are presented as top-12 normalized eigenvalues and reconstruction errors. **Rows 1 & 2:** Eigenvalues for f_{pre} and f_{exact}, respectively. **Rows 3 & 4:** Reconstruction errors for f_{pre} and f_{exact}, respectively. **Columns 1-4:** $\mathcal{D}_{\text{fg}}$ including images from labels 1,3,5,7, respectively. **Column 5:** $\mathcal{D}_{\text{fg}}$ comprising images randomly sampled from all classes.

ing to label-0 in CIFAR10 are selected as the forgetting data $\mathcal{D}_{\text{fg}}$. To further strengthen this empirical study, additional evidences under different forgetting-remaining splits are provided in this section.

In addition to $\mathcal{D}_{\text{fg}}$ containing label-0 in Figure 2, we consider 4 other classes in CIFAR10, labels-1,3,5,7, as $\mathcal{D}_{\text{fg}}$, and execute the same spectrum and reconstruction analyses described in Section 3. The corresponding results are shown in the first 4 columns of Figure S1. Besides, we further extend our analyses to the more challenging instance-wise unlearning setting, where $\mathcal{D}_{\text{fg}}$ consists of images randomly-sampled across all classes; these results are presented in the last column of Figure S1.

As illustrated in Figure S1, for additional class-wise forgetting cases, the separability between $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ under a low-dimensional subspace from f_{exact} remains observable through the disparities in eigenvalues and the gaps in reconstruction errors. In contrast, for instance-wise unlearning, where $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ are highly similar and nearly drawn from the same distribution, the retrained model f_{exact} effectively learns that distribution from $\mathcal{D}_{\text{rm}}$, resulting in nearly identical eigenvalues and largely overlapped reconstruction errors. Thus, the separability achieved by directly applying PCA to features from f_{exact} is considerably weaker in instance-wise unlearning than in class-wise forgetting.

As we have remarked at the end of Section 3, the purpose of this analysis is *not to claim universal separability across all unlearning scenarios*, but rather to motivate our feature subspace learning perspective. For class-wise unlearning, the natural separability is evident, and we seek to approximate this ideal subspace from the pretrained f_{pre}.

For instance-wise unlearning, however, such separability cannot be expected to emerge naturally via unsupervised non-parametric methods like PCA, as $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ are nearly drawn from the same underlying distribution. *This highlights exactly why a dedicated subspace learning objective is necessary.* Instead of relying on natural emergence, our proposed method explicitly induces this separability. By actively minimizing the projected variance of the specific $\mathcal{D}_{\text{fg}}$ while preserving the structure of $\mathcal{D}_{\text{rm}}$ (as formulated in our objective of (1)), our method forces the projection matrix to discover a specific low-dimensional direction where $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ manifest separability. This explains why our approach remains effective for instance-wise unlearning in Section 5, despite the absence of natural PCA separability.

C Theoretical Discussions

C.1 Theoretical Discussions on Convergence

In this section, we show that $J(\mathbf{U})$ is L-smooth and thereby the convergence rate of Riemannian Adam on minimizing $J(\mathbf{U})$ is guaranteed.

$J(\mathbf{U})$ is a polynomial. Recall that

$$J(\mathbf{U}) = \underbrace{\Big(\frac{\operatorname{Tr}(\mathbf{U}^\top \mathbf{\Sigma}_{\text{fg}}^{\text{pre}} \mathbf{U})}{\operatorname{Tr}(\mathbf{\Sigma}_{\text{fg}}^{\text{pre}})}\Big)^2}_{J_{\text{fg}}} + \underbrace{\Big(\frac{\operatorname{Tr}\left(\mathbf{\Sigma}_{\text{rm}}^{\text{pre}} - \mathbf{U}\mathbf{U}^\top \mathbf{\Sigma}_{\text{rm}}^{\text{pre}} \mathbf{U}\mathbf{U}^\top\right)}{\operatorname{Tr}\left(\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}\right)}\Big)^2}_{J_{\text{rm}}}.$$

We can find that both terms of $J(\mathbf{U})$ are polynomials in the entries of $\mathbf{U}$:

- $J_{\text{fg}} = \frac{1}{c_1^2}\left(\operatorname{Tr}(\mathbf{U}^\top \mathbf{\Sigma}_{\text{fg}}^{\text{pre}} \mathbf{U})\right)^2$ is a polynomial of degree ≤ 4 with $c_1 = \operatorname{Tr}(\mathbf{\Sigma}_{\text{fg}}^{\text{pre}})$.
- $J_{\text{rm}} = \frac{1}{c_2^2}(c_2 - \operatorname{Tr}(\mathbf{U}^\top \mathbf{\Sigma}_{\text{rm}}^{\text{pre}} \mathbf{U}))^2$ is a polynomial of degree ≤ 8 with $c_2 = \operatorname{Tr}(\mathbf{\Sigma}_{\text{rm}}^{\text{pre}})$.

Hence $J(\mathbf{U}) = J_{\text{fg}} + J_{\text{rm}}$ is a polynomial (degree ≤ 8) with $\mathbf{U} \in \mathbb{R}^{d\times s}$. Consequently, $J(\mathbf{U})$ is C^∞ and all its partial derivatives are polynomials.

Euclidean Gradient is Lipschitz on the Compact Stiefel Manifold. Let $\nabla J(\mathbf{U})$ denote the Euclidean (standard) gradient in $\mathbb{R}^{d\times s}$. Because J is a polynomial of degree ≤ 8, $\nabla J(\mathbf{U})$ is a polynomial of degree ≤ 7, and its derivative $\nabla^2 J(\mathbf{U})$ (the Euclidean Hessian) is a polynomial of degree ≤ 6.

The Stiefel manifold $\operatorname{St}(d, s)$ is compact (closed and bounded in $\mathbb{R}^{d\times s}$). Therefore the continuous function $\mathbf{U} \mapsto \|\nabla^2 J(\mathbf{U})\|_{\text{op}}$ attains a maximum on this compact set. Define

$$L_E \coloneqq \max_{\mathbf{U}\in\operatorname{St}(d,s)} \|\nabla^2 J(\mathbf{U})\|_{\text{op}} < \infty.$$

By the mean value inequality (or Taylor's theorem with remainder), for any $\mathbf{U}, \mathbf{V} \in \operatorname{St}(d, s)$,

$$\|\nabla J(\mathbf{U}) - \nabla J(\mathbf{V})\|_F \leq L_E \|\mathbf{U} - \mathbf{V}\|_F.$$

Thus the Euclidean gradient is L_E-Lipschitz on the Stiefel manifold with respect to the Frobenius norm.

Riemannian Gradient and Its Lipschitz Property. For a submanifold of $\mathbb{R}^{d\times s}$, the Riemannian gradient $\operatorname{grad} J(\mathbf{U})$ is the orthogonal projection of the Euclidean gradient onto the tangent space:

$$\operatorname{grad} J(\mathbf{U}) = \Pi_{\mathbf{U}}(\nabla J(\mathbf{U})),$$

where $\Pi_{\mathbf{U}} : \mathbb{R}^{d\times s} \to T_{\mathbf{U}}\operatorname{St}(d, s)$ is the orthogonal projector. $\Pi_{\mathbf{U}}$ depends smoothly on $\mathbf{U}$ and satisfies $\|\Pi_{\mathbf{U}}\|_{\text{op}} = 1$.

We need to bound the difference between Riemannian gradients after parallel transport. Let $\tau_{\mathbf{U}\to\mathbf{V}}$ denote parallel transport along a minimizing geodesic from $\mathbf{U}$ to $\mathbf{V}$. A standard result in Riemannian optimization (see e.g. [58], Section 4.3) states:

Lemma C.1. *If a function f on a compact Riemannian manifold is C^2 and its Hessian is uniformly bounded, then its gradient is Lipschitz continuous. More precisely, there exists a constant $L_f > 0$ such that for all $\mathbf{U}, \mathbf{V}$ in the manifold,*

$$\| \operatorname{grad} f(\mathbf{U}) - \tau_{\mathbf{V}\to\mathbf{U}} \operatorname{grad} f(\mathbf{V})\| \leq L_f \operatorname{dist}(\mathbf{U}, \mathbf{V}),$$

where dist *denotes the Riemannian distance.*

Lemma C.2. *$J(\mathbf{U})$ is L-smooth.*

Proof. Because our $J(\mathbf{U})$ is C^∞ (hence C^2) and $\mathrm{St}(d, s)$ is compact, the Riemannian Hessian $\operatorname{Hess} J(\mathbf{U})$ is uniformly bounded in operator norm. Therefore Lemma C.1 applies, and we conclude that there exists a constant $L > 0$ such that

$$\| \operatorname{grad} J(\mathbf{U}) - \tau_{\mathbf{V}\to\mathbf{U}} \operatorname{grad} J(\mathbf{V})\| \leq L \operatorname{dist}(\mathbf{U}, \mathbf{V}), \qquad \forall \mathbf{U}, \mathbf{V} \in \mathrm{St}(d, s).$$

For completeness, we provide an elementary bound without invoking parallel transport directly, using the Euclidean Lipschitz property and the smoothness of the projector.

For any $\mathbf{U}, \mathbf{V} \in \mathrm{St}(d, s)$,

$$\begin{aligned}\| \operatorname{grad} J(\mathbf{U}) - \operatorname{grad} J(\mathbf{V})\|_F &= \|\Pi_{\mathbf{U}} \nabla J(\mathbf{U}) - \Pi_{\mathbf{V}} \nabla J(\mathbf{V})\|_F \\ &\leq \|\Pi_{\mathbf{U}}(\nabla J(\mathbf{U}) - \nabla J(\mathbf{V}))\|_F + \|(\Pi_{\mathbf{U}} - \Pi_{\mathbf{V}})\nabla J(\mathbf{V})\|_F.\end{aligned}$$

Using the Euclidean Lipschitz bound,

$$\|\nabla J(\mathbf{U}) - \nabla J(\mathbf{V})\|_F \leq L_E \|\mathbf{U} - \mathbf{V}\|_F.$$

For the second term, note that the map $\mathbf{U} \mapsto \Pi_{\mathbf{U}}$ is smooth on the compact Stiefel manifold, hence Lipschitz. Thus there exists a constant $C_\Pi > 0$ such that

$$\|\Pi_{\mathbf{U}} - \Pi_{\mathbf{V}}\|_{\mathrm{op}} \leq C_\Pi \|\mathbf{U} - \mathbf{V}\|_F.$$

Moreover, $\|\nabla J(\mathbf{V})\|_F$ is bounded on the compact set $\mathrm{St}(d, s)$; denote $G = \max_{\mathbf{V}\in\mathrm{St}(d,s)} \|\nabla J(\mathbf{V})\|_F < \infty$. Therefore,

$$\|(\Pi_{\mathbf{U}} - \Pi_{\mathbf{V}})\nabla J(\mathbf{V})\|_F \leq C_\Pi G \|\mathbf{U} - \mathbf{V}\|_F.$$

Combining both estimates, we obtain

$$\| \operatorname{grad} J(\mathbf{U}) - \operatorname{grad} J(\mathbf{V})\|_F \leq (L_E + C_\Pi G)\|\mathbf{U} - \mathbf{V}\|_F.$$

On a compact Riemannian manifold, the Euclidean distance $\|\mathbf{U} - \mathbf{V}\|_F$ and the Riemannian distance $\operatorname{dist}(\mathbf{U}, \mathbf{V})$ are equivalent (the ratio is bounded by the maximum of the inverse of the injectivity radius). Hence there exists a constant $L' > 0$ such that

$$\| \operatorname{grad} J(\mathbf{U}) - \operatorname{grad} J(\mathbf{V})\|_F \leq L' \operatorname{dist}(\mathbf{U}, \mathbf{V}).$$

This shows the Riemannian gradient is Lipschitz without explicitly using parallel transport. The Lipschitz constant L in the parallel-transport sense can be taken as the same L' because parallel transport is an isometry and the gradient vectors are identified appropriately. □

As $J(\mathbf{U})$ is smooth, by existing results in [59], we can draw the following conclusion:

Lemma C.3 (Theorem 4.1 in [59])**.** *If $J(\mathbf{U})$ is L-smooth, by using Riemannian Adam algorithm,*

$$\min_{t\in[T]} \|\nabla J(\mathbf{U}_t)\| \leq O\left(\frac{1}{\sqrt{T}} + (1 - \beta_2)^2\right).$$

This guarantees the convergence rate of Riemannian Adam on minimizing $J(\mathbf{U})$.

Here, T denotes the number of optimization steps and β_2 is the exponential decay rate for the second moment estimate (the uncentered variance) in the Riemannian Adam optimizer with a typical default value of 0.999. Clearly, the gradients of $J(\mathbf{U})$ approximate 0 given a large step T, and thereby the convergence is guaranteed.

C.2 Theoretical Discussions on Unlearning

A detailed version of Proposition 4.2 in Section 4 is stated as follows.

Proposition C.4. *For the pretrained model $f_{\text{pre}}(\cdot) = h_{\text{pre}}(g_{\text{pre}}(\cdot))$ and the retrained model $f_{\text{exact}}(\cdot) = h_{\text{exact}}(g_{\text{exact}}(\cdot))$, let their linear classifiers correspondingly be $h_{\text{pre}}(\boldsymbol{z}) = \mathbf{W}_{\text{pre}}^\top \boldsymbol{z}$ and $h_{\text{exact}}(\boldsymbol{z}) = \mathbf{W}_{\text{exact}}^\top \boldsymbol{z}$ (bias omitted for simplicity). Assuming the following conditions hold:*

- *The learned projection matrix $\hat{\mathbf{U}} \in \text{St}(d, s)$ obtained by solving* (1) *satisfies*

$$\text{Tr}(\hat{\mathbf{U}}^\top \boldsymbol{\Sigma}_{\text{fg}}^{\text{pre}} \hat{\mathbf{U}}) \le \epsilon_{\text{fg}}, \qquad \text{Tr}((\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top)\boldsymbol{\Sigma}_{\text{rm}}^{\text{pre}}(\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top)) \le \epsilon_{\text{rm}}$$

- *There exists $\delta_{\text{rm}} > 0$ such that for all $\boldsymbol{x} \in \mathcal{D}_{\text{rm}}$, $\|g_{\text{pre}}(\boldsymbol{x}) - g_{\text{exact}}(\boldsymbol{x})\|_2 \le \delta_{\text{rm}}$*
- *There exists constants $M_{\text{rm}}^{\text{exact}}$ and $M_{\text{fg}}^{\text{exact}} > 0$ such that*

$$\sup_{\boldsymbol{x} \in \mathcal{D}_{\text{rm}}} \|g_{\text{exact}}(\boldsymbol{x})\|_2 \le M_{\text{rm}}^{\text{exact}}, \quad \sup_{\boldsymbol{x} \in \mathcal{D}_{\text{fg}}} \|g_{\text{exact}}(\boldsymbol{x})\|_2 \le M_{\text{fg}}^{\text{exact}}.$$

- *The linear classifiers satisfy $\|\mathbf{W}_{\text{pre}} - \mathbf{W}_{\text{exact}}\|_2 \le \delta_W$.*
- *For the pretrained f_{pre}, features of $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ are centered, i.e.,*

$$\mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{rm}}}[g_{\text{pre}}(\boldsymbol{x})] = \mathbf{0}, \ \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_{\text{fg}}}[g_{\text{pre}}(\boldsymbol{x})] = \mathbf{0},$$

Then, for any $\boldsymbol{x} \in \mathcal{D}$, the expected output difference between the unlearned model $f_{\hat{\mathbf{U}}}$ and f_{exact} satisfies:

$$\mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_{\text{rm}}}\left[\|f_{\hat{\mathbf{U}}}(\boldsymbol{x}) - f_{\text{exact}}(\boldsymbol{x})\|_2\right] \le \|\mathbf{W}_{\text{pre}}\|_2(\delta_{\text{rm}} + \sqrt{\epsilon_{\text{rm}}}) + \delta_W M_{\text{rm}}^{\text{exact}}$$
$$\mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_{\text{fg}}}\left[\|f_{\hat{\mathbf{U}}}(\boldsymbol{x}) - f_{\text{exact}}(\boldsymbol{x})\|_2\right] \le \|\mathbf{W}_{\text{pre}}\|_2\sqrt{\epsilon_{\text{fg}}} + \|\mathbf{W}_{\text{exact}}\|_2 M_{\text{fg}}^{\text{exact}}$$

Proof. We treat the two cases separately.

For $\boldsymbol{x} \in \mathcal{D}_{\text{rm}}$, we have:

$$\begin{aligned}
\|f_{\hat{\mathbf{U}}}(\boldsymbol{x}) - f_{\text{exact}}(\boldsymbol{x})\|_2 &= \|\mathbf{W}_{\text{pre}}^\top \hat{\mathbf{U}}\hat{\mathbf{U}}^\top g_{\text{pre}}(\boldsymbol{x}) - \mathbf{W}_{\text{exact}}^\top g_{\text{exact}}(\boldsymbol{x})\|_2 \\
&\le \|\mathbf{W}_{\text{pre}}^\top \hat{\mathbf{U}}\hat{\mathbf{U}}^\top g_{\text{pre}}(\boldsymbol{x}) - \mathbf{W}_{\text{pre}}^\top g_{\text{pre}}(\boldsymbol{x})\|_2 \\
&+ \|\mathbf{W}_{\text{pre}}^\top g_{\text{pre}}(\boldsymbol{x}) - \mathbf{W}_{\text{pre}}^\top g_{\text{exact}}(\boldsymbol{x})\|_2 \\
&+ \|(\mathbf{W}_{\text{pre}} - \mathbf{W}_{\text{exact}})^\top g_{\text{exact}}(\boldsymbol{x})\|_2.
\end{aligned} \tag{S1}$$

The first term is bounded by $\|\mathbf{W}_{\text{pre}}\|_2\|(\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top)g_{\text{pre}}(\boldsymbol{x})\|_2$. By the assumption on the solution of LOFT, we have $\mathbb{E}_{\boldsymbol{x} \in \mathcal{D}_{\text{rm}}}\|\mathbf{W}_{\text{pre}}\|_2\|(\mathbf{I} - \hat{\mathbf{U}}\hat{\mathbf{U}}^\top)g_{\text{pre}}(\boldsymbol{x})\|_2 \le \|\mathbf{W}_{\text{pre}}\|_2\sqrt{\epsilon_{\text{rm}}}$. The second term is bounded by $\|\mathbf{W}_{\text{pre}}\|_2\delta_{\text{rm}}$. The third term is bounded by $\|\mathbf{W}_{\text{pre}} - \mathbf{W}_{\text{exact}}\|_2\|g_{\text{exact}}(\boldsymbol{x})\|_2 \le \delta_W M_{\text{rm}}^{\text{exact}}$. Summing these gives the claimed bound for $\mathcal{D}_{\text{rm}}$.

For $\boldsymbol{x} \in \mathcal{D}_{\text{fg}}$, similarly, we have:

$$\begin{aligned}
\|f_{\hat{\mathbf{U}}}(\boldsymbol{x}) - f_{\text{exact}}(\boldsymbol{x})\|_2 &\le \|\mathbf{W}_{\text{pre}}^\top \hat{\mathbf{U}}\hat{\mathbf{U}}^\top g_{\text{pre}}(\boldsymbol{x})\|_2 + \|\mathbf{W}_{\text{exact}}^\top g_{\text{exact}}(\boldsymbol{x})\|_2 \\
&\le \|\mathbf{W}_{\text{pre}}\|_2\|\hat{\mathbf{U}}\hat{\mathbf{U}}^\top g_{\text{pre}}(\boldsymbol{x})\|_2 + \|\mathbf{W}_{\text{exact}}\|_2\|g_{\text{exact}}(\boldsymbol{x})\|_2.
\end{aligned} \tag{S2}$$

By the assumption on the solution of LOFT, the first term is bounded by $\|\mathbf{W}_{\text{pre}}\|_2\sqrt{\epsilon_{\text{fg}}}$ in expectation. The second term is bounded by $\|\mathbf{W}_{\text{exact}}\|_2 M_{\text{fg}}^{\text{exact}}$. Therefore the proof is finished. □

Proposition 4.2 establishes that the expected output discrepancy between LOFT and the exact unlearned model is explicitly controlled by the two subspace quality measures, ϵ_{fg} and ϵ_{rm}, which LOFT directly minimizes, and other proximity assumptions. If LOFT succeeds in making ϵ_{fg} and ϵ_{rm} small, and if the pretrained and retrained models are close on the remaining data, then the unlearned model will be close to the gold-standard f_{exact}.

Remark. As we have noted in the end of Section 4, LOFT does not update the parameters of the pretrained model f_{pre}, and thereby cannot be adapted to the existing certified unlearning framework that certifies the parameter proximity between f_{pre} and the unlearned model. Proposition 4.2 is a conditional analysis to support the approximation behavior of LOFT, rather than a guarantee of certified unlearning. A more rigorous theoretical development for subspace approximate MU remains a valuable direction for future work.

Table S1: A detailed analysis on the time cost of the whole pipeline of LOFT. Empirical setups follow the same as that of Tables 1 and 2.

method	offline one-shot feature fetching	on-demand unlearning	
		calculating covariances	optimizing $\mathbf{U}$
LOFT	11.82	1.05	0.24

D In-depth Efficiency Analysis into LOFT in Time and Memory

We provide a comprehensive analysis on the time and memory cost of LOFT throughout its entire pipeline. As demonstrated in Algorithm 1, LOFT can be partitioned into two stages, *offline one-shot feature fetching* and *on-demand unlearning*. In the offline stage, LOFT performs a full pass over the training dataset using the pretrained network f_{pre}, in order to extract the remaining and forgetting features ($\mathbf{Z}_{\text{rm}}^{\text{pre}}$ and $\mathbf{Z}_{\text{fg}}^{\text{pre}}$). Then, in the unlearning stage, given a forgetting request, LOFT computes the corresponding covariance matrices ($\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$), and executes its approximate unlearning to learn a projection matrix $\mathbf{U} \in \text{St}(d \times s)$ by solving (1) based on $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$.

Despite iterating through the entire training dataset for feature extraction, LOFT remains highly efficient in its offline stage, as summarized in Table S1. For this evaluation, we adopt the same setup as Tables 1 and 2 in Section 5.2, and employ Tiny-ImageNet with Swin-T, where 4 of 200 classes are randomly selected as the forgetting data $\mathcal{D}_{\text{fg}}$. As demonstrated in Table S1, feature extraction is performed via a single inference-only forward pass through the pretrained f_{pre} without backward gradient computations, costing only 11.82 seconds on a GPU. Subsequently, the two covariance matrices (each 768×768) are computed from the resulting feature matrices (98,000×768 of $\mathbf{Z}_{\text{rm}}^{\text{pre}}$ and 2,000×768 of $\mathbf{Z}_{\text{fg}}^{\text{pre}}$), a process that finishes in mere 1.05 seconds on CPU. Compared with the time consumption of other baselines in Table 2, the entire pipeline of LOFT still retains a clear efficiency advantage.

Regarding the memory cost, the peak overhead of LOFT arises from storing the intermediate training features (100,000×768) during its preparation stage, which nevertheless remains minimal at approximately 300 MB. Aside from that, the unlearning stage works on only two feature covariance matrices of 768×768, and learns a projection matrix of 768×250 without updating the parameters of the pretrained network f_{pre}. As a result, the overall memory footprint of LOFT remains nearly negligible.

Remark. LOFT is highly efficient particularly for large-scale data. During its unlearning stage, LOFT learns $\mathbf{U}$ based on the two fixed pre-computed feature covariance matrices $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$. This discards the need for iteratively visiting the raw data or updating the pretrained network f_{pre}. Meanwhile, the feature fetching stage is thereby a one-time cost by processing the training data in a single, gradient-free forward pass to yield $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$ and $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$. We think such computational complexity of LOFT is acceptable for deep learning in practice and makes LOFT particularly suitable for large-scale data scenarios. In contrast, mainstream unlearning methods face significant scalability bottlenecks: they either *(i)* require *repeated* forward-*backward* passes on full data with gradients calculations, which is prohibitively expensive for large-scale data, or *(ii)* rely solely on the forgetting data for efficiency, but leading to limited performance due to the absence of the remaining data information. For example, the ImageNet-1K dataset includes over 1.28M training images. Existing methods that calculates gradients on both forgetting and remaining data with multiple epochs face nearly prohibitive calculations on ImageNet-1K, while LOFT can be efficiently and effectively executed by iterating the ImageNet-1K training dataset only once. Therefore, LOFT strikes a favorable balance and achieves effective unlearning by leveraging the forgetting and remaining feature covariance matrices through a single, computationally-manageable forward pass, thereby offering remarkable scalability in practical applications.

Table S2: Training hyper-parameters of different MU methods w.r.t. results of multi-class unlearning in Table 1 with Swin-T on Tiny-ImageNet.

method	$\mathcal{D}_{\mathrm{rm}}$	$\mathcal{D}_{\mathrm{fg}}$	hyper-parameters
f_{pre}	✓	✓	20 epochs, lr $= 10^{-4}$, cosine scheduler
f_{exact}	✓	✗	20 epochs, lr $= 10^{-4}$, cosine scheduler
GA	✗	✓	10 epochs, lr $= 2 \times 10^{-6}$, constant scheduler
GA	✓	✓	5 epochs, lr $= 10^{-5}$, constant scheduler
RL	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler
RL	✓	✓	10 epochs, lr $= 10^{-4}$, cosine scheduler
SalUn	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler, saliency sparsity 50%
SalUn	✓	✓	10 epochs, lr $= 10^{-4}$, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	10 epochs, lr $= 2 \times 10^{-5}$, cosine scheduler
COUN+RL	✓	✓	10 epochs, lr $= 2 \times 10^{-4}$, cosine scheduler
L2UL	✗	✓	lr $= 10^{-5}$, constant scheduler, regularization coefficient = 1.0
DELETE	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}}$	only $\mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}}$	50 steps, $s = 250$, lr $= 1$, constant scheduler

E Implementation Details of Experiments in Main Text

E.1 Training Details of Multi-Class Unlearning

For multi-class unlearning in Table 1 with Swin-T on Tiny-ImageNet, we adopt the PyTorch-released checkpoint trained on ImageNet-1K as a starting point, and fine-tune this checkpoint on Tiny-ImageNet to obtain the pretrained model. In the repeated 3 experiments of Table 1, the randomly selected 4 forgetting labels (classes) are $\{11, 83, 115, 153\}$, $\{44, 65, 150, 168\}$ and $\{53, 57, 108, 179\}$, respectively. All compared methods use the AdamW optimizer [60] with a weight decay of 0.05 and a batch size of 128. In L2UL [28], to generate adversarial examples, the ℓ_2-PGD targeted attack is employed with a step size of 0.1, a perturbation bound of 0.4 and 100 iteration steps, and 200 adversarial examples are generated per image. The termination of the L2UL unlearning is carefully selected until the accuracy on the training forgetting data is sufficiently low without a substantial accuracy drop on the training remaining data. All the training hyper-parameters to obtain results in Table 1 are listed in Table S2. Our LOFT adopts the Riemannian Adam optimizer [19] for the projection matrix $\mathbf{U}$ with a weight decay of 0.05, where the penultimate layer feature dimension with Swin-T on Tiny-ImageNet is $d = 768$.

E.2 Training Details of Single-Class Unlearning

For single-class unlearning in Table 3 with ResNet50 on ImageNet-1K, we deploy the PyTorch-released checkpoint that is exactly pretrained on ImageNet-1K as the pretrained model f_{pre}. Regarding the retrained model f_{exact} for unlearning, we train ResNet50 from scratch on the remaining data $\mathcal{D}_{\mathrm{rm}}$ for 90 epochs on 4 NVIDIA GeForce RTX 4090 GPUs, which runs for around 24 hours. In the repeated 3 experiments of Table 3, the randomly selected forgetting labels are 97, 316, and 852, respectively. All the compared methods use the SGD optimizer [61] with a weight decay of 1×10^{-4}, the momentum of 0.9, and a batch size of 128. All the training hyper-parameters for the results in Table 3 are listed Table S3. Our LOFT keeps the settings of using the Riemannian Adam optimizer [19] for $\mathbf{U}$ with a weight decay of 0.05, where the penultimate layer feature dimension with ResNet50 on ImageNet-1K is $d = 2048$.

E.3 Training Details of Instance Unlearning

For instance unlearning in Table 4 with Swin-T on Tiny-ImageNet, the basic settings are with that in Section E.1. 1,000 samples from the total 100,000 training samples are randomly selected as the forgetting data $\mathcal{D}_{\mathrm{fg}}$ (1%). Each of the repeated 3 experiments of Table 4 is corresponding to different random seeds, i.e., 0, 1 and 2. All compared methods use the AdamW optimizer [60] with a weight decay of 0.05 and a batch size of 128. The hyper-parameters are listed in Table S4. Our LOFT uses the Riemannian Adam optimizer [19] with a weight decay of 0.05.

Table S3: Training hyper-parameters of different MU methods w.r.t. results of single-class unlearning in Table 3 with ResNet50 on ImageNet-1K.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	hyper-parameters
f_{pre}	✓	✓	-
f_{exact}	✓	✗	90 epochs, lr $= 10^{-1}$, LR decay: 0.1 every 30 epochs
GA	✗	✓	3 epochs, lr $= 10^{-4}$, constant scheduler
RL	✗	✓	5 epochs, lr $= 5 \times 10^{-5}$, cosine scheduler
SalUn	✗	✓	10 epochs, lr $= 10^{-4}$, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	3 epochs, lr $= 10^{-6}$, cosine scheduler
DELETE	✗	✓	5 epochs, lr $= 10^{-1}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	100 steps, $s = 500$, lr $= 10$, constant scheduler

Table S4: Training hyper-parameters of different MU methods w.r.t. results of instance unlearning in Table 4 with Swin-T on Tiny-ImageNet.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	hyper-parameters
f_{pre}	✓	✓	20 epochs, lr $= 10^{-4}$, cosine scheduler
f_{exact}	✓	✗	20 epochs, lr $= 10^{-4}$, cosine scheduler
GA	✗	✓	10 epochs, lr $= 6 \times 10^{-6}$ constant scheduler
RL	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler
SalUn	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	10 epochs, lr $= 10^{-5}$, cosine scheduler
L2UL	✗	✓	lr $= 10^{-4}$, constant scheduler, regularization coefficient = 1.0
DELETE	✗	✓	10 epochs, lr $= 2 \times 10^{-5}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	100 steps, $s = 300$, lr $= 1$, constant scheduler

E.4 Sensitivity Analysis into Weighted Losses for LOFT

The optimization objective of (1) for LOFT consists of two terms, J_{fg} and J_{rm}, which target the minimization of forgetting data information and the maximization of remaining data information, respectively. While the two terms are equally weighted by default, we explore a weighted sum for J_{fg} and J_{rm} controlled by a coefficient $\rho \in [0, 1]$, defined as follows:

$$\min_{\mathbf{U} \in \text{St}(d,s)} J(\mathbf{U}) = (1-\rho) \cdot \underbrace{\left(\frac{\text{Tr}(\mathbf{U}^\top \mathbf{\Sigma}_{\text{fg}}^{\text{pre}} \mathbf{U})}{\text{Tr}(\mathbf{\Sigma}_{\text{fg}}^{\text{pre}})} \right)^2}_{J_{\text{fg}}} + \rho \cdot \underbrace{\left(\frac{\text{Tr}\left(\mathbf{\Sigma}_{\text{rm}}^{\text{pre}} - \mathbf{U}\mathbf{U}^\top \mathbf{\Sigma}_{\text{rm}}^{\text{pre}} \mathbf{U}\mathbf{U}^\top\right)}{\text{Tr}\left(\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}\right)} \right)^2}_{J_{\text{rm}}}, \tag{S3}$$

Figure S2 illustrates the sensitivity of unlearning performance of LOFT to the loss coefficient ρ. In this experiment, 4 of 200 classes in Tiny-ImageNet is chosen as the forgetting data, following the settings of Table 1. As observed, a lower ρ emphasizes the elimination of information from the forgetting set (J_{fg}) with a performance degradation on the remaining set, while a higher ρ prioritizes the preservation of remaining data (J_{rm}), hindering unlearning efficacy. Additionally, our results suggest that assigning an insufficient weight to J_{fg} (small values of $(1-\rho)$) can be detrimental (high accuracy on forgetting data), when dealing with a limited amount of forgetting data (2%) in this case. We therefore suggest that a simple, equal-weighting scheme for J_{fg} and J_{rm} is typically sufficient and effective.

F Extreme Unlearning and Continual Unlearning

F.1 Extreme Unlearning

Setups. We conduct evaluations to investigate the robustness of MU methods under an extreme scenario, where a very high ratio (over 90%) of the training samples is required to be removed. This situation may also arise in practice, such as deleting data across a large number of users on one occasion. In this experiment, 180 of 200 classes (90%) in Tiny-ImageNet are randomly selected as $\mathcal{D}_{\text{fg}}$, and experiments are repeated 3 times. We adopt different random seeds, i.e., 0, 1 and 2, to guarantee that the forgetting 180 labels are different in each experiment. All compared methods use

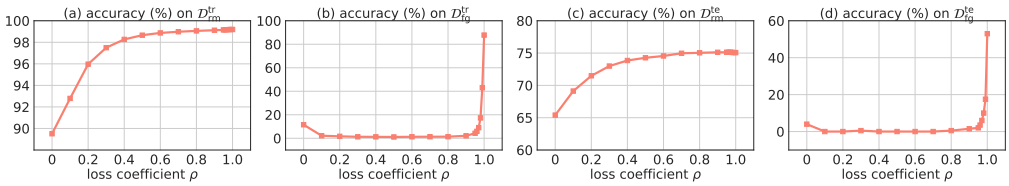


Figure S2: Unlearning performance w.r.t. varied values of the loss coefficient ρ. Empirical setups follow the same as that of Table 1.

Table S5: Training hyper-parameters of different MU methods under extreme unlearning with Swin-T on Tiny-ImageNet.

method	$\mathcal{D}_{\mathrm{rm}}$	$\mathcal{D}_{\mathrm{fg}}$	hyper-parameters
f_{pre}	✓	✓	20 epochs, lr $= 10^{-4}$, cosine scheduler
f_{exact}	✓	✗	20 epochs, lr $= 10^{-4}$, cosine scheduler
GA	✗	✓	constant scheduler
GA	✓	✓	constant scheduler
RL	✗	✓	5 epochs, lr $= 5 \times 10^{-7}$, cosine scheduler
RL	✓	✓	10 epochs, lr $= 2 \times 10^{-5}$, cosine scheduler
SalUn	✗	✓	5 epochs, lr $= 5 \times 10^{-7}$, cosine scheduler, saliency sparsity 50%
SalUn	✓	✓	10 epochs, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	5 epochs, lr $= 10^{-4}$, cosine scheduler
COUN+RL	✓	✓	5 epochs, lr $= 10^{-4}$, cosine scheduler
L2UL	✗	✓	lr $= 2 \times 10^{-4}$, constant scheduler, regularization coefficient = 1.0
DELETE	✗	✓	10 epochs, lr $= 2 \times 10^{-4}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\mathrm{rm}}^{\mathrm{pre}}$	only $\mathbf{\Sigma}_{\mathrm{fg}}^{\mathrm{pre}}$	50 steps, $s = 18$, lr $= 1$, constant scheduler

the AdamW optimizer [60] with weight decay of 0.05 and a batch size of 128. The hyper-parameters are listed in Table S5. In Table S5, the $\mathcal{D}_{\mathrm{fg}}$-based GA and SalUn involving both $\mathcal{D}_{\mathrm{rm}}$ and $\mathcal{D}_{\mathrm{fg}}$ are highly-sensitive to the choice of the learning rate under this extreme unlearning scenario. To obtain satisfactory results, we use different learning rates for GA and SalUn in each experiment w.r.t. different random seeds. For GA, the hyper-parameters are {4 epochs, lr $= 3 \times 10^{-7}$}, {7 epochs, lr $= 2 \times 10^{-7}$} and {3 epochs, lr $= 4 \times 10^{-7}$} w.r.t. random seeds 0, 1 and 2. For SalUn based on $\mathcal{D}_{\mathrm{rm}}$ and $\mathcal{D}_{\mathrm{fg}}$, the hyper-parameters are {lr $= 2 \times 10^{-4}$}, {lr $= 1.5 \times 10^{-4}$} and {lr $= 10^{-4}$} w.r.t. random seeds 0, 1 and 2. For L2UL, we only generate two adversarial examples per image for efficiency. Our LOFT optimizes $\mathbf{U}$ through the Riemannian Adam optimizer [19] with the weight decay of 0.05.

Results. Table S6 shows results of extreme unlearning with Swin-T on Tiny-ImageNet. In this extreme unlearning scenario, methods that directly update f_{pre} struggle to maintain good performance on $\mathcal{D}_{\mathrm{rm}}$, as the gradient updates to f_{pre} are dominated by penalizing performance on $\mathcal{D}_{\mathrm{fg}}$ with a substantial number of samples. In contrast, our LOFT does not modify f_{pre} and its projection matrix is optimized on the feature covariances w.r.t. $\mathcal{D}_{\mathrm{rm}}$ and $\mathcal{D}_{\mathrm{fg}}$, which is less affected by the relative data sizes of $\mathcal{D}_{\mathrm{rm}}$ and $\mathcal{D}_{\mathrm{fg}}$. LOFT achieves the best average gap (Avg.G.) of 0.42 with about 0.05% parameters w.r.t. the given model, and takes the shortest running time (RTE) about 0.29 seconds.

F.2 Continual Unlearning

Setups. We provide evaluations for different MU methods under a practical scenario where multiple unlearning requests are in a continual order. To be specific, we consider a 4-round continual unlearning. In each round, 2 classes (1%) are randomly selected as the forgetting data $\mathcal{D}_{\mathrm{fg}}$, and the forgetting labels at each round are {18,170}, {80,49}, {117,51}, and {16,118}, respectively. We introduce two new metrics $\mathrm{Acc}_{\mathrm{fgp}}^{\mathrm{tr}}$ and $\mathrm{Acc}_{\mathrm{fgp}}^{\mathrm{te}}$, i.e., accuracy on the forgetting data in previous rounds $\mathcal{D}_{\mathrm{fgp}}$, to evaluate whether the model indeed forget data from early unlearning requests. Results of continual unlearning are shown in Table S7 including multiple mainstream MU methods.

In this continual unlearning, our LOFT optimizes to guarantee the unlearning performance on forgetting data from previous rounds $\mathcal{D}_{\mathrm{fgp}}$, where we introduce an additional loss J_{fgp} to the objective

Table S6: Comparison on performance under extreme unlearning with Swin-T on Tiny-ImageNet.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathbf{Acc}^{\mathbf{tr}}_{\mathbf{rm}}$	$\mathbf{Acc}^{\mathbf{tr}}_{\mathbf{fg}}$	$\mathbf{Acc}^{\mathbf{te}}_{\mathbf{rm}}$	$\mathbf{Acc}^{\mathbf{te}}_{\mathbf{fg}}$	MIA	Avg.G.↓	RTE ↓	# param. (%)
f_{pre}	✓	✓	$99.64_{\pm 0.02}$	$99.63_{\pm 0.00}$	$75.43_{\pm 2.61}$	$74.45_{\pm 0.29}$	$95.34_{\pm 0.43}$	-	882.05	27,673,154 (100%)
f_{exact}	✓	✗	$99.97_{\pm 0.01}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$90.57_{\pm 0.75}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	0.00	95.17	27,673,154 (100%)
GA	✗	✓	$14.26_{\pm 6.06}$ (85.71)	$9.61_{\pm 4.36}$ (9.61)	$12.20_{\pm 4.61}$ (78.37)	$8.10_{\pm 3.46}$ (8.10)	$11.41_{\pm 5.19}$ (11.41)	38.64	178.07	27,673,154 (100%)
GA	✓	✓	$19.96_{\pm 2.63}$ (80.01)	$11.42_{\pm 3.45}$ (11.42)	$16.67_{\pm 1.14}$ (73.90)	$9.59_{\pm 2.70}$ (9.59)	$13.57_{\pm 4.04}$ (13.57)	37.70	242.46	27,673,154 (100%)
RL	✗	✓	$26.98_{\pm 3.11}$ (72.99)	$16.33_{\pm 2.69}$ (16.33)	$23.43_{\pm 2.05}$ (67.14)	$13.75_{\pm 2.35}$ (13.75)	$5.39_{\pm 0.75}$ (5.39)	35.12	195.26	27,673,154 (100%)
RL	✓	✓	$98.77_{\pm 0.22}$ (1.20)	$20.40_{\pm 0.83}$ (20.40)	$90.03_{\pm 0.58}$ (0.54)	$15.38_{\pm 0.47}$ (15.38)	$1.10_{\pm 0.51}$ (1.10)	7.72	431.45	27,673,154 (100%)
SalUn	✗	✓	$27.39_{\pm 0.48}$ (72.58)	$17.57_{\pm 3.28}$ (17.57)	$23.30_{\pm 2.1}$ (67.27)	$15.05_{\pm 2.65}$ (15.05)	$12.25_{\pm 1.57}$ (12.25)	36.95	235.23	13,836,577 (50%)
SalUn	✓	✓	$98.72_{\pm 0.37}$ (1.25)	$13.88_{\pm 3.09}$ (13.88)	$87.93_{\pm 1.33}$ (2.64)	$11.61_{\pm 2.29}$ (11.61)	$2.74_{\pm 1.14}$ (2.14)	6.31	476.78	13,836,577 (50%)
COUN+RL	✗	✓	$35.77_{\pm 3.93}$ (64.20)	$27.33_{\pm 0.42}$ (27.33)	$33.10_{\pm 3.50}$ (57.47)	$24.07_{\pm 0.25}$ (24.07)	$3.28_{\pm 3.46}$ (3.28)	35.27	337.69	27,673,154 (100%)
COUN+RL	✓	✓	$99.86_{\pm 0.05}$ (0.11)	$24.50_{\pm 0.63}$ (24.50)	$88.50_{\pm 1.61}$ (2.07)	$20.01_{\pm 0.89}$ (20.01)	$4.05_{\pm 0.44}$ (4.05)	10.15	357.45	27,673,154 (100%)
L2UL	✗	✓	$23.90_{\pm 7.94}$ (76.07)	$15.00_{\pm 3.50}$ (15.00)	$19.10_{\pm 6.35}$ (71.47)	$11.91_{\pm 2.50}$ (11.91)	$3.66_{\pm 1.95}$ (3.66)	35.63	7361.76	27,673,154 (100%)
DELETE	✗	✓	$97.37_{\pm 0.36}$ (2.60)	$6.95_{\pm 0.31}$ (6.95)	$81.07_{\pm 1.53}$ (9.50)	$10.96_{\pm 0.48}$ (10.96)	$1.16_{\pm 0.16}$ (1.16)	6.23	446.62	27,673,154 (100%)
LOFT	only $\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$	only $\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$	$99.63_{\pm 0.18}$ (**0.34**)	$0.03_{\pm 0.03}$ (**0.03**)	$92.17_{\pm 0.65}$ (**1.60**)	$0.02_{\pm 0.02}$ (**0.02**)	$0.08_{\pm 0.12}$ (**0.08**)	**0.42**	**0.29**	**13,824 (0.05%)**

Table S7: Comparison results of Swin-T on Tiny-ImageNet under continual unlearning.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathcal{D}_{\text{fgp}}$	$\mathbf{Acc}^{\mathbf{tr}}_{\mathbf{rm}}$	$\mathbf{Acc}^{\mathbf{tr}}_{\mathbf{fg}}$	$\mathbf{Acc}^{\mathbf{tr}}_{\mathbf{fgp}}$	$\mathbf{Acc}^{\mathbf{te}}_{\mathbf{rm}}$	$\mathbf{Acc}^{\mathbf{te}}_{\mathbf{fg}}$	$\mathbf{Acc}^{\mathbf{te}}_{\mathbf{fgp}}$
					Round-1				
f_{pre}	✓	✓	-	99.63	99.90	-	74.49	80.00	-
f_{exact}	✓	✗	-	99.65 (0.00)	0.00 (0.00)	-	75.08 (0.00)	0.00 (0.00)	-
RL	✗	✓	-	97.00 (2.65)	8.50 (8.50)	-	70.87 (4.21)	1.00 (1.00)	-
RL	✓	✓	-	99.93 (0.28)	4.90 (4.90)	-	74.34 (0.74)	2.00 (2.00)	-
DELETE	✗	✓	-	99.21 (0.44)	8.00 (8.00)	-	73.87 (0.62)	0.00 (0.00)	-
LOFT	$\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$	-	96.86 (2.77)	4.20 (4.20)	-	72.65 (2.43)	0.00 (0.00)	-
					Round-2				
f_{pre}	✓	✓	✓	99.63	99.60	99.90	74.55	69.00	80.00
f_{exact}	✓	✗	✗	99.69 (0.00)	0.00 (0.00)	0.00 (0.00)	75.57 (0.00)	0.00 (0.00)	0.00 (0.00)
RL	✗	✓	✗	80.30 (19.39)	6.30 (6.30)	1.00 (1.00)	59.68 (15.89)	2.00 (2.00)	0.00 (0.00)
RL	✓	✓	✗	99.98 (0.29)	6.30 (6.30)	1.00 (1.00)	74.37 (1.20)	1.00 (1.00)	0.00 (0.00)
DELETE	✗	✓	✗	98.33 (1.30)	10.60 (10.60)	3.90 (3.90)	72.40 (3.17)	5.00 (5.00)	0.00 (0.00)
LOFT	$\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$	95.77 (3.92)	7.50 (7.50)	0.50 (0.50)	72.01 (3.56)	4.00 (4.00)	1.00 (1.00)
					Round-3				
f_{pre}	✓	✓	✓	99.62	99.60	99.75	74.53	69.00	74.50
f_{exact}	✓	✗	✗	99.66 (0.00)	0.00 (0.00)	0.00 (0.00)	73.06 (0.00)	0.00 (0.00)	0.00 (0.00)
RL	✗	✓	✗	45.16 (54.50)	0.70 (0.70)	0.30 (0.30)	36.27 (36.79)	0.00 (0.00)	0.50 (0.50)
RL	✓	✓	✗	99.98 (0.32)	7.70 (7.70)	0.30 (0.30)	73.79 (0.73)	2.00 (2.00)	0.00 (0.00)
DELETE	✗	✓	✗	96.27 (3.39)	5.10 (5.10)	3.85 (3.85)	70.05 (3.01)	3.00 (3.00)	1.50 (1.50)
LOFT	$\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$	96.39 (3.27)	1.70 (1.70)	0.35 (0.35)	72.80 (0.26)	0.00 (0.00)	0.50 (0.50)
					Round-4				
f_{pre}	✓	✓	✓	99.63	99.30	99.80	74.54	73.00	75.33
f_{exact}	✓	✗	✗	99.67 (0.00)	0.00 (0.00)	0.00 (0.00)	75.72 (0.00)	0.00 (0.00)	0.00 (0.00)
RL	✗	✓	✗	10.96 (88.71)	0.30 (0.30)	0.00 (0.00)	9.97 (65.75)	1.00 (1.00)	0.00 (0.00)
RL	✓	✓	✗	99.98 (0.31)	17.60 (17.60)	1.07 (1.07)	73.43 (2.29)	9.00 (9.00)	0.00 (0.00)
DELETE	✗	✓	✗	93.07 (6.60)	3.80 (3.80)	2.43 (2.43)	67.91 (7.81)	3.00 (3.00)	1.33 (1.33)
LOFT	$\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$	$\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$	96.48 (3.19)	5.50 (5.50)	0.07 (0.07)	72.50 (3.22)	3.00 (3.00)	0.00 (0.00)

of $J(\mathbf{U})$ in (1) for achieving the task of continual unlearning as follows:

$$\min_{\mathbf{U}\in\text{St}(d,s)} \underbrace{\left(\frac{\text{Tr}(\mathbf{U}^\top \mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}\mathbf{U})}{\text{Tr}(\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}})}\right)^2}_{J_{\text{fgp}}} + \underbrace{\left(\frac{\text{Tr}(\mathbf{U}^\top \mathbf{\Sigma}^{\text{pre}}_{\text{fg}}\mathbf{U})}{\text{Tr}(\mathbf{\Sigma}^{\text{pre}}_{\text{fg}})}\right)^2}_{J_{\text{fg}}} + \underbrace{\left(\frac{\text{Tr}\left(\mathbf{\Sigma}^{\text{pre}}_{\text{rm}} - \mathbf{U}\mathbf{U}^\top \mathbf{\Sigma}^{\text{pre}}_{\text{rm}}\mathbf{U}\mathbf{U}^\top\right)}{\text{Tr}\left(\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}\right)}\right)^2}_{J_{\text{rm}}}. \quad \text{(S4)}$$

J_{fgp} takes a similar form as J_{fg} and measures the projected variance of the covariance matrix $\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$ w.r.t. $\mathcal{D}_{\text{fgp}}$ captured within the subspace spanned by $\mathbf{U}$. Hence, minimizing J_{fgp} achieves unlearning on $\mathcal{D}_{\text{fgp}}$. Note that J_{fgp} can be efficiently implemented without additional computation, since $\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$ has been accessed in previous rounds.

Results. During the multiple rounds of unlearning, other MU methods constantly modify the parameters of the pretrained f_{pre} and directly affect the learning ability of f_{pre}. As a result, in Table S7, these methods show significant accuracy drops on the remaining data $\mathcal{D}_{\text{rm}}$ and even slight accuracy increases on the forgetting data $\mathcal{D}_{\text{fgp}}$ in previous rounds, after 4 rounds of unlearning. In contrast, LOFT does not update parameters of f_{pre}, and well preserves the learning ability of f_{pre}. By only optimizing the projection matrix $\mathbf{U}$ given $\mathbf{\Sigma}^{\text{pre}}_{\text{rm}}$, $\mathbf{\Sigma}^{\text{pre}}_{\text{fg}}$ and $\mathbf{\Sigma}^{\text{pre}}_{\text{fgp}}$, the learned subspace of LOFT successfully maintains the accuracy on $\mathcal{D}_{\text{rm}}$ and reduces the accuracy on $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{fgp}}$.

Table S8: Training hyper-parameters of different MU methods w.r.t. Table S9 with ResNet50 on VGGFace2.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	hyper-parameters
f_{pre}	✓	✓	200 epochs, lr $= 10^{-1}$, cosine scheduler
f_{exact}	✓	✗	200 epochs, lr $= 10^{-1}$, cosine scheduler
GA	✗	✓	5 epochs, lr $= 5 \times 10^{-4}$, constant scheduler
GA	✓	✓	5 epochs, lr $= 5 \times 10^{-4}$, constant scheduler
RL	✗	✓	10 epochs, lr $= 5 \times 10^{-4}$, cosine scheduler
RL	✓	✓	10 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler
SalUn	✗	✓	10 epochs, lr $= 10^{-4}$, cosine scheduler, saliency sparsity 50%
SalUn	✓	✓	10 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	10 epochs, lr $= 2 \times 10^{-5}$, cosine scheduler
COUN+RL	✓	✓	15 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler
L2UL	✗	✓	lr $= 1 \times 10^{-4}$, constant scheduler, regularization coefficient = 1.0
DELETE	✗	✓	10 epochs, lr $= 5 \times 10^{-4}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	50 steps, $s = 21$, lr $= 1$, constant scheduler

Table S9: Comparison results of ResNet50 on VGGFace2, forgetting one of 200 identities.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathbf{Acc}_{\text{rm}}^{\text{tr}}$	$\mathbf{Acc}_{\text{fg}}^{\text{tr}}$	$\mathbf{Acc}_{\text{rm}}^{\text{te}}$	$\mathbf{Acc}_{\text{fg}}^{\text{te}}$	MIA	Avg.G.↓	RTE ↓	# param. (%)
f_{pre}	✓	✓	100.00	100.00	98.15	99.49	100.00	-	5970.95	25,557,032 (100%)
f_{exact}	✓	✗	100.00 (0.00)	0.00 (0.00)	98.21 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00	5870.35	25,557,032 (100%)
GA	✗	✓	91.91 (8.09)	0.00 (0.00)	86.34 (11.87)	0.00 (0.00)	0.00 (0.00)	4.00	3.35	25,557,032 (100%)
GA	✓	✓	100.00 (0.00)	0.25 (0.25)	97.96 (0.25)	2.03 (2.03)	0.00 (0.00)	0.51	147.99	25,557,032 (100%)
RL	✗	✓	99.64 (0.36)	11.25 (11.25)	95.27 (2.94)	13.20 (13.20)	0.00 (0.00)	5.55	5.59	25,557,032 (100%)
RL	✓	✓	99.11 (0.89)	0.00 (0.00)	94.66 (3.55)	0.00 (0.00)	0.00 (0.00)	0.89	303.81	25,557,032 (100%)
SalUn	✗	✓	99.68 (**0.32**)	5.50 (5.50)	95.59 (**2.62**)	5.08 (5.08)	0.00 (0.00)	2.70	9.99	12,778,516 (50%)
SalUn	✓	✓	99.96 (0.04)	0.00 (0.00)	97.00 (1.21)	0.00 (0.00)	0.00 (0.00)	0.25	312.71	12,778,516 (50%)
COUN+RL	✗	✓	99.36 (0.64)	2.75 (2.75)	94.80 (3.41)	1.52 (1.52)	0.00 (0.00)	1.66	8.11	25,557,032 (100%)
COUN+RL	✓	✓	99.35 (0.65)	0.00 (0.00)	91.34 (6.87)	0.00 (0.00)	0.00 (0.00)	1.50	956.15	25,557,032 (100%)
L2UL	✗	✓	98.09 (1.91)	2.75 (2.75)	92.88 (5.33)	3.55 (3.55)	0.00 (0.00)	2.71	176.69	25,557,032 (100%)
DELETE	✗	✓	99.50 (0.50)	5.50 (5.50)	95.15 (3.06)	5.58 (5.58)	0.00 (0.00)	2.93	5.82	25,557,032 (100%)
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	97.20 (2.80)	0.00 (**0.00**)	94.22 (3.99)	0.51 (**0.51**)	0.00 (**0.00**)	**1.46**	**0.45**	**43,008 (0.17%)**

G Applications: Face Recognition and Emotion Recognition

Both identities and emotions are key facial attributes closely related to user privacy. Exploring the unlearning of such information can benefit privacy protection. Thereby we investigate MU for this two real-world applications of face recognition and emotion recognition.

G.1 Face Recognition with Machine Unlearning

Setups. Experiments are conducted on a prevalent face identity recognition dataset VGGFace2 [62], including approximately 3.31 million images collected from 9,131 identities. Particularly, we adopt its publicly-available version[3], where all face images are aligned and cropped to $112 \times 112 \times 3$. Our experiments utilize a filtered set of 200 identities, each with over 500 images. Then, we randomly select 400 images for each identity, forming a training set containing 80,000 images, and the rest is adopted as the test set. One of the 200 identities (0.5%) is randomly selected as the forgetting data $\mathcal{D}_{\text{fg}}$, which is a challenging setting for class-centric unlearning. The adopted neural network model is ResNet50 [27]. All the compared methods use the SGD optimizer [61] with a weight decay of 5×10^{-4}, the momentum of 0.9, and a batch size of 128. More training details are in Table S8 with comparison results in Table S9.

Results. This unlearning setting, i.e., forgetting one of the 200 identities, implies a substantial size of $\mathcal{D}_{\text{rm}}$. Therefore, MU methods involving $\mathcal{D}_{\text{rm}}$ can easily achieve the ideal performance of the exact unlearning model of f_{exact}, as illustrated by GA (w/ $\mathcal{D}_{\text{rm}}$), RL (w/ $\mathcal{D}_{\text{rm}}$), SalUn (w/ $\mathcal{D}_{\text{rm}}$), and COUN+RL (w/ $\mathcal{D}_{\text{rm}}$) in Table S9. In contrast, it is challenging for MU methods by only accessing $\mathcal{D}_{\text{fg}}$ under this unlearning setting, as demonstrated by the relatively high accuracy on $\mathcal{D}_{\text{fg}}$ of GA (w/o $\mathcal{D}_{\text{rm}}$), RL (w/o $\mathcal{D}_{\text{rm}}$), SalUn (w/o $\mathcal{D}_{\text{rm}}$), COUN+RL (w/o $\mathcal{D}_{\text{rm}}$), L2UL and DELETE in Table S9. Our LOFT does not visit the original samples and learns to preserve and remove essential patterns

[3] https://www.kaggle.com/datasets/yakhyokhuja/vggface2-112x112

Table S10: Training hyper-parameters of different MU methods w.r.t. Table S11 with ResNet18 on RAF-DB.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	hyper-parameters
f_{pre}	✓	✓	200 epochs, lr $= 10^{-1}$, cosine scheduler
f_{exact}	✓	✗	200 epochs, lr $= 10^{-1}$, cosine scheduler
GA	✗	✓	10 epochs, lr $= 5 \times 10^{-4}$, constant scheduler
GA	✓	✓	10 epochs, lr $= 5 \times 10^{-3}$, constant scheduler
RL	✗	✓	20 epochs, lr $= 5 \times 10^{-4}$, cosine scheduler
RL	✓	✓	10 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler
SalUn	✗	✓	20 epochs, lr $= 10^{-4}$, cosine scheduler, saliency sparsity 50%
SalUn	✓	✓	10 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler, saliency sparsity 50%
COUN+RL	✗	✓	10 epochs, lr $= 2 \times 10^{-5}$, cosine scheduler
COUN+RL	✓	✓	15 epochs, lr $= 5 \times 10^{-2}$, cosine scheduler
L2UL	✗	✓	lr $= 2 \times 10^{-4}$, constant scheduler, regularization coefficient = 1.0
DELETE	✗	✓	20 epochs, lr $= 2 \times 10^{-4}$, cosine scheduler
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	50 steps, $s = 6$, lr $= 1$, constant scheduler

Table S11: Comparison results of ResNet18 on RAF-DB.

method	$\mathcal{D}_{\text{rm}}$	$\mathcal{D}_{\text{fg}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{tr}}$	$\mathbf{Acc}_{\mathbf{rm}}^{\mathbf{te}}$	$\mathbf{Acc}_{\mathbf{fg}}^{\mathbf{te}}$	**MIA**	Avg.G.
f_{pre}	✓	✓	100.00	100.00	85.10	56.76	100.00	-
f_{exact}	✓	✗	100.00 (0.00)	0.00 (0.00)	84.90 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00
GA	✗	✓	91.38 (8.62)	0.36 (0.36)	76.75 (8.15)	0.00 (0.00)	0.36 (0.36)	3.50
GA	✓	✓	98.39 (1.61)	0.00 (0.00)	82.00 (2.90)	0.00 (0.00)	0.00 (0.00)	0.90
RL	✗	✓	98.96 (1.04)	15.66 (15.66)	79.93 (4.97)	5.41 (5.41)	0.00 (0.00)	5.42
RL	✓	✓	99.92 (0.08)	0.00 (0.00)	84.10 (0.80)	0.00 (0.00)	0.00 (0.00)	0.18
SalUn	✗	✓	99.83 (0.17)	12.46 (12.46)	82.53 (2.37)	2.70 (2.70)	0.36 (0.36)	3.61
SalUn	✓	✓	99.98 (0.02)	0.00 (0.00)	84.57 (0.33)	0.00 (0.00)	0.00 (0.00)	0.07
COUN+RL	✗	✓	99.57 (0.43)	6.76 (6.76)	81.63 (3.27)	1.35 (1.35)	1.07 (1.07)	2.58
COUN+RL	✓	✓	92.04 (7.96)	0.00 (0.00)	83.20 (1.70)	0.00 (0.00)	0.00 (0.00)	1.93
L2UL	✗	✓	99.99 (0.01)	10.68 (10.68)	84.03 (0.87)	1.35 (1.35)	0.00 (0.00)	2.58
DELETE	✗	✓	99.72 (0.28)	9.25 (9.25)	82.20 (2.70)	2.70 (2.70)	1.42 (1.42)	3.27
LOFT	only $\mathbf{\Sigma}_{\text{rm}}^{\text{pre}}$	only $\mathbf{\Sigma}_{\text{fg}}^{\text{pre}}$	100.00 (**0.00**)	0.00 (**0.00**)	84.90 (**0.00**)	0.00 (**0.00**)	0.00 (**0.00**)	**0.00**

from the feature covariance matrices of $\mathcal{D}_{\text{rm}}$ and $\mathcal{D}_{\text{fg}}$ through low-dimensional projections, managing to simultaneously maintain high accuracy on $\mathcal{D}_{\text{rm}}$ and suppress accuracy on $\mathcal{D}_{\text{fg}}$.

G.2 Emotion Recognition with Machine Unlearning

Setups. Experiments are conducted on a popular emotion recognition dataset RAF-DB [63], RAF-DB contains 12,271 and 3,068 training and test images with a size of $100 \times 100 \times 3$, respectively, within 7 different emotions: "surprise", "fear", "disgust", "happiness", "sadness", "anger" and "neutral". In the experiments, one emotion is randomly selected as the forgetting data $\mathcal{D}_{\text{fg}}$ ("fear"). The adopted model is ResNet18 [27]. All the compared methods use the SGD optimizer [61] with a weight decay of 5×10^{-4}, the momentum of 0.9, and a batch size of 128. More training details are listed in Table S10. The comparison results among different MU methods are shown in Table S11.

Results. As illustrated in Table S11, when forgetting the "fear" emotion from the total 7 emotions, LOFT shows the best approximation performance towards that of the retrained model f_{exact}. Other MU methods either heavily rely on the remaining data $\mathcal{D}_{\text{rm}}$ (RL and SalUn), or show unsatisfactory results on $\mathcal{D}_{\text{fg}}$ (L2UL and DELETE). LOFT explores the low-dimensional subspace in which the features of $\mathcal{D}_{\text{fg}}$ and $\mathcal{D}_{\text{rm}}$ can be well distinguished through its reconstruction performances, achieving distinctively better unlearning performances.

H Discussions: Privacy Scope, Threat Model and Limitations

LOFT is *not* a certified unlearning method and does not provide formal privacy guarantees such as differential privacy or certified deletion. The privacy-related advantage of LOFT should instead be understood in a narrower sense: compared with mainstream approximate unlearning methods

that repeatedly visit raw forgetting and remaining samples and update the full pretrained model through multiple forward-backward passes, LOFT requires only one-shot feature extraction to form covariance matrices and subsequently optimizes a lightweight projection matrix without repeatedly revisiting raw data. Therefore LOFT primarily reduces the exposure surface of the unlearning pipeline and mitigates privacy risk associated with repeated raw-data access during unlearning.

Black-box adversaries. In common practical deployment settings, the attacker may only access model outputs through queries. Such a black-box threat model is not aware of the inserted projection matrix $\mathbf{U}$ in LOFT. Under this black-box setting, we empirically audit privacy leakage using Membership Inference Attacks (MIAs). Across our experiments in Section 5, LOFT attains a small gap to the retrained model under a classic MIA method [30], indicating that the learned feature projection can suppress information about the forgetting data in a way that is difficult to exploit through output-based attacks. In the following Section H, we provide additional evaluations against two advanced MIA methods, LiRA [31] and RMIA [32], demonstrating that our LOFT maintains its superior privacy protection capability.

White-box adversaries and deployment considerations. A stronger threat model considers white-box adversaries with access to the internal components of the unlearned model, including its architecture, parameters, and even gradients. For LOFT, we distinguish between different levels of exposure. Access to the projection matrix $\mathbf{U}$ alone does not immediately enable recovery of the forgotten information, since $\mathbf{U}$ specifies only a feature projection and does not directly contain the raw forgetting data or the pretrained model parameters. Exploiting $\mathbf{U}$ would generally require additional knowledge of the model architecture and the pretrained feature representations. If both $\mathbf{U}$ and the complete pretrained model are exposed, an adversary may simply remove the plug-in projection to recover the original model behavior. To mitigate this risk, the projector $\mathbf{U}\mathbf{U}^\top$ can be absorbed into the subsequent linear classifier before deployment, so that $\mathbf{U}$ is no longer released as an independent and removable module. This reduces the practical attack surface, although it does not provide a formal privacy guarantee against white-box adversaries. Accordingly, our empirical privacy analysis focuses on the commonly considered black-box MIAs, while rigorous evaluation and defense under stronger white-box threat models remain important directions for future work.

Additional Results against More MIAs

We further evaluate unlearned models using two stronger score-based black-box MIAs, LiRA [31] and RMIA [32], to assess whether the forgetting samples remain distinguishable under black-box threat models. LiRA [31] estimates the likelihood of membership from a model-centric perspective by training multiple "shadow-in" and "shadow-out" models and comparing the target model's confidence on a query sample against the score distributions of member and non-member cases induced by these shadow models. RMIA [32], in contrast, takes a data-side perspective and calibrates the membership signal using a reference population dataset, and infers membership by comparing the target model's response on a candidate sample with its responses on population samples.

Setups. Our setup follows Table 1, where 4 out of 200 classes in Tiny-ImageNet are randomly selected as $\mathcal{D}_{\text{fg}}$. For LiRA, we train 16 "shadow-in" models on the full training set and 16 "shadow-out" models on the remaining training set, each with different random seeds. For RMIA, we train a reference model on the remaining training set, and utilize the remaining test samples as the population set. We evaluate LiRA and RMIA by distinguishing the training forgetting samples from the test forgetting samples. Figure S3 reports the ROC curves and AUC values of different unlearning methods under LiRA and RMIA. For fairness, GA, RL, SalUn, and COUN+RL are all optimized with access to the training remaining data.

Results. As shown in Figure S3, LOFT achieves AUC values close to 0.5 under both LiRA and RMIA, indicating that these stronger black-box MIAs can hardly distinguish the forgetting samples from non-members after unlearning. This result suggests that LOFT effectively suppresses exploitable membership signals on the forgetting data. By exploiting the separability between the remaining and forgetting data in the feature subspace, LOFT removes feature components that are particularly informative for the forgetting set, thereby making membership inference substantially more difficult in a black-box setting.

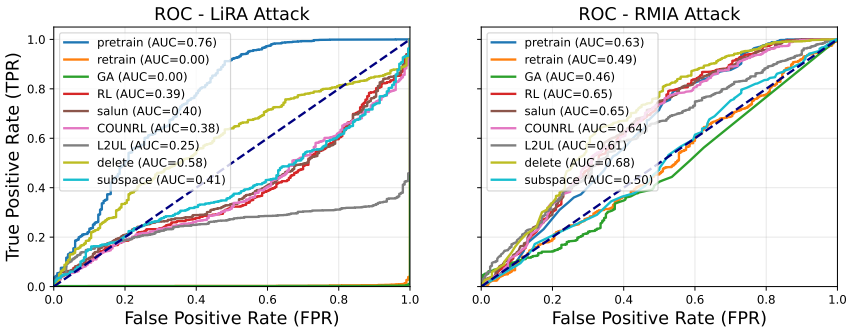


Figure S3: Results against two advanced MIAs, LiRA and RMIA. Empirical setups follow the same as that of Table 1.

Additionally, under LiRA, the retrained model achieves near-zero AUROC, potentially because retraining completely removes the forget-set classes, leaving the model unable to discriminate those samples. Consequently, its predictive confidence for forgotten members is uniformly and substantially lower than for non-members.

I Low-Dimensional Feature Subspaces for MU in Generation Tasks

In this section, we provide a preliminary exploration of using low-dimensional feature subspaces for machine unlearning in generation tasks. As our LOFT does not update the parameters of the pretrained model f_{pre}, LOFT is not directly applicable to models in generation tasks. Meanwhile, there are distinctively different generation models in different tasks. For example, in image generation, the convolution network in DDPM [64] and LDM [65] significantly differs from the vision transformer in DiT [66], while the transformer itself functions differently between image generation and language-related generation tasks [55]. Another challenge is that generative models are particularly sensitive to internal changes in their feature representations. For instance, even subtle alterations in the intermediate layers of a DDPM can lead to artifacts or meaningless pixels in the generated images. Therefore, crafting an appropriate low-dimensional feature subspace must be carefully tailored to the specific generative model and the particular unlearning scenario. This direction, to the best of our knowledge, still remains underexplored. In the following, we present promising unlearning results from LOFT applied to a *machine translation* task. This initial success suggests the potential of our feature subspace perspective for MU in generative tasks, and we hope it will advocate further research in this area.

Setups. We evaluate LOFT on a machine translation task with an unlearning target as *forcing the model to forget the ability of translating a specific language*. This natural language processing task is not a classification one and the subspace shall not be tightly coupled with class-specific features. The experiment is conducted on the WMT-19 dataset[4] and a modern large model named `m2m100_1.2B`[5]. The pretrained `m2m100_1.2B` model is fine-tuned in a parameter-efficient way (i.e., LoRA) on two language pairs from WMT-19: translating Czech into English and translating Lithuanian into English. This fine-tuned `m2m100_1.2B` model is adopted as the initial startpoint to be unlearned, and the to-be-unlearned language is Lithuanian.

Implementation. With this modern large model of `m2m100_1.2B`, we apply LOFT to the *hidden states* from the output of the encoder of `m2m100_1.2B`. Specifically, the token-level hidden states get weighted averaged by the self-attention from the encoder and lead to weighted averaged 1,024-dimensional features, where 1,024 is the hidden dimension of `m2m100_1.2B`. In this way, we avoid the massive tokens and extremely large vocabulary size and focus on the attention-based weighted average features to apply LOFT. Then, two feature covariance matrices are calculated w.r.t. Czech and Lithuanian inputs, and the optimization objective of LOFT in (1) is conducted to learn a projection

[4]https://huggingface.co/datasets/wmt/wmt19
[5]https://huggingface.co/facebook/m2m100_1.2B

matrix. In inference, the projector is employed at the token-level hidden states to proceed the propagation, similar as the way in Section 5.3.

Results. Our results are presented in the following Table S12. We evaluate translation quality using the BLEU score [67] on the WMT-19 validation set for Czech-to-English (cs→en) and Lithuanian-to-English (lt→en) directions. The forgetting data (fg) and remaining (rm) data correspond to the lt→en and cs→en pairs, respectively. Here, fully-trained denotes `m2m100_1.2B` fine-tuned on both language pairs, while retrained refers to `m2m100_1.2B` trained exclusively on the remaining (cs→en) data without seeing (lt→en) inputs. Blue numbers in parentheses denote the BLEU score gap relative to the retrained method (lower is better). After unlearning, our LOFT achieves performance on the forgetting language pair (lt→en) that is very close to the retrained model, implying successful forgetting of the Lithuanian-to-English translation capability. Meanwhile, LOFT maintains good performance on the remaining Czech-to-English direction, demonstrating effective knowledge preservation. As an exemplary trial on large language models, this experiment validates LOFT's potential for extension to non-classification tasks and its scalability to modern large models.

Table S12: Results of LOFT on machine translation with a modern large model of `m2m100_1.2B`.

method	cs→en (rm)	lt→en (fg)
fully-trained	27.63	17.54
retrained	26.19	12.31
GA	23.51 (2.68)	15.79 (3.48)
LOFT	22.88 (3.31)	13.04 (0.73)

Impact Statement

This paper presents work aimed at advancing the privacy protection of modern deep learning models by integrating classic machine learning techniques. This advancement has the potential to benefit a wide range of fields and societal applications that is related to user privacy protection. By releasing our code along with this work, we aim to provide the machine unlearning research community with a fresh methodological perspective and an interpretable and practical plug-in module that can be optimized with superior effectiveness and efficiency. While we do not foresee direct negative societal impacts arising from this work, we intend to further refine and extend the method in future research.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are discussed in the last paragraph of the main text.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The full set of assumptions and a complete and correct proof are provided in the appendices.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Detail experimental settings are provided in the appendices and the code to reproduce the main experimental results is released.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code to faithfully reproduce the main experimental results is publicly released.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details necessary to understand the results are specified in appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The mean and standard deviations in main results are reported from repeated experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information of the computer resources is provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Societal impacts of this work are discussed in the end of the appendices.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Assets used are properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code is well documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.