IMU: Influence-guided Machine Unlearning

Xindi Fan[1], Jing Wu[2], Mingyi Zhou[3], Pengwei Liang[1], Dinh Phung[2]

[1] Harbin Institute of Technology [2] Data Science & AI, Monash University [3] School of Software, Beihang University
xindi.fan@stu.hit.edu.cn, {jing.wu1,dinh.phung}@monash.edu, zhoumingyi@buaa.edu.cn.

Abstract

Recent studies have shown that deep learning models are vulnerable to attacks and tend to memorize training data points, raising significant concerns about privacy leakage. This motivates the development of machine unlearning (MU), *i.e.*, a paradigm that enables models to selectively forget specific data points upon request. However, most existing MU algorithms require partial or full fine-tuning on the retain set. This necessitates continued access to the original training data, which is often impractical due to privacy concerns and storage constraints. A few retain-data-free MU methods have been proposed, but some rely on access to auxiliary data and precomputed statistics of the retain set, while others scale poorly when forgetting larger portions of data. In this paper, we propose Influence-guided Machine Unlearning (IMU), a simple yet effective method that conducts MU using only the forget set. Specifically, IMU employs gradient ascent and innovatively introduces dynamic allocation of unlearning intensities across different data points based on their influences. This adaptive strategy significantly enhances unlearning effectiveness while maintaining model utility. Results across vision and language tasks demonstrate that IMU consistently outperforms existing retain-data-free MU methods.

1 Introduction

Deep learning (DL) models are widely deployed in various applications, but their vulnerability to adversarial attacks, such as membership inference attacks (Rezaei and Liu 2021; Watson et al. 2022) and model inversion attacks (Zhu, Liu, and Han 2019; Geiping et al. 2020; Balunović et al. 2022), raises significant concerns about privacy leakage. In response, legislation such as GDPR grants users the *right to be forgotten*, compelling models to remove data points upon request. The straightforward method is to retrain the model without using the forgetting data from scratch. However, this is highly inefficient, particularly for large-scale models trained on massive datasets. As a result, machine unlearning (MU), *i.e.*, removing the influence of specific data points from a well-trained model while preserving the model utility on unrelated information without the need for costly and prolonged retraining, has emerged as a crucial approach for safeguarding data privacy.

Current MU methods typically require fine-tuning on the retain data, which are part of the original training data. However, accessing the original training data is often impractical in real-world deployment. Recently, a few studies have sought to overcome this limitation by developing retain-data-free MU methods (Cha et al. 2024; Bonato, Cotogni, and Sabetta 2024; Foster et al. 2025). Among these efforts, Cha et al. (2024) presents instance-wise unlearning that generates adversarial examples w.r.t. the forgetting data and fine-tunes over these examples, specifically targeting model parameters responsible for the correct classification of the forgetting data points. Meanwhile, Bonato, Cotogni, and Sabetta (2024) introduces selective-distillation for class and architecture-agnostic Learning (SCAR), which uses the Mahalanobis distance to shift the feature vectors w.r.t. the forgetting data toward the nearest wrong class distribution, then distills the knowledge from the original model into the scrubbed model using out-of-distribution images.

While these retain-data-free methods have shown promising results in deleting data from DL models, they exhibit important limitations. Cha et al. (2024) only consider evaluation when the forgetting data is randomly selected from the training data (*e.g.*, around 0.5% on CIFAR-10), raising concerns about its scalability to larger forgetting sets. SCAR needs an auxiliary dataset to maintain model performance and assumes that the statistics (*e.g.*, mean, and covariance) of the retain set are stored, which may not always be available in practical scenarios. **These constraints highlight the need for a more robust and generalizable retain-data-free unlearning framework that can handle diverse forgetting scenarios without external data requirements.**

Influence function (Koh and Liang 2017), which estimates parameter changes induced by data point removal without full retraining, has attracted substantial attention and research interest in recent years (Basu, Pope, and Feizi 2021; Bae et al. 2022; Chhabra et al. 2024). While influence-based methods have been adapted for MU (Sekhari et al. 2021; Neel, Roth, and Sharifi-Malvajerdi 2021; Mehta et al. 2022; Wu, Hashemi, and Srinivasa 2022), two critical challenges persist: (1) Influence function approximations are known to be fragile in deep learning (Basu, Pope, and Feizi 2021). The influence estimation is fairly accurate for shallow networks where the loss function is convex, but prone to error in deep models due to the non-convexity of their loss landscapes; (2) for large neural networks with complex architectures and millions of parameters, computing inverse

Preprint.

Hessian-vector products, even with the Fisher Information approximation to estimate the influence function, remains prohibitively expensive (Mehta et al. 2022).

In this work, we argue that the full potential of the influence function on MU remains underexplored. While conventional influence-based unlearning (IU) methods primarily focus on computational efficiency (Mehta et al. 2022), they overlook two critical aspects: (1) the stability of influence estimates in deep non-convex networks, and (2) the heterogeneous influence distribution across forgetting data points, where certain samples exert disproportionately stronger effects on the model than others. Therefore, we propose a novel Influence-guided Machine Unlearning (*IMU*) method to dynamically allocate different attention to the forgetting data points based on their influence value. We denote the final fully connected layer as the classifier and the preceding layers as the feature extractor. To improve the accuracy of the estimation of influence value and avoid expensive matrix inversion computations, we view the feature representations extracted by the feature extractor as the input, and then estimate the influence value at the classifier. In this way, the classifier is convex, and the influence estimates would be more reliable. Experimental results demonstrate that our proposed algorithm effectively removes the influence of forgetting data while preserving model utility, showing both efficiency and stability in the MU process.

Our main contributions are summarized as:

- We propose a retain-data-free MU method, *IMU*, which leverages influence functions on non-convex models by estimating the influence at the final fully connected layer. This design avoids full-model Hessian inversion while maintaining estimation accuracy.
- *IMU* automatically adjusts the unlearning strength for each forgetting data point proportionally to its influence score, applying more aggressive parameter updates for high-influence samples while preserving knowledge from less influential ones.
- Extensive experiments across vision and language tasks show that our proposed method *IMU* consistently surpasses state-of-the-art retain-data-free MU methods in both forget quality and model utility.

2 Related Work

2.1 Machine Unlearning

Current MU algorithms typically fall into two categories: exact unlearning methods (Ginart et al. 2019; Cao and Yang 2015; Romero, Barrio, and Belanche 2007; Karasuyama and Takeuchi 2010) that use data partitioning strategies (Bourtoule et al. 2021) or only focus on traditional machine learning models like k-means clustering (Ginart et al. 2019), and approximate unlearning methods (Golatkar, Achille, and Soatto 2020a; Golatkar et al. 2021; Golatkar, Achille, and Soatto 2020b; Chen et al. 2023; Heng and Soh 2023; Kumari et al. 2023; Kurmanji et al. 2023; Foster, Schoepf, and Brintrup 2024; Lyu et al. 2024; Bui et al. 2024; Ko et al. 2024; Lin et al. 2024; Spartalis et al. 2025; Alberti et al. 2025) that trade perfect deletion for computational efficiency. So far, these MU methods have demonstrated effectiveness through two key metrics: (1) degraded performance on the forgetting data, and (2) maintained utility on retain and unseen set.

Gradient ascent (GA)-based MU methods (Wu, Dobriban, and Davidson 2020; Gandikota et al. 2023; Jang et al. 2022) perform unlearning by moving model parameters away from the forgetting data through gradient ascent. While simple to implement, these methods can be sensitive to hyperparameters and may interfere with retain knowledge (Jia et al. 2023; Fan et al. 2024a). Fisher unlearning methods (Golatkar, Achille, and Soatto 2020a,b; Golatkar et al. 2021) assume that the unlearned model and the retrained model are close to each other in parameter space, and formulate Fisher Forgetting, which induces unlearning by injecting noise into the parameters proportional to their relative importance in the forgetting data compared to the retain set. Influence function-based methods estimate the impact of individual data points to perform approximate unlearning. Guo et al. (2020) propose a one-step Newton-update procedure based on the influence function to excise the effect of specific data points from a pretrained model. Subsequent works (Mehta et al. 2022; Liu et al. 2022) focus on ways for more efficient computation over the estimation. However, these methods make strong convexity assumptions and are still computationally expensive for large-scale models (Mehta et al. 2022). Two-stage methods (Jia et al. 2023; Fan et al. 2024b; Wu and Harandi 2024) have emerged as a practical alternative, first erasing information related to the forgetting data, then fine-tuning on retain data. These methods avoid expensive computations while achieving strong empirical results, particularly when combined with sparsity techniques.

These MU methods, however, still require access to the retain set, presenting practical challenges when storage constraints or privacy concerns limit data availability. Recent work has begun addressing this limitation through alternative approaches (Cha et al. 2024; Bonato, Cotogni, and Sabetta 2024; Foster et al. 2025). Cha et al. (2024) proposes an instance-wise unlearning method that generates adversarial examples w.r.t. the forgetting data and fine-tunes the model on these examples, specifically targeting parameters responsible for the correct classification of the forgetting data. Meanwhile, Bonato, Cotogni, and Sabetta (2024) introduces SCAR, which leverages the Mahalanobis distance to shift feature representations of forgetting data toward the nearest incorrect class distribution. The scrubbed model then undergoes knowledge distillation using out-of-distribution images to preserve model utility. NPO (Zhang et al. 2024) treats the forgetting data as negative examples in DPO (Rafailov et al. 2023), presenting an adaptive control in large language model (LLM) unlearning. Fan et al. (2024a) further improve NPO via a reference-free method SimNPO (Meng, Xia, and Chen 2024), which addresses the reference model bias issue.

These advances are particularly valuable for real-world deployment, where retaining full training data is often impractical. *However, existing retain-data-free MU methods exhibit notable limitations. Cha et al. (2024) evaluates only on small, randomly selected forgetting sets, raising concerns about scalability to larger deletions. Bonato, Cotogni, and Sabetta (2024) relies on an auxiliary dataset to preserve*

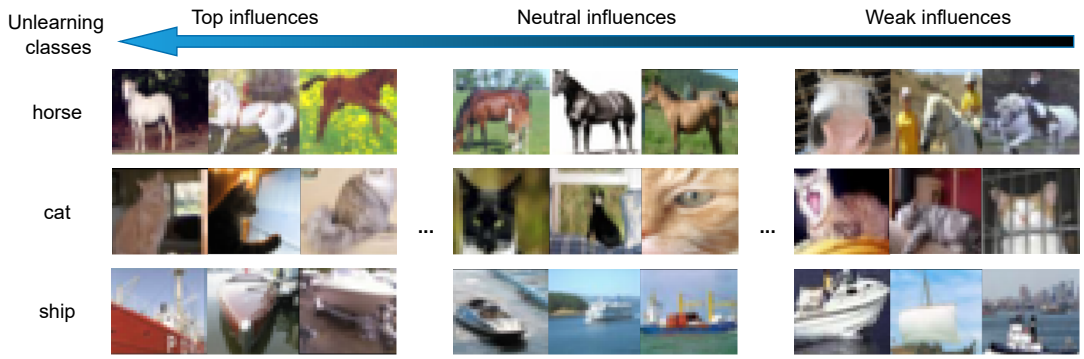

Figure 1: Examples of data points with different degrees of influence on the data set that belong to their own class.

*model utility, and requires that the statistics (*e.g.*, mean and covariance) of retain data are stored, a requirement that may not always be feasible. These constraints underscore the need for a more robust and generalizable retain-free unlearning framework capable of handling diverse forgetting scenarios without external data dependencies.*

2.2 Influence Function

Influence functions (Hampel 1974; Koh and Liang 2017; Pruthi et al. 2020) assess how infinitesimal changes to training data weights affect performance on a validation set, by analyzing their impact on a target evaluation metric.

For linear models, influence functions are well-defined thanks to the convexity of the loss function, but in deep learning, where loss functions are typically non-convex, the behavior of influence functions remains poorly understood (Basu, Pope, and Feizi 2021). Basu, Pope, and Feizi (2021) presents a comprehensive empirical study on the reliability of influence functions in DNNs. The findings reveal that factors such as network architecture, depth, width, parameterization, and regularization significantly impact influence estimation accuracy. Overall, the results demonstrate that influence functions in DNNs frequently fail to accurately predict the effects of retraining, leading to the conclusion that influence estimates are often brittle and unreliable.

However, recent works Bae et al. (2022); Epifano et al. (2023) have challenged these claims, making their applicability in DNNs an ongoing topic of debate. So far, several strategies have been proposed to extend influence functions to non-convex models: (1) replacing non-convex embeddings with those from linear models (Li and Liu 2022; Chhabra et al. 2024); (2) adding damping terms to make the Hessian matrix positive definite (Koh and Liang 2017; Han, Wallace, and Tsvetkov 2020); and (3) deriving task-specific or second-order influence functions (Basu, You, and Feizi 2020; Alaa and Van Der Schaar 2020).

Existing influence function-based MU methods mainly focus on computational efficiency (Mehta et al. 2022; Jia et al. 2023), they overlook two key challenges: (1) the instability of influence estimates in deep, non-convex models, and (2) the heterogeneous influence distribution across the forgetting data points, where certain samples have a disproportionately large impact on the model. In response, we aim to enhance influence-based unlearning by addressing these challenges through a more stable and adaptive influence estimation framework.

3 Methodology

In this section, we propose *IMU*, a retain-data-free unlearning framework that scrubs data from a model by dynamically eliminating its influence from the model. Throughout the paper, we denote scalars and vectors/matrices by lowercase and bold symbols, respectively (*e.g.*, a, $\boldsymbol{a}$, and $\boldsymbol{A}$).

3.1 Preliminaries

Notation. Let $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ be a model with parameters $\boldsymbol{\theta}$, classifying inputs $\boldsymbol{x} \in \mathcal{X}$ to labels $\boldsymbol{y}$ in the label space $\mathcal{Y}$. Let $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ denote our full training set. We partition $\mathcal{D}_{\text{train}}$ into two disjoint subsets: the forgetting data $\mathcal{D}_f \subset \mathcal{D}_{\text{train}}$, whose samples are required to be removed from the model, and the retain data $\mathcal{D}_r = \mathcal{D}_{\text{train}} \setminus \mathcal{D}_f$, whose knowledge we wish to preserve. We decompose $f_{\boldsymbol{\theta}}$ into two modules: a feature extractor $\phi_{\boldsymbol{\theta}_e} : \mathbb{R}^n \to \mathbb{R}^d$, comprising all layers up to but excluding the final classification stage, and a classifier $h_{\boldsymbol{\theta}_c} : \mathbb{R}^d \to \mathbb{R}^C$, corresponding to the last fully connected layer. Our goal is to adjust $f_{\boldsymbol{\theta}}$ so that it effectively erases the information associated with $\mathcal{D}_f$ while maintaining performance on $\mathcal{D}_r$ and the unseen set $\mathcal{D}_t$.

Influence Function. Refer to (Koh and Liang 2017), denote the empirical risk minimizer of the loss over the training set as $\boldsymbol{\theta}^* := \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^N \ell(\boldsymbol{x}_i, \boldsymbol{y}_i; \boldsymbol{\theta})$, the change in model parameters caused by an infinitesimal up-weighting of the training data point $\boldsymbol{x}$ can be approximated by

$$\mathcal{I}(\boldsymbol{x}) := -\boldsymbol{H}_{\boldsymbol{\theta}^*}^{-1} \nabla_{\boldsymbol{\theta}} \ell\big(f_{\boldsymbol{\theta}^*}(\boldsymbol{x})\big), \tag{1}$$

where $\boldsymbol{H}_{\boldsymbol{\theta}^*} = \frac{1}{N} \sum_{i=1}^N \nabla_{\boldsymbol{\theta}}^2 \ell\big(f_{\boldsymbol{\theta}^*}(\boldsymbol{x}_i)\big)$. Then, the change in the loss value for $\mathcal{D}_t$ when the training data point $\boldsymbol{x}$ is removed from the training set can be approximated by

$$\mathcal{I}(\boldsymbol{x}, \mathcal{D}_t) := -\nabla_{\boldsymbol{\theta}} \ell\big(f_{\boldsymbol{\theta}^*}(\mathcal{D}_t)\big)^\top \boldsymbol{H}_{\boldsymbol{\theta}^*}^{-1} \nabla_{\boldsymbol{\theta}} \ell\big(f_{\boldsymbol{\theta}^*}(\boldsymbol{x})\big). \tag{2}$$

Proofs can be found in (Koh and Liang 2017). However, the derivation relies on the assumption that the underlying loss function is strictly convex w.r.t. the model parameters and that the Hessian matrix is positive definite. Besides, for deep neural networks, computing the exact inverse of the Hessian matrix is computationally expensive (Mehta et al. 2022).

3.2 *IMU*

Our proposed method, *IMU*, addresses two key challenges in applying influence functions to deep neural networks: (1) the instability of influence estimates in highly non-convex networks, and (2) the heterogeneous influence distribution among forgetting samples, where certain data points have disproportionately large effects on the model.

Influence estimation. Bengio, Courville, and Vincent (2013) suggests that high-dimensional data tends to lie on lower-dimensional manifolds. Hence, rather than estimating influence on the full deep network, which would require inverting a high-dimensional, potentially ill-conditioned Hessian, we instead compute influence at the level of learned representations. Specifically, we treat the feature representation $\boldsymbol{z} = \phi(\boldsymbol{x})$ as the input and estimate the influence of each data point $\boldsymbol{x}$ w.r.t. the classifier $h(\cdot)$. Therefore, we estimate the influence value w.r.t. the data point $\boldsymbol{x} \in \mathcal{D}_f$ as

$$\mathcal{I}^c(\boldsymbol{x}) = -\boldsymbol{H}_{\boldsymbol{\theta}_c^*}^{-1} \nabla_{\boldsymbol{\theta}_c} \ell\big(h_{\boldsymbol{\theta}_c^*}(\boldsymbol{z})\big), \tag{3}$$

where the latent representation $\boldsymbol{z} = \phi_{\boldsymbol{\theta}_e^*}(\boldsymbol{x})$. And the change in the loss value for $\mathcal{D}_f$ when the training data point $\boldsymbol{x}$ is removed from the training set can then be approximated by

$$\mathcal{I}^c(\boldsymbol{x}, \mathcal{D}_f) = -\nabla_{\boldsymbol{\theta}_c} \ell\big(h_{\boldsymbol{\theta}_c^*}(\mathbf{Z}_f)\big)^\top \boldsymbol{H}_{\boldsymbol{\theta}_c^*}^{-1} \nabla_{\boldsymbol{\theta}_c} \ell\big(h_{\boldsymbol{\theta}_c^*}(\boldsymbol{z})\big), \tag{4}$$

where $\mathbf{Z}_f = \phi_{\boldsymbol{\theta}_e^*}(\mathcal{D}_f)$. This can significantly improve the stability and accuracy of influence estimation and avoid the computational cost of full Hessian inversion. Figure 1 presents examples with different levels of influence on the data from their respective class. Top influences are examples whose omission from the training data is predicted to most increase the loss on the corresponding samples. In Figure 2, we show that equally treating these examples (denoted as GA) in unlearning potentially causes either under-unlearning or over-unlearning. Hence, in the following, we introduce the influence-guided loss to dynamically allocate attention for different data points.

Influence-guided MU. To realize dynamically allocating different degrees of attention to data points according to their influence values, we integrate their corresponding influence values into the MU process by defining an influence-guided loss as

$$\mathcal{L}_{\texttt{IMU}}(\mathcal{D}_f; \boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{D}_f} \left[\hat{\mathcal{I}}^c(\boldsymbol{x}_i, \mathcal{D}_f) \cdot \ell\big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\big) \right], \tag{5}$$

where $\hat{\mathcal{I}}^c(\boldsymbol{x}_i, \mathcal{D}_f) = \left| \sqrt{\mathcal{I}^c(\boldsymbol{x}_i, \mathcal{D}_f)} \right| / \sum_j \left| \sqrt{\mathcal{I}^c(\boldsymbol{x}_j, \mathcal{D}_f)} \right|$ denotes the normalized score. Different from the naive gradient ascent, Equation (5) is equivalent to taking a weighted average of the per-sample losses, with weights proportional depending on their influence function value. Whereas, in practice, due to the large variance of the influence value distribution of different samples and the error of approximate estimation, the influence value of a certain sample can be very large. Such extreme value can dominate the weighted average, effectively diminishing the relative contribution of others that are equally important, leading the model to overlook them. To tackle this problem, we try to smooth and truncate after computing $\boldsymbol{I} = [\mathcal{I}^c(\boldsymbol{x}_1, \mathcal{D}_f), \cdots, \mathcal{I}^c(\boldsymbol{x}_m, \mathcal{D}_f)]^\top$. If $\mathcal{I}^c(\boldsymbol{x}_i, \mathcal{D}_f) < 0$, it indicates that this sample is helpful in improving the model's performance on $\mathcal{D}_f$. Therefore, we only need to select these samples with a negative impact to participate in unlearning. Algorithm 1 describes the procedure of our algorithm *IMU* in detail. We use these influence scores to dynamically adjust the unlearning strategy, assigning sample-specific attention during the parameter update process. Samples with higher influence values receive more aggressive updates, enabling targeted and efficient forgetting while preserving the performance on retain and unseen data.

Algorithm 1: Influence-guided Machine Unlearning (*IMU*).

Input: Model f parameterized by $\boldsymbol{\theta}$, consist of a feature extractor ϕ with parameters $\boldsymbol{\theta}_e$ and a classifier h parameterized by $\boldsymbol{\theta}_c$, forgetting data $\mathcal{D}_f$.

Output: Parameters $\boldsymbol{\theta}^*$ for the scrubbed model.

$\boldsymbol{\theta}^0 = \boldsymbol{\theta}$, learning rate η, number of iterations T.

for iteration t in T **do**

  Compute $\boldsymbol{H}_{\boldsymbol{\theta}_c} = \mathbb{E}_{\mathcal{D}_f}\left[\nabla^2_{\boldsymbol{\theta}_c} \ell\left(h_{\boldsymbol{\theta}_c}(\boldsymbol{z}_i)\right)\right]$ and $\boldsymbol{Z}_f = \phi_{\boldsymbol{\theta}_e}(\mathcal{D}_f)$ where $\boldsymbol{z}_i = \phi_{\boldsymbol{\theta}_e}(\boldsymbol{x}_i), \forall \boldsymbol{x}_i \sim \mathcal{D}_f$.

  $\forall \boldsymbol{x} \sim \mathcal{D}_f$, estimate influence value $\mathcal{I}^c(\boldsymbol{x}, \mathcal{D}_f)$ via Eq. (4): $-\nabla_{\boldsymbol{\theta}_c} \ell\big(h_{\boldsymbol{\theta}_c}(\mathbf{Z_f})\big)^\top \boldsymbol{H}^{-1}_{\boldsymbol{\theta}_c} \nabla_{\boldsymbol{\theta}_c} \ell\big(h_{\boldsymbol{\theta}_c}(\boldsymbol{z})\big)$.

  Select the samples satisfying $\mathcal{I}^c(\boldsymbol{x}_i, \mathcal{D}_f) < 0$ and process unlearning with these samples $\mathcal{D}_f^{\mathcal{I}^c<0}$.

  Compute the influence-guided loss $\mathcal{L}_{\texttt{IMU}}(\mathcal{D}_f; \boldsymbol{\theta}^{(t)})$ via Eq. (5): $-\mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{D}_f^{\mathcal{I}^c<0}} \left[\hat{\mathcal{I}}^c(\boldsymbol{x}_i, \mathcal{D}_f) \cdot \ell\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\right]$.

  Updating: $\boldsymbol{\theta}_c^{(t+1)} = \boldsymbol{\theta}_c^{(t)} - \eta \nabla_{\boldsymbol{\theta}_c^t} \mathcal{L}_{\texttt{IMU}}(\mathcal{D}_f; \boldsymbol{\theta}^{(t)})$.

end for

return $\boldsymbol{\theta}^{(T)}$

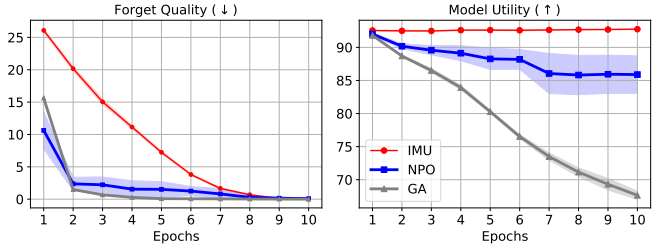

Figure 2: Comparison between GA, NPO, and *IMU*.

3.3 Connection with NPO

Negative Preference Optimization (NPO) (Zhang et al. 2024) stands out in LLM unlearning, it defines

$$\mathcal{L}_{\texttt{NPO}}(\boldsymbol{\theta}) \coloneqq \frac{2}{\beta} \mathbb{E}_{\mathcal{D}_f} \left[\log \left(1 + \left(\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})}{\pi_{\texttt{ref}}(\boldsymbol{y}|\boldsymbol{x})} \right)^{\beta} \right) \right], \tag{6}$$

where $\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ denotes the output probability of the current model, and $\pi_{\texttt{ref}}(\boldsymbol{y}|\boldsymbol{x})$ is the output probability from a reference model (*e.g.*, the original model). For image classification, the model typically outputs a softmax probability distribution over classes as $\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \texttt{softmax}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$, where $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ are the logits, and $\ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = -\log \pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$.

Refer to Proposition 1 in NPO (Zhang et al. 2024), NPO is a strict generalization of the gradient ascent (GA) method. The gradients of GA, NPO, and *IMU* are

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\texttt{GA}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f}\left[\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})\right], \quad (7)$$

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\texttt{NPO}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f}\left[\mathcal{W}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y})\cdot\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})\right], \quad (8)$$

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\texttt{IMU}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f}\left[\hat{\mathcal{I}}^c(\boldsymbol{x},\mathcal{D}_f)\cdot\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})\right], \quad (9)$$

where $\mathcal{W}_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{y}) = 2\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})^{\beta}/\left(\pi_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})^{\beta}+\pi_{\texttt{ref}}(\boldsymbol{y}|\boldsymbol{x})^{\beta}\right)$ is the adaptive smoothing weight, controlling the divergence speed in unlearning. NPO adaptively assigns weights to data points to help prevent catastrophic collapse in GA; however, Fan et al. (2024a) shows that NPO exhibits reference model bias, causing an uneven allocation of unlearning power. *IMU* instead uses explicitly estimated influence scores to guide the unlearning process. In this sense, *IMU* can be viewed as an influence-weighted counterpart to GA, offering a more direct and interpretable mechanism for prioritizing which data points to forget. Figure 2 presents the class-wise forgetting on CIFAR-10. We can observe that the forget quality of NPO decreases to 2.37% quickly and then stabilizes in later epochs, but it is accompanied by a continuous and rapid decline in model utility, from 92.01% to 85.89%. In contrast, the forget accuracy of *IMU* has decreased more slowly and has consistently maintained a high test accuracy around 92.60%. Further comparisons in language tasks can be found in § 4, and divergence analysis can be found in the appendix.

4 Experiment

We empirically evaluate *IMU* to answer the following research questions (RQs): **RQ1** How well can *IMU* balance forget quality and model utility across various unlearning scenarios? **RQ2** How well does *IMU* generalize across different tasks and datasets? **RQ3** How efficient is *IMU*?

4.1 Setup

Dataset and models. We evaluate MU methods on image classification using CIFAR-10 and CIFAR-100 (Krizhevsky, Hinton et al. 2009) datasets with the model ResNet-18 (He et al. 2016). Person re-identification, which matches a person's identity across different cameras or locations in a video or image sequence, is also included. ResNet-50 with a fully-connected layer is trained on the Market-1501 dataset (Zheng et al. 2015), which contains labeled person IDs captured under varying camera views. We also assess unlearning in a sequence modeling setting using a GPT-2 model on synthetic data (Fan et al. 2024a). For large language model unlearning, we employ the Llama-3.2-3B-Instruct model on the TOFU benchmark (Dorna et al. 2025).
Scenarios. We benchmark across five different unlearning settings: (i) Class-wise unlearning: forget a full category from CIFAR-10. (ii) Subclass-wise unlearning: forget a subclass within a superclass (*e.g.*, boy from people) in CIFAR-100. (iii) Sample-wise unlearning (see appendix): randomly select samples from $\mathcal{D}_{\texttt{train}}$ as the forget set. (iv) Person re-id unlearning: forget a specific identity from the 751 IDs in the Market-1501 train set, and evaluate model generalization on the query and gallery sets. (v) Distributional unlearning in sequence modeling: forget two Markov sub-distributions in a synthetic task. (vi) LLM unlearning (see appendix): perform the forget05 task on the TOFU benchmark.
Baselines. (1) Retrain, (2) Gradient Ascent (GA) (Thudi et al. 2022), (3) Random Label (RL) (Jia et al. 2023), (4) Influence Unlearning (IU*) (Jia et al. 2023), (5) SCAR* (Bonato, Cotogni, and Sabetta 2024), without the information about $\mathcal{D}_r$, (6) SSD (Foster, Schoepf, and Brintrup 2024), (7) NPO (Zhang et al. 2024), (8) SimNPO (Fan et al. 2024a).
Metrics. (1) Accuracy on $\mathcal{D}_f$, $\mathcal{D}_r$, and $\mathcal{D}_t$, denoted as $\texttt{Acc}_{\mathcal{D}_f}$, $\texttt{Acc}_{\mathcal{D}_r}$, and $\texttt{Acc}_{\mathcal{D}_t}$, respectively. (2) Membership inference attacks (*MIA*) aim to infer information about the training data. (3) W_{dist} (Tarun et al. 2023), *i.e.*, Wasserstein-1 distance, by measuring the similarity between the output distributions of the unlearned model and the retrained model on $\mathcal{D}_r$. (4) *mAP* (mean average precision) represents the average of the area under the precision-recall curve for each query, reflecting the overall retrieval quality.

4.2 Results on class-wise unlearning

We first evaluate on CIFAR-10, trying to forget an entire class category. As shown in Table 1, our proposed method *IMU* achieves superior unlearning performance. IU* fails to effectively preserve the model utility, showing significant drops of $\sim$15% in accuracy in both $\mathcal{D}_r$ and the test set $\mathcal{D}_t$. GA and RL perform even worse in this regard. The SOTA retain-data-free MU method SCAR* demonstrates strong forgetting ability; however, it comes at the cost of notable degradation in model utility, with accuracies on $\mathcal{D}_r$ and $\mathcal{D}_t$ reduced by approximately 14%. We further employ the potent LLM unlearning method, NPO, for image classification. It significantly outperforms baselines in both the forget quality and model utility preservation.

Similarly, our proposed method *IMU*, not only achieves strong forget capabilities, *i.e.*, 0.02% on $\mathcal{D}_f$, but also best preserves the model utility, with only about a 2% drop on both $\mathcal{D}_r$ and $\mathcal{D}_t$, indicating that *IMU* strikes a desirable balance between effective forgetting and minimal impact on model utility. In addition, *IMU* demonstrates favorable efficiency, is over 3$\times$ faster than SCAR*.

4.3 Results on subclass-wise unlearning

We further evaluate MU methods on CIFAR-100, trying to forget a single subclass from a semantic superclass. Specifically, (1) for subclass-wise unlearning across all super-classes, we randomly select a subclass from each of the 20 super-classes in CIFAR-100 to be forgotten, and report the average performance; (2) for subclass-wise unlearning within a single super-class, we focus on the super-class 'people', where each time, one of the five subclasses is selected to be forgotten while the remaining four are retained. These settings are more challenging than the standard class-wise unlearning, as subclasses within a superclass share semantically and visually similar features, making the forgetting more fine-grained and less separable.

Despite the increased difficulty, *IMU* consistently demonstrates better performance than other MU methods. As shown in Table 1, all MU methods completely remove the

Table 1: Quantitative results on CIFAR-10 and CIFAR-100. Results are averaged over all 10 classes for class-wise unlearning on CIFAR-10, across 20 superclasses, and across 5 subclasses in one superclass for subclass-wise unlearning on CIFAR-100.

Setting	Method	$\mathcal{D}_r$	$\mathcal{D}_f$	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{ACC}_{\mathcal{D}_r}(\uparrow)$	$\texttt{ACC}_{\mathcal{D}_t}(\uparrow)$	MIA (↑)	$W_{\texttt{dist}}(\downarrow)$	Run time (s) (↓)
CIFAR-10	Original	✓	✓	99.71±0.00	99.45±0.00	94.43±0.00	0.01±0.00	–	–
	Retrain	✓	✗	0.00±0.00	99.98±0.00	94.41±0.35	1.00±0.00	0.00±0.00	–
	IU*	✓	✓	0.56±0.76	84.04±9.71	79.33±8.46	0.99±0.00	4.51±1.66	55±1
	GA	✗	✓	0.48±0.10	75.89±6.28	72.43±5.83	0.99±0.00	3.59±0.39	29±0
	RL	✗	✓	0.71±0.42	79.01±13.25	73.33±12.25	0.99±0.00	4.04±0.61	25±5
	SCAR*	✗	✓	0.50±1.03	85.78±4.15	80.99±4.07	1.00±0.00	4.80±0.62	325±11
	NPO	✗	✓	0.20±0.28	91.63±7.23	85.84±6.66	1.00±0.00	0.24±0.03	39±2
	IMU (ours)	✗	✓	**0.02**±0.05	**97.68**±0.83	**91.67**±1.16	**1.00**±0.00	**0.11**±0.05	99±0
CIFAR-100	Original	✓	✓	99.89±0.19	99.99±0.02	76.63±5.47	0.01±0.00	–	–
	Retrain	✓	✗	0.00±0.00	99.53±2.79	74.05±8.95	1.00±0.00	0.00±0.00	–
	IU*	✓	✓	0.00±0.00	69.29±4.00	61.85±5.18	1.00±0.00	4.82±2.00	19±2
	SSD	✓	✓	0.00±0.00	77.37±9.93	46.85±2.90	1.00±0.00	3.15±1.27	10±0
	GA	✗	✓	0.00±0.00	68.29±5.27	40.89±8.61	1.00±0.00	5.22±0.14	9±0
	RL	✗	✓	0.00±0.00	73.87±8.81	42.80±7.68	1.00±0.00	7.73±0.81	22±3
	SCAR*	✗	✓	0.00±0.00	77.79±1.28	60.00±1.48	1.00±0.00	7.13±1.51	68±0
	NPO	✗	✓	0.00±0.00	90.47±3.15	55.80±7.55	1.00±0.00	5.35±1.10	25±3
	IMU (ours)	✗	✓	**0.00**±0.00	**98.06**±1.35	**63.75**±7.85	**1.00**±0.00	**2.09**±0.37	14±0
CIFAR-100	Original	✓	✓	100.00±0.00	100.00±0.00	77.80±2.98	0.00±0.00	–	–
	Retrain	✓	✗	0.00±0.00	99.33±1.16	76.49±2.53	1.00±0.00	0.00±0.00	–
	IU*	✓	✓	0.00±0.00	60.58±4.89	54.50±5.31	1.00±0.00	5.80±0.08	18±0
	SSD	✓	✓	0.00±0.00	83.07±1.13	52.20±1.32	1.00±0.00	4.81±0.50	10±0
	GA	✗	✓	0.00±0.00	75.56±9.46	49.50±6.16	1.00±0.00	7.54±1.29	9±0
	RL	✗	✓	0.00±0.00	76.49±11.51	47.69±4.54	1.00±0.00	7.57±0.07	19±2
	SCAR*	✗	✓	0.00±0.00	78.72±4.80	56.55±5.31	1.00±0.00	5.88±0.25	67±0
	NPO	✗	✓	0.00±0.00	90.22±4.57	58.90±6.34	1.00±0.00	5.61±0.25	26±2
	IMU (ours)	✗	✓	**0.00**±0.00	**98.61**±0.85	**67.60**±3.44	**1.00**±0.00	**3.69**±0.16	14±0

Note: IU* is the improved version of IU presented in Jia et al. (2023), and SCAR* denotes the version using $\mathcal{D}_f$ only.

Table 2: Unlearning on person re-identification.

Method	mAP (↑)	Top-1 (↑)	Top-5 (↑)	Run time (↓)
Original	68.50	85.45	94.27	–
IU*	44.79	69.69	84.59	147
GA	4.12	18.59	32.66	43
RL	0.83	0.02	0.06	104
SSD	51.79	73.57	87.35	137
NPO	41.40	63.13	81.03	55
SCAR*	50.97	73.87	88.54	135
IMU (Ours)	**55.85**	**76.07**	**88.75**	43

Table 3: Unlearning on a sequence modeling problem.

Method	Model utility		Forget quality	
	ℓ_r (↓)	$\ell_r^{\texttt{KL}}$ (↓)	ℓ_f (↑)	$\ell_f^{\texttt{KL}}$ (↑)
Original	1.99	-	2.18	-
GA	4.24	2.21	6.95	3.42
NPO	3.97	1.95	7.26	3.46
SimNPO	4.61	2.57	7.44	3.83
IMU (Ours)	**3.86**	**1.83**	**7.53**	**3.96**

knowledge about $\mathcal{D}_f$, while *IMU* best preserving the performance over $\mathcal{D}_r$, having an accuracy around 98% and achieves a high test accuracy compared to other baselines.

4.4 Results on person re-identification

Pedestrians' body shapes, clothing, facial features, and other biometric characteristics are often sensitive and uniquely identifiable, making them important targets for protection in MU scenarios. To this end, we also evaluate MU methods on the person re-identification task. This task involves matching images of the same individual captured under varying camera viewpoints or multiple cameras. Here, we aim to forget all samples associated with a particular identity (*e.g.*, $\texttt{pid} = 1$). This task poses unique challenges to MU methods, as forgetting a specific individual requires precise removal of identity-related features while preserving general person-level recognition performance.

We adopt GradCAM (Selvaraju et al. 2017) to visualize regions where models focus on **w/o** and **w/** *IMU*. As shown in Figure 3, when evaluated on $\mathcal{D}_r$, *IMU* preserves focus on discriminative yet identity-agnostic regions. In contrast, when evaluated on $\mathcal{D}_f$, our scrubbed model signifi-

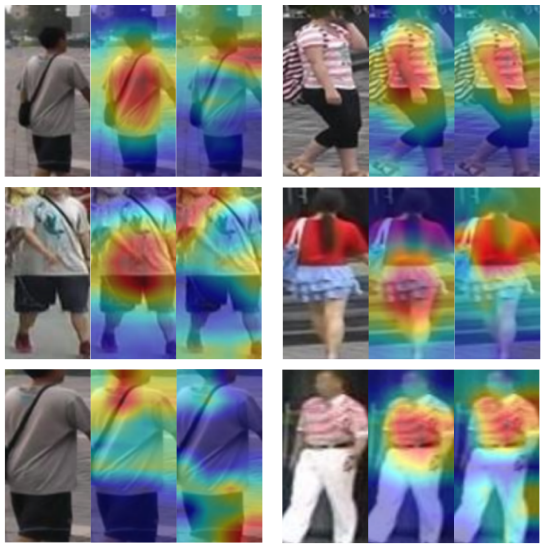

Figure 3: Visualizations of regions where models focus on for $\mathcal{D}_f$ and $\mathcal{D}_r$, respectively. For each triplet, from left to right are the original image, the activation map generated by the original model, and *IMU* scrubbed model, respectively.

cantly shifts attention away from identity-revealing regions (*e.g.*, clothing logos, face, and hairstyle), indicating successful removal of sensitive cues. In Table 2, we further evaluate model utility by measuring mAP and Top-k accuracy on a query-gallery split consisting of individuals entirely unseen during training. In general, *IMU* achieves a good trade-off between forget quality and model utility.

4.5 Case study: Markov chains

Aside from vision tasks, we further evaluate our method on a simple sequence modeling problem. Following the setting in SimNPO (Fan et al. 2024a), we construct a mixture of Markov chains with a state space of size 10, the retain distribution consists of Markov chains that transition uniformly among states $\{1, 2, 3\}$, while the forget distribution is a mixture of two Markov chains Forget1 (transition uniformly among $\{4, 5, 6\}$) and Forget2 (transition uniformly among $\{7, 8, 9\}$) with equal probability. Each sequence has length T and is denoted as $s = (s_1, s_2, \cdots, s_T)$ where $s_t \in \{0, 1, \cdots, 9\}$ for $t \in [0, T]$. A GPT-2 model is trained to approximate the conditional distribution. Details can be found in §7 of SimNPO (Fan et al. 2024a). Table 3 presents results compared to NPO and SimNPO. Performance is evaluated by the loss values over the retain and forget sets, as well as the respective KL divergence distance calculated compared with the retrained model. Our method surpasses NPO and SimNPO in both forget quality and model utility.

4.6 Ablation study

We finally conduct ablation studies to analyze the effects of two key design choices in our method: (1) the ratio r of top-ranked forgetting data selected based on influence values, and (2) the update frequency v of influence scores during the unlearning process. Table 4 presents the effect of varying the top-r fraction of the forgetting data $\mathcal{D}_f$ for unlearning. As r decreases, the model retains strong forget quality as evidenced by near-zero accuracy on $\mathcal{D}_f$, while achieving slightly improved accuracy on $\mathcal{D}_r$ and $\mathcal{D}_t$. Notably, using only 5% of the most influential data points is sufficient to induce forgetting, while also enhancing generalization and utility, highlighting the efficiency of targeting high-impact data points. Table 5 investigates the impact of varying the frequency v of influence score updates. All variants achieve strong forget quality, with minor differences in model utility. Infrequent updates are sufficient for maintaining effective unlearning, suggesting that *IMU* is robust.

Table 4: Impact of varying the ratio (r) of top-ranked $\mathcal{D}_f$ (selected based on $\mathcal{I}^c(\boldsymbol{x}, \mathcal{D}_f)$) on unlearning performance.

r	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{Acc}_{\mathcal{D}_r}(\uparrow)$	$\texttt{Acc}_{\mathcal{D}_t}(\uparrow)$	MIA (↑)	Run time (↓)
1.00	0.02±0.03	97.93±0.84	91.22±0.77	1.00±0.00	70±0
0.80	0.18±0.32	97.89±0.80	91.19±0.58	1.00±0.00	63±0
0.60	0.04±0.08	97.94±0.86	91.32±0.62	1.00±0.00	62±0
0.40	0.03±0.05	98.00±0.71	91.14±0.83	1.00±0.00	60±0
0.20	0.00±0.00	98.62±0.70	91.98±0.53	1.00±0.00	58±0
0.05	0.00±0.00	99.79±0.19	93.78±0.24	1.00±0.00	57±0

Table 5: Impact of frequency (ν) of influence value updates on unlearning performance. $\nu = 0$ means only update at the first epoch; $\nu = 1$ and $\nu = 2$ mean update every epoch and every two epochs, respectively.

ν	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{Acc}_{\mathcal{D}_r}(\uparrow)$	$\texttt{Acc}_{\mathcal{D}_t}(\uparrow)$	MIA (↑)	Run time (↓)
0	0.02±0.02	97.75±0.70	91.86±0.53	1.00±0.00	76±0
1	0.00±0.00	97.65±0.73	91.71±0.54	1.00±0.00	268±1
2	0.00±0.00	97.72±0.73	91.82±0.58	1.00±0.00	175±3

5 Conclusion, Limitations, Broader Impacts

In this paper, we introduce the retain-data-free MU method *IMU*, a novel paradigm that dynamically allocates attention to each data point according to its estimated influence value. Extensive experiments on various unlearning settings across vision and language tasks demonstrate that *IMU* effectively strikes the balance between forget quality and model utility, showing superiority compared to existing baselines.

However, our method still requires estimating the influence function for each forgetting data point. As a result, the computational cost scales with the number of data points, which can be time-consuming in real-world scenarios. One possible alternative is to estimate the influence at the mini-batch level rather than per data point. While this may reduce computation time, it could also introduce estimation errors. Besides, adversarial examples or outliers may interfere with influence estimation, leading to inaccurate decisions.

As MU becomes increasingly important, our method offers a practical solution that does not rely on the retain set. However, care must be taken to ensure robustness against adversarial manipulation of influence scores and to mitigate any potential misuse of the method.

References

Alaa, A.; and Van Der Schaar, M. 2020. Discriminative jackknife: Quantifying uncertainty in deep learning via higher-order influence functions. In *International Conference on Machine Learning (ICML)*, 165–174. PMLR.

Alberti, S.; Hasanaliyev, K.; Shah, M.; and Ermon, S. 2025. Data Unlearning in Diffusion Models. *International Conference on Learning Representations (ICLR)*.

Bae, J.; Ng, N.; Lo, A.; Ghassemi, M.; and Grosse, R. B. 2022. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 17953–17967.

Balunović, M.; Dimitrov, D. I.; Staab, R.; and Vechev, M. 2022. Bayesian Framework for Gradient Leakage. In *International Conference on Learning Representations (ICLR)*.

Basu, S.; Pope, P.; and Feizi, S. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations (ICLR)*.

Basu, S.; You, X.; and Feizi, S. 2020. On second-order group influence functions for black-box predictions. In *International conference on machine learning (ICML)*, 715–724. PMLR.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828.

Bonato, J.; Cotogni, M.; and Sabetta, L. 2024. Is Retain Set All You Need in Machine Unlearning? Restoring Performance of Unlearned Models with Out-of-Distribution Images. In *European Conference on Computer Vision (ECCV)*, 1–19. Springer.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine Unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159.

Bui, A.; Doan, K.; Le, T.; Montague, P.; Abraham, T.; and Phung, D. 2024. Removing Undesirable Concepts in Text-to-Image Generative Models with Learnable Prompts. *arXiv preprint arXiv:2403.12326*.

Cao, Y.; and Yang, J. 2015. Towards Making Systems Forget with Machine Unlearning. In *2015 IEEE Symposium on Security and Privacy (SP)*, 463–480.

Cha, S.; Cho, S.; Hwang, D.; Lee, H.; Moon, T.; and Lee, M. 2024. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 11186–11194.

Chen, M.; Gao, W.; Liu, G.; Peng, K.; and Wang, C. 2023. Boundary Unlearning: Rapid Forgetting of Deep Networks via Shifting the Decision Boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7766–7775.

Chhabra, A.; Li, P.; Mohapatra, P.; and Liu, H. 2024. " What Data Benefits My Classifier?" Enhancing Model Performance and Interpretability through Influence-Based Data Selection. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Dorna, V.; Mekala, A.; Zhao, W.; McCallum, A.; Lipton, Z. C.; Kolter, J. Z.; and Maini, P. 2025. OpenUnlearning: Accelerating LLM Unlearning via Unified Benchmarking of Methods and Metrics. *arXiv preprint arXiv:2506.12618*.

Epifano, J. R.; Ramachandran, R. P.; Masino, A. J.; and Rasool, G. 2023. Revisiting the fragility of influence functions. *Neural Networks*, 162: 581–588.

Fan, C.; Liu, J.; Lin, L.; Jia, J.; Zhang, R.; Mei, S.; and Liu, S. 2024a. Simplicity Prevails: Rethinking Negative Preference Optimization for LLM Unlearning. *arXiv preprint arXiv:2410.07163*.

Fan, C.; Liu, J.; Zhang, Y.; Wei, D.; Wong, E.; and Liu, S. 2024b. SalUn: Empowering Machine Unlearning via Gradient-based Weight Saliency in Both Image Classification and Generation. In *International Conference on Learning Representations (ICLR)*.

Foster, J.; Fogarty, K.; Schoepf, S.; Dugue, Z.; Öztireli, C.; and Brintrup, A. 2025. An information theoretic approach to machine unlearning. *Transactions on Machine Learning Research*.

Foster, J.; Schoepf, S.; and Brintrup, A. 2024. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 12043–12051.

Gandikota, R.; Materzynska, J.; Fiotto-Kaufman, J.; and Bau, D. 2023. Erasing concepts from diffusion models. In *2023 IEEE International Conference on Computer Vision (ICCV)*.

Geiping, J.; Bauermeister, H.; Dröge, H.; and Moeller, M. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 16937–16947.

Ginart, A.; Guan, M.; Valiant, G.; and Zou, J. Y. 2019. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Golatkar, A.; Achille, A.; Ravichandran, A.; Polito, M.; and Soatto, S. 2021. Mixed-Privacy Forgetting in Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 792–801.

Golatkar, A.; Achille, A.; and Soatto, S. 2020a. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9301–9309.

Golatkar, A.; Achille, A.; and Soatto, S. 2020b. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision (ECCV)*, 383–398. Springer.

Guo, C.; Goldstein, T.; Hannun, A.; and Van Der Maaten, L. 2020. Certified Data Removal from Machine Learning Models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, 3832–3842. PMLR.

Hampel, F. R. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346): 383–393.

Han, X.; Wallace, B. C.; and Tsvetkov, Y. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. *arXiv preprint arXiv:2005.06676*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Heng, A.; and Soh, H. 2023. Continual Learning for Forgetting in Deep Generative Models. In *International Conference on Machine Learning workshop*.

Jang, J.; Yoon, D.; Yang, S.; Cha, S.; Lee, M.; Logeswaran, L.; and Seo, M. 2022. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504*.

Jia, J.; Liu, J.; Ram, P.; Yao, Y.; Liu, G.; Liu, Y.; Sharma, P.; and Liu, S. 2023. Model sparsification can simplify machine unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Karasuyama, M.; and Takeuchi, I. 2010. Multiple Incremental Decremental Learning of Support Vector Machines. *IEEE Transactions on Neural Networks*, 21(7): 1048–1059.

Ko, M.; Li, H.; Wang, Z.; Patsenker, J.; Wang, J. T.; Li, Q.; Jin, M.; Song, D.; and Jia, R. 2024. Boosting alignment for post-unlearning text-to-image generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 37: 85131–85154.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning (ICML)*, 1885–1894. PMLR.

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

Kumari, N.; Zhang, B.; Wang, S.-Y.; Shechtman, E.; Zhang, R.; and Zhu, J.-Y. 2023. Ablating concepts in text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 22691–22702.

Kurmanji, M.; Triantafillou, P.; Hayes, J.; and Triantafillou, E. 2023. Towards unbounded machine unlearning. *Advances in neural information processing systems (NeurIPS)*, 36: 1957–1987.

Li, P.; and Liu, H. 2022. Achieving fairness at no utility cost via data reweighing with influence. In *International conference on machine learning (ICML)*, 12917–12930. PMLR.

Lin, S.; Zhang, X.; Susilo, W.; Chen, X.; and Liu, J. 2024. GDR-GMA: Machine Unlearning via Direction-Rectified and Magnitude-Adjusted Gradients. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 9087–9095.

Liu, Y.; Xu, L.; Yuan, X.; Wang, C.; and Li, B. 2022. The Right to be Forgotten in Federated Learning: An Efficient Realization with Rapid Retraining. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 1749–1758.

Lyu, M.; Yang, Y.; Hong, H.; Chen, H.; Jin, X.; He, Y.; Xue, H.; Han, J.; and Ding, G. 2024. One-dimensional Adapter to Rule Them All: Concepts Diffusion Models and Erasing Applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7559–7568.

Mehta, R.; Pal, S.; Singh, V.; and Ravi, S. N. 2022. Deep Unlearning via Randomized Conditionally Independent Hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10412–10421.

Meng, Y.; Xia, M.; and Chen, D. 2024. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37: 124198–124235.

Neel, S.; Roth, A.; and Sharifi-Malvajerdi, S. 2021. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, Proceedings of Machine Learning Research, 931–962. PMLR.

Pruthi, G.; Liu, F.; Kale, S.; and Sundararajan, M. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 19920–19930.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36: 53728–53741.

Rezaei, S.; and Liu, X. 2021. On the Difficulty of Membership Inference Attacks. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7888–7896.

Romero, E.; Barrio, I.; and Belanche, L. 2007. Incremental and decremental learning for linear support vector machines. In *International Conference on Artificial Neural Networks*, 209–218. Springer.

Sekhari, A.; Acharya, J.; Kamath, G.; and Suresh, A. T. 2021. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 18075–18086.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–626.

Spartalis, C. N.; Semertzidis, T.; Gavves, E.; and Daras, P. 2025. LoTUS: Large-Scale Machine Unlearning with a Taste of Uncertainty. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 10046–10055.

Tarun, A. K.; Chundawat, V. S.; Mandal, M.; and Kankanhalli, M. 2023. Deep regression unlearning. In *International Conference on Machine Learning*, 33921–33939. PMLR.

Thudi, A.; Deza, G.; Chandrasekaran, V.; and Papernot, N. 2022. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, 303–319. IEEE.

Watson, L.; Guo, C.; Cormode, G.; and Sablayrolles, A. 2022. On the Importance of Difficulty Calibration in Membership Inference Attacks. In *International Conference on Learning Representations (ICLR)*.

Wu, G.; Hashemi, M.; and Srinivasa, C. 2022. Puma: Performance unchanged model augmentation for training data removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8675–8682.

Wu, J.; and Harandi, M. 2024. Scissorhands: Scrub data influence via connection sensitivity in networks. In *European Conference on Computer Vision (ECCV)*, 367–384. Springer.

Wu, Y.; Dobriban, E.; and Davidson, S. 2020. DeltaGrad: Rapid retraining of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, 10355–10366. PMLR.

Zhang, R.; Lin, L.; Bai, Y.; and Mei, S. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. In *Conference on Language Modeling (COLM)*.

Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; and Tian, Q. 2015. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, 1116–1124.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep leakage from gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

IMU: Influence-guided Machine Unlearning

Supplementary Material

6 Details

6.1 Implementation details

All the experiments were performed on NVIDIA RTX3090 with Intel Xeon processors. For **image classification** scenario, the original model and retrained model were both trained with over 182 epochs using the SGD optimizer with a cosine-scheduled, and learning rate was initialized at 0.1. For both NPO and RL, training spans 10 epochs within the interval $[10^{-4}, 10^{-1}]$, and β was selected from the range $[0.1, 5]$. GA's training settings around a 5-epoch learning rate search within the interval $[10^{-5}, 10^{-2}]$. In the case of IU, we explore the parameter α within the range $[1, 20]$. For SSD, searching for the parameter selection weighting α and dampening constant γ is executed within the range $[1, 100]$ and $[0.1, 1]$ respectively, while searching for the learning rate within the range $[10^{-4}, 10^{-2}]$. For the SCAR method, the value of temperature was defined within the range $[2, 5]$, the parameter λ_2 was set as 0.01, and λ_1 was in the range of $[1, 10]$ with the learning rate searching in the interval $[10^{-6}, 10^{-2}]$. Learning rates was selected in the range $[10^{-4}, 10^{-2}]$ for SalUn and let sparsity ratios equal to 0.5. Lastly, for *IMU* method, we trained for 3 epochs, searching for learning rates in the range $[10^{-5}, 10^{-2}]$ and l_1 regularization intensity in the range $[2 \times 10^{-4}, 2 \times 10^{-2}]$. For **person re-identification** unlearning task, original model was trained over 60 epochs with learning rate as 3×10^{-4}. In the case of *IMU*, it is trained for 1.5×10^{-3} iterations with a learning rate of 1.5×10^{-3}, with α set to 0.02. For **sequence modeling unlearning** problem, the original model and retrained model were both trained with over 5 epochs using softmax activation, and the learning rate was initialized at 5×10^{-4}. As for SimNPO and NPO method, the value of β is searched in interval of $[0.1, 5]$ with learning rate in range $[2 \times 10^{-4}, 8 \times 10^{-3}]$, while the learning rate of GA was selected from $[2 \times 10^{-4}, 10^{-2}]$. For our *IMU* method, learning rate was selected as 5×10^{-4} with α set as 0.002. Additionally, all the unlearning methods were conducted over 50 iterations. Lastly, in **LLM unlearning** scenario, β was selected from $[1, 5]$ for NPO and SimNPO, and the learning rate was searched in range $[10^{-6}, 10^{-4}]$. Additionally, all methods in this task were executed over 6 epochs and distributed across 2 NVIDIA RTX 3090 GPUs. Our source code will be publicly available to support the reproducibility of the results.

6.2 Divergence analysis

To recap, the gradients of GA, NPO, and *IMU* are

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\texttt{GA}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f} \left[\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})\right], \tag{10}$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\texttt{NPO}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f} \left[\mathcal{W}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{y}) \cdot \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})\right], \tag{11}$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\texttt{IMU}} = -\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_f} \left[\hat{\mathcal{I}}^c(\boldsymbol{x}, \mathcal{D}_f) \cdot \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta})\right]. \tag{12}$$

Now, let us take the logistic regression as an example, where $P(y = 1|x) = \sigma(\boldsymbol{z}), P(y = 0|x) = 1 - \sigma(\boldsymbol{z})$ with $\boldsymbol{z} = \boldsymbol{\theta}^\top \boldsymbol{x} + b, \sigma(\boldsymbol{z}) = \frac{1}{1+\exp^{-z}}$ and b is a constant. The logistic model then can be written as $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma\left((2\boldsymbol{y} - 1) \cdot \boldsymbol{z}\right)$ and the loss function would be $\ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = -\log f_{\boldsymbol{\theta}}(\boldsymbol{x}) = -\log \sigma\left((2\boldsymbol{y} - 1) \cdot \boldsymbol{z}\right)$. Hence, we have

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \boldsymbol{x}\left(2\boldsymbol{y} - 1\right)\left(1 - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\right), \tag{13}$$

To perform the gradient update for one step, we can obtain

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \eta \cdot \boldsymbol{x}\left(2\boldsymbol{y} - 1\right)\left(1 - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}\right)\right). \tag{14}$$

Denote n_f as the number of training data points $\boldsymbol{X} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n_f}\right]^\top$. Assuming every step stochastically selects samples to update and update with stepsize η for t iterations.

For GA, we have

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^0 + \eta \sum_{i=1}^{n_f} \gamma_i^t \cdot \boldsymbol{x}_i\left(2\boldsymbol{y}_i - 1\right)\left(1 - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)\right), \tag{15}$$

where γ_i^t denotes the frequency of the sample $\boldsymbol{x}_i$ in the previous t steps. Then, the weighted norm would be

$$\begin{aligned}
\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^0\|^2_{\boldsymbol{X}^\top\boldsymbol{X}} &= \left(\boldsymbol{\theta}^t - \boldsymbol{\theta}^0\right)^\top \boldsymbol{X}^\top \boldsymbol{X}\left(\boldsymbol{\theta}^t - \boldsymbol{\theta}^0\right) \\
&= \eta^2 \sum_{i=1}^{n_f}\sum_{j=1}^{n_f} \gamma_i^t \gamma_j^t \cdot \left(2\boldsymbol{y}_i - 1\right)\left(2\boldsymbol{y}_j - 1\right) \cdot \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right)\left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}_j)\right) \cdot \boldsymbol{x}_i^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{x}_j, \\
&= \boldsymbol{a}^\top \boldsymbol{G} \boldsymbol{a},
\end{aligned} \tag{16}$$

where $a_i = \eta\gamma_i^t\left(2\boldsymbol{y}_i - 1\right)\left(1 - f_{\boldsymbol{\theta}}(x)\right)$ so $\boldsymbol{a} = \left[a_1, \cdots, a_{n_f}\right]^\top$, and $\boldsymbol{G}_{ij} = \boldsymbol{x}_i^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{x}_j$.

Namely, for GA, we have

$$\boxed{\|\boldsymbol{\theta}_{\texttt{GA}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 = \boldsymbol{a}^\top \boldsymbol{G}\boldsymbol{a}.} \tag{17}$$

For NPO, we have

$$\boldsymbol{\theta}_{\texttt{NPO}}^t = \boldsymbol{\theta}^0 + \eta \sum_{i=1}^{n_f} \gamma_i^t \cdot \mathcal{W}_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{y}_i) \cdot \boldsymbol{x}_i \left(2\boldsymbol{y}_i - 1\right)\left(1 - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)\right), \tag{18}$$

and

$$\boxed{\|\boldsymbol{\theta}_{\texttt{NPO}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 = (\mathcal{W}\boldsymbol{a})^\top \boldsymbol{G}(\mathcal{W}\boldsymbol{a}),} \tag{19}$$

where $\mathcal{W} = \operatorname{diag}\left(\mathcal{W}(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, \mathcal{W}(\boldsymbol{x}_{n_f}, \boldsymbol{y}_{n_f})\right)$ and $\mathcal{W}(\boldsymbol{x}_i, \boldsymbol{y}_i) \in [0, 2]$.

Similarly, for *IMU*, we have

$$\boldsymbol{\theta}_{\texttt{IMU}}^t = \boldsymbol{\theta}^0 + \eta \sum_{i=1}^{n_f} \gamma_i^t \cdot \hat{\mathcal{I}}^c(\boldsymbol{x}_i, \boldsymbol{X}) \cdot \boldsymbol{x}_i \left(2\boldsymbol{y}_i - 1\right)\left(1 - f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right)\right), \tag{20}$$

and

$$\boxed{\|\boldsymbol{\theta}_{\texttt{IMU}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 = (\hat{\mathcal{I}}^c\boldsymbol{a})^\top \boldsymbol{G}(\hat{\mathcal{I}}^c\boldsymbol{a}),} \tag{21}$$

where $\hat{\mathcal{I}}^c = \operatorname{diag}\left(\hat{\mathcal{I}}^c(\boldsymbol{x}_1, \boldsymbol{X}), \cdots, \hat{\mathcal{I}}^c(\boldsymbol{x}_{n_f}, \boldsymbol{X})\right)$ and $\hat{\mathcal{I}}^c(\boldsymbol{x}_i, \boldsymbol{X}) \in [0, 1]$.

From Equations (17), (19) and (21), we can bound the divergence for GA, NPO, and *IMU* as

$$\boxed{\begin{aligned}
\lambda_{\min}(\boldsymbol{G}) \cdot \|\boldsymbol{a}\|^2 \leq &\|\boldsymbol{\theta}_{\texttt{GA}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 \leq \lambda_{\max}(\boldsymbol{G}) \cdot \|\boldsymbol{a}\|^2, &(22)\\
\lambda_{\min}(\boldsymbol{G}) \cdot \|\mathcal{W}\boldsymbol{a}\|^2 \leq &\|\boldsymbol{\theta}_{\texttt{NPO}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 \leq \lambda_{\max}(\boldsymbol{G}) \cdot \|\mathcal{W}\boldsymbol{a}\|^2, &(23)\\
\lambda_{\min}(\boldsymbol{G}) \cdot \|\hat{\mathcal{I}}^c\boldsymbol{a}\|^2 \leq &\|\boldsymbol{\theta}_{\texttt{IMU}}^t - \boldsymbol{\theta}^0\|_{\boldsymbol{X}^\top\boldsymbol{X}}^2 \leq \lambda_{\max}(\boldsymbol{G}) \cdot \|\hat{\mathcal{I}}^c\boldsymbol{a}\|^2, &(24)
\end{aligned}}$$

where $\lambda_{\min}(\boldsymbol{G})$ and $\lambda_{\max}(\boldsymbol{G})$ are the smallest and largest eigenvalue of the Gram matrix $\boldsymbol{G}$, respectively. Since $\mathcal{W} \in [0, 2]$ (typically $\mathcal{W} \leq 1$) and $\hat{\mathcal{I}}^c \in [0, 1]$, we usually have $\|\mathcal{W}\boldsymbol{a}\|^2 \leq \|\boldsymbol{a}\|^2$, $\|\hat{\mathcal{I}}^c\boldsymbol{a}\|^2 \leq \|\boldsymbol{a}\|^2$. GA treats each sample equally and exhibits the fastest divergence, which may cause over-unlearning. NPO slows divergence by dampening high-confidence samples, while *IMU* reduces divergence by selectively updating only the most influential samples, promoting stability.

6.3 Derivation of influence function

The following derivation is adapted from the appendix of (Koh and Liang 2017), where the influence function is analyzed via a first-order perturbation approach. We consider the empirical risk minimizer $\boldsymbol{\theta}^*$, which is defined as

$$R(\boldsymbol{\theta}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{x}_i, \boldsymbol{\theta}), \tag{25}$$

where $\ell(\boldsymbol{x}_i, \boldsymbol{\theta})$ denotes the loss on training sample $\boldsymbol{x}_i$. Then, assume that R is twice differentiable and strongly convex w.r.t. $\boldsymbol{\theta}$. In particular, the Hessian of the empirical risk at the optimum $\boldsymbol{\theta}^*$ is given as

$$\boldsymbol{H}_{\boldsymbol{\theta}^*} \coloneqq \nabla_{\boldsymbol{\theta}}^2 R(\boldsymbol{\theta}^*) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}}^2 \ell(\boldsymbol{x}_i, \boldsymbol{\theta}^*), \tag{26}$$

which is assumed to be positive definite. This ensures that $\boldsymbol{H}_{\boldsymbol{\theta}^*}^{-1}$ exists and will be used in subsequent analysis.

Now consider perturbing the empirical risk by upweighting a particular training data point $\boldsymbol{x}$ by a small amount ϵ. So that there is

$$\boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} \left\{R(\boldsymbol{\theta}) + \epsilon\ell(\boldsymbol{x}, \boldsymbol{\theta})\right\}. \tag{27}$$

Then, define the parameter change under this perturbation as $\Delta_\epsilon = \boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^* - \boldsymbol{\theta}^*$. Since $\boldsymbol{\theta}^*$ is independent of ϵ, we have $\frac{d\boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^*}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}$. Because $\boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^*$ minimizes the perturbed objective, it satisfies the first-order optimality condition as follows

$$0 = \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^*) + \epsilon\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{x}, \boldsymbol{\theta}_{\epsilon,\boldsymbol{x}}^*). \tag{28}$$

We approximate this expression using a first-order Taylor expansion around $\boldsymbol{\theta}^*$, thus it derive that

$$0 \approx \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) + \epsilon \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*) + \left[\nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) + \epsilon \nabla^2_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*)\right] \Delta_\epsilon. \tag{29}$$

Solving for Δ_ϵ, we can obtain

$$\Delta_\epsilon \approx -\left[\nabla^2_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) + \epsilon \nabla^2_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*)\right]^{-1} \left[\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) + \epsilon \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*)\right]. \tag{30}$$

Since $\boldsymbol{\theta}^*$ minimizes R, we have $\nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}^*) = 0$, so Equation (30) can be rewritten as

$$\Delta_\epsilon \approx -\boldsymbol{H}^{-1}_{\boldsymbol{\theta}^*} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*) \epsilon. \tag{31}$$

Hence, the influence of $\boldsymbol{x}$ is given by

$$\left.\frac{d\boldsymbol{\theta}^*_{\epsilon,\boldsymbol{x}}}{d\epsilon}\right|_{\epsilon=0} = -\boldsymbol{H}^{-1}_{\boldsymbol{\theta}^*} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*). \tag{32}$$

Lastly, we can define the influence function as below

$$\mathcal{I}(\boldsymbol{x}) \coloneqq -\boldsymbol{H}^{-1}_{\boldsymbol{\theta}^*} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{x}, \boldsymbol{\theta}^*). \tag{33}$$

7 Additional results

7.1 Results on sample-wise unlearning

In Table 7, we evaluate the unlearning performance of *IMU* method on ResNet-18 for random forgetting on CIFAR-10. The results clearly manifest that our proposed approach achieves robust balance across these metrics with feasible run time, compared with other existing methods. Additionally, we also examine how different influence update frequencies affect sample-wise forgetting performance in Table 6. And we can also develop the conclusion that in this scenario, even infrequent updates are sufficient for effective unlearning.

Table 6: Impact of frequency (ν) of influence value updates on unlearning performance. $\nu = 0$ means only update at the first epoch; $\nu = 1$ and $\nu = 2$ mean update every epoch and every two epochs, respectively.

ν	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{Acc}_{\mathcal{D}_r}(\uparrow)$	$\texttt{Acc}_{\mathcal{D}_t}(\uparrow)$	MIA ($\uparrow$)	Run time ($\downarrow$)
0	97.82±0.31	97.78±0.53	91.90±0.82	0.12±0.06	55±0
1	97.47±0.42	97.55±0.49	91.82±0.	0.08±0.03	224±1
2	97.17±0.00	97.17±0.97	91.50±1.07	0.11±0.03	139±1

7.2 Results on LLM unlearning

In Table 8, we generalize our MU method on large language model and perform forgetting `forget05` task on the TOFU benchmark. Compared with other LLM unlearning baselines, *IMU* (ours) strikes a balance on forget quality and model utility. From Figure , we can clearly see the catastrophic collapse issue in the GA method and better performance of the NPO. Although the model structure of Llama-3.2-3B-Instruct is more complex, which is not conducive to the accurate estimation of influence function, our method has achieved an almost equivalent forgetting effect to SimNPO method.

7.3 Experiments on SalUn

Under the same dataset and experimental settings, we replicated and evaluated the current SOTA method based on the retained dataset and forgotten dataset SalUn (Fan et al. 2024b). The experimental results are shown in the Table 9. On the CIFAR-10 dataset, the *IMU* method we proposed performs similarly to SalUn, while on the CIFAR-100 dataset, SalUn still achieves relatively better performance.

7.4 Changes of influence score over epochs

In Figure 4, we respectively present the changes in the influence scores of 100 samples randomly selected from $\mathcal{D}_f$ over the 5 unlearning epochs in the class-wise and sample-wise forgetting tasks. In the class-wise forgetting task, since the samples are all belong to the same class, their values should all be negative according to the definition of the influence function formula. It is verified in Figure 4, and we can see the changes in the influence score of a specific sample after epoch 2 do not significantly fluctuate. Additionally, compared to the class-wise task, since the samples are randomly selected from different classes in the sample-wise forgetting scenario, the overall influence of each sample on forgetting data is possibly positive or negative. We can also observe that with more forgetting rounds, the number of samples with negative influence values gradually increases, demonstrating the effectiveness of our unlearning method.

Table 7: Quantitative results on CIFAR-10 and CIFAR-100. Performance is averaged over 10 independent runs with different random seeds for sample-wise unlearning.

Setting	Method	$\mathcal{D}_r$	$\mathcal{D}_f$	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{Acc}_{\mathcal{D}_r}(\uparrow)$	$\texttt{Acc}_{\mathcal{D}_t}(\uparrow)$	MIA (↑)	$W_{\texttt{dist}}(\downarrow)$	Run time (s) (↓)
CIFAR-10	Original	✓	✓	99.47±0.00	100.00±0.00	94.61±0.00	0.00±0.00	–	–
	Retrain	✓	✗	94.33±0.00	99.93±0.00	94.42±0.00	0.13±0.00	0.00±0.00	–
	IU*	✓	✓	99.06±0.11	99.03±0.03	92.98±0.06	0.02±0.01	0.06±0.04	54±1
	GA	✗	✓	99.04±0.47	98.75±0.27	92.88±0.18	0.02±0.00	0.07±0.04	31±1
	RL	✗	✓	98.90±0.55	98.90±0.51	93.49±0.55	0.11±0.00	0.03±0.01	24±0
	SSD	✓	✓	99.04±0.44	99.01±0.36	93.34±0.54	0.03±0.00	0.04±0.01	55±3
	SCAR*	✗	✓	98.78±0.25	98.94±0.14	93.17±0.24	0.13±0.02	0.04±0.01	703±2
	NPO	✗	✓	98.74±0.16	98.83±0.15	92.98±0.22	0.03±0.02	0.05±0.03	38±1
	IMU (ours)	✗	✓	**98.64±0.11**	**99.06±0.20**	**93.62±0.21**	0.03±0.00	0.08±0.01	35±0
CIFAR-100	Original	✓	✓	97.69±0.00	97.48±0.00	76.25±0.00	0.06±0.00	–	–
	Retrain	✓	✗	75.71±0.00	99.98±0.00	74.24±0.00	0.50±0.00	0.00±0.00	–
	IU*	✓	✓	94.58±1.62	94.71±1.76	70.11±0.77	0.12±0.01	0.67±0.21	57±1
	GA	✗	✓	94.42±0.93	93.36±0.57	70.37±0.15	0.11±0.01	1.07±0.51	32±1
	RL	✗	✓	93.82±1.46	94.14±1.51	66.58±1.05	0.06±0.03	1.07±0.18	31±1
	SSD	✗	✓	93.97±0.73	94.22±0.80	70.30±0.23	0.13±0.01	0.31±0.14	50±0
	SCAR*	✗	✓	93.82±1.04	95.18±1.03	70.70±0.60	0.14±0.01	0.81±0.11	726±4
	NPO	✗	✓	94.43±0.70	94.31±0.53	70.86±0.60	0.12±0.00	1.02±0.25	38±2
	IMU (ours)	✗	✓	**93.55±0.23**	**95.21±0.09**	**72.15±0.88**	**0.14±0.01**	0.74±0.31	62±2

Table 8: Results of LLM unlearning.

Method	Model Utility (↑)	Extraction Strength (↓)	Forget Q A Prob (↓)	Forget Q A ROUGE (↓)	Privleak (↓)
GA	0.00	0.03	0.03	0.00	48
NPO	0.28	0.05	0.06	0.21	80
SimNPO	0.35	0.05	0.17	0.33	-12
IMU (Ours)	0.33	0.06	0.09	0.29	-83

Table 9: Unlearning performance of SalUn under the same settings.

Setting	$\texttt{Acc}_{\mathcal{D}_f}(\downarrow)$	$\texttt{Acc}_{\mathcal{D}_r}(\uparrow)$	$\texttt{Acc}_{\mathcal{D}_t}(\uparrow)$	MIA (↑)	Run time (↓)
CIFAR-10 single class	0.07±0.18	99.40±0.39	87.29±0.48	1.00±0.00	151±2
CIFAR-100 subclass	0.00±0.00	96.80±4.05	83.40±9.17	1.00±0.00	24±0
CIFAR-100 subclass	0.00±0.00	99.61±0.47	91.35±2.83	1.00±0.00	29±4
CIFAR-10 random	97.35±0.23	98.98±0.16	93.03±0.15	0.12±0.00	237±3
CIFAR-100 random	78.27±1.08	98.85±0.42	70.03±0.78	0.69±0.02	249±6

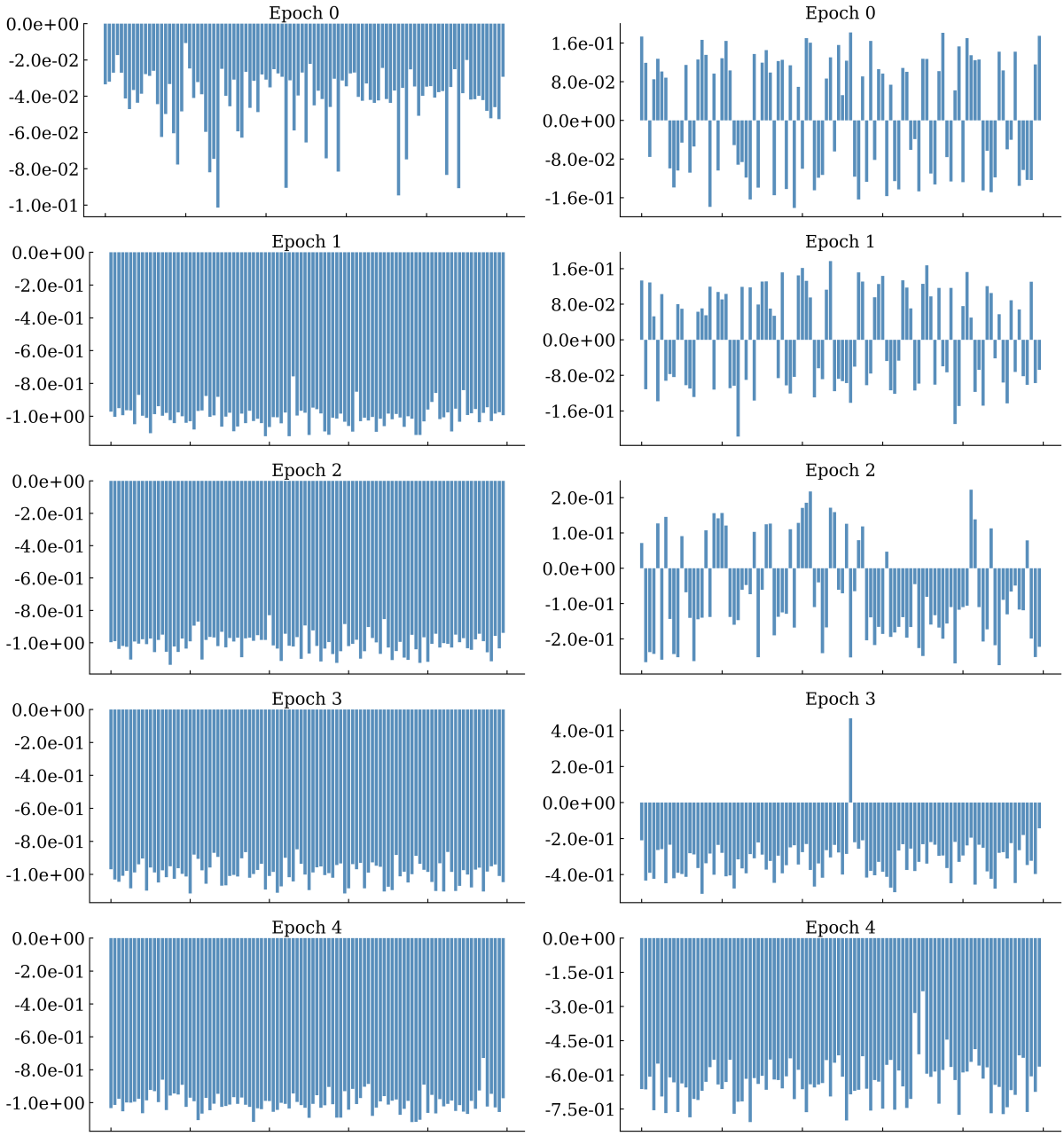

Figure 4: Change of sample influences for 5 epochs in CIFAR-10, the left column represents the class-wise forgetting tasks and the right column represents the random-wise forgetting tasks.