ACU: Analytic Continual Unlearning for Efficient and Exact Forgetting with Privacy Preservation

Jianheng Tang[1] **Huiping Zhuang**[2] **Di Fang**[2] **Jiaxu Li**[3] **Feijiang Han**[4] **Yajiang Huang**[3]
Kejia Fan[3] **Leye Wang**[1] **Zhanxing Zhu**[5] **Shanghang Zhang**[1] **Houbing Song**[6] **Yunhuai Liu**[1]

[1]Peking University, China   [2]South China University of Technology, China
[3]Central South University, China   [4]University of Pennsylvania, USA
[5]University of Southampton, UK   [6]University of Maryland, Baltimore County, USA

Abstract

The development of artificial intelligence demands that models incrementally update knowledge by Continual Learning (CL) to adapt to open-world environments. To meet privacy and security requirements, Continual Unlearning (CU) emerges as an important problem, aiming to sequentially forget particular knowledge acquired during the CL phase. However, existing unlearning methods primarily focus on single-shot joint forgetting and face significant limitations when applied to CU. First, most existing methods require access to the retained dataset for re-training or fine-tuning, violating the inherent constraint in CL that historical data cannot be revisited. Second, these methods often suffer from a poor trade-off between system efficiency and model fidelity, making them vulnerable to being overwhelmed or degraded by adversaries through deliberately frequent requests. In this paper, we identify that the limitations of existing unlearning methods stem fundamentally from their reliance on gradient-based updates. To bridge the research gap at its root, we propose a novel gradient-free method for CU, named **A**nalytic **C**ontinual **U**nlearning (ACU), for efficient and exact forgetting with historical data privacy preservation. In response to each unlearning request, our ACU recursively derives an analytical (i.e., closed-form) solution in an interpretable manner using the least squares method. Theoretical and experimental evaluations validate the superiority of our ACU on unlearning effectiveness, model fidelity, and system efficiency.

1 Introduction

The development of Artificial Intelligence (AI) in open-world environments demands models capable of continual updates in response to external requests throughout their lifespan [1, 2, 3]. Each update can be either incremental or decremental. Incremental updates enable models to learn new tasks over time, ideally without accessing previous data and without forgetting previous knowledge [4, 5, 6]. This issue has been extensively studied within Continual Learning (CL), where the currently predominant methods leverage pre-trained models to assist with feature extraction [7, 8, 9]. In contrast, decremental updates focus on selectively erasing specific knowledge tied to certain data mainly for upholding the granted "*right to be forgotten*", as explored in the emerging field of machine unlearning [10, 11, 12].

Although many recent advances have shown progress in machine unlearning, most existing methods focus solely on single-shot joint forgetting and require access to the historically retained dataset, violating the requirements of CL [13, 14, 15, 16]. In practice, due to frequent triggers from various legitimate or adversarial users, it is common to face multiple requests for Continual Unlearning (CU), which necessitate the sequential removal of knowledge acquired during the CL phase, as shown in Figure 1. Following the CL philosophy that online task data is discarded after training [13, 14, 15, 16], the historically retained dataset is typically unavailable during the CU phase. Thus, the only available information for each unlearning request is limited to the designated samples to be forgotten.
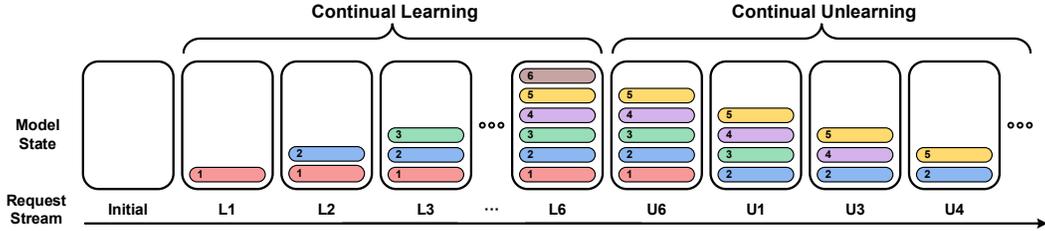
Figure 1: Illustration of model state evolution under the CL and CU phases. The model begins blank with no knowledge. As the CL phase progresses, the model incrementally acquires knowledge from online tasks. After the CL phase, the CU requests are sequentially issued to remove particular knowledge acquired during the CL phase. Since the online data from the CL phase is discarded after training, only the designated samples to be forgotten can be accessible for each CU request.

Even if historical data are available and privacy concerns are ignored, frequent triggering of requests during CU still poses a key challenge. Considering the cumulative effect of multiple requests, system efficiency and model fidelity are key concerns for CU. However, existing unlearning methods often exhibit a poor trade-off between these two indicators [10, 17, 18]. On the one hand, exact unlearning methods typically rely on partial re-training using the retained dataset, which incurs substantial computational and time overhead [18, 19, 20]. The accumulated inefficiency in CU creates a potential vulnerability to Denial-of-Service (DoS) attacks, where a single forgetting target may be fragmented into multiple frequent requests to overwhelm the system. On the other hand, relatively efficient methods are often limited to approximate unlearning, which inevitably risks unintended removal of valuable knowledge that ought to be preserved, thus compromising the model fidelity on non-forgotten data [21, 22, 23]. Even minor compromises in model fidelity can accumulate and compound under multiple requests, potentially leading to catastrophic degradation or even collapse of the model.

In summary, existing methods exhibit two prominent limitations in CU: (1) the prevailing reliance on the historically retained data, and (2) the poor trade-off between system efficiency and model fidelity. In this paper, we identify that the limitations of existing unlearning methods stem fundamentally from their reliance on gradient-based model updates. Such updates render learning from data a complex, entangled, and incremental process with inherent stochasticity, making the influence of data on the model challenging to explicitly characterize, let alone exactly erase. It is precisely for this reason that existing methods require access to the retained data for re-training or fine-tuning, as a means to recover model fidelity. Many unlearning studies have recognized the limitations from gradient-based updates, but few have attempted to fundamentally address this issue at its root [10, 24, 25].

Concurrently, the CL community has witnessed the emergence of gradient-free methods grounded in analytic learning [15, 16, 26, 27, 28], which have been widely acknowledged for achieving state-of-the-art performances [29, 30, 31, 32, 33]. The applications of analytic learning in CL provide a timely and valuable opportunity for us to fundamentally address the identified challenges in CU by circumventing gradient-based updates. Motivated by the preceding observations, we propose a novel gradient-free method for CU, named **A**nalytic **C**ontinual **U**nlearning (ACU), to achieve efficient and exact forgetting while preserving the privacy of historical data. In our proposed ACU, we align with the prevailing practice in CL by leveraging frozen pre-trained models as powerful gradient-free feature extractors. In response to each unlearning request, our ACU can recursively derive the analytical (i.e., closed-form) solution based on the least squares method, achieving exact knowledge forgetting in an efficient and interpretable manner. Our main contributions are summarized as follows:

1. We introduce a novel and practical problem, CU, which aims at sequentially removing the knowledge acquired during the CL phase, without access to the historically retained dataset.

2. By identifying the fundamental limitations of existing methods in addressing CU, we propose our ACU as a gradient-free method for efficient and exact forgetting with data privacy.

3. We theoretically validate that our ACU achieves exact unlearning using only the designated samples to be forgotten per request, yielding an identical result to that from joint re-training on the remaining dataset. This rare property aligns with the pursuit of responsible AI.

4. Extensive experiments on benchmark datasets demonstrate the superiority of our ACU over other baselines regarding unlearning effectiveness, model fidelity, and system efficiency. In particular, its advantages become even more pronounced in the frequent-request scenarios.

2 Related Works

2.1 Machine Unlearning

Machine unlearning has emerged as a popular topic, aiming to delete specific data and eliminate the corresponding influence on AI models [10, 12, 34]. This topic was originally motivated by the need to meet privacy and regulatory requirements, and later expanded to include the removal of erroneous or biased knowledge from toxic data [11, 35, 36, 37]. Existing unlearning methods typically assume that the original model is trained in a centralized joint manner, with access to the retained data for re-training or fine-tuning [10, 18, 22, 38]. Yet, this assumption is often impractical in open-world environments, where models are trained in a CL fashion and historical data cannot be revisited.

Moreover, most unlearning methods suffer from a poor trade-off between system efficiency and model fidelity, which is particularly problematic under the frequent unlearning requests of CU [36, 39, 40]. Specifically, existing methods for exact unlearning tend to incur substantial overhead [18, 19, 20], while the approximate ones risk compromising model fidelity on non-forgotten knowledge [21, 22, 23]. In this context, a malicious user can fragment a single forgetting target into multiple frequent requests to overwhelm or degrade the system [10]. Several studies have identified that the core challenge of unlearning lies in gradient-based updates, which render the influence of data on the model a complex, entangled, and incremental process with inherent stochasticity [10, 24, 25]. However, few efforts have been made to fundamentally address this issue at its root by circumventing gradient-based updates. In contrast, our ACU differs from existing work by introducing a novel gradient-free method, which enables efficient and exact forgetting while preserving the privacy of historical data.

2.2 Analytic Learning

Analytic learning is a prominent gradient-free approach, initially developed to resolve challenges related to gradients, such as the vanishing and exploding gradient problems [27, 26]. This approach is also termed pseudoinverse learning, owing to its flexible application of matrix inversion [41, 42, 43]. A prime example of shallow analytic learning is the radial basis network, which utilizes least squares estimation to train its parameters after performing a kernel transformation in the first layer [44]. Furthermore, by employing least squares methods to linearize the learning process of nonlinear networks, multilayer analytic learning has seen widespread use in various applications [45, 46, 47, 48].

Earlier methods of analytic learning encountered significant memory limitations, as they required the entire dataset to be processed in a single batch [49]. This challenge has been addressed through the block-wise recursive Moore-Penrose inverse, which substitutes the traditional joint learning process with a more efficient recursive approach [49]. Building on this improvement, analytic learning has demonstrated state-of-the-art performance across a wide range of tasks [15, 16, 50, 51], particularly within the field of CL [7, 26, 27, 28, 29, 30, 31, 32, 33]. In this paper, we build upon the recent developments in analytic learning for CL, aiming to bridge the research gap in CU. By leveraging the advantages of gradient-free updates, our ACU enables efficient and exact forgetting while preserving historical data privacy, which presents a highly rare and nearly ideal property for CU.

3 Our Proposed ACU Method

In the CU phase, we aim at continuously erasing the influence of designated data on the model in response to the online unlearning requests, where the data to be forgotten was previously used for training during the CL phase. Thus, we consider the original model $\mathbf{W}_0$ to be trained in a CL fashion, where all the training data is collectively denoted as $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{N}$, comprising a total of N samples, with x_j and y_j representing the input and its corresponding label of the j-th sample. Then, in the CU phase, the data designated for unlearning within the online requests are denoted as a sequence of forgetting sets $\check{\mathcal{D}} = \{\check{\mathcal{D}}_1, \cdots, \check{\mathcal{D}}_i, \cdots\}$, where $\check{\mathcal{D}}_i \subseteq \mathcal{D}$ and $\check{\mathcal{D}}_i \cap \check{\mathcal{D}}_{i'} = \emptyset$ for $\forall i \neq i'$. For each unlearning request i, we define the corresponding unlearned model as $\mathbf{W}_i$. The goal of each unlearning request is to make the model as close as possible to the one re-trained from scratch on the retained set $\hat{\mathcal{D}}_i = \mathcal{D} \setminus \bigcup_{k=1}^{i} \check{\mathcal{D}}_k$ (excluding all data to be forgotten), as if the unlearned data had never been used in training. Notably, the retained set $\hat{\mathcal{D}}_i$ is inaccessible in practice, because the online task data are all discarded once training is completed during the CL phase. Thus, the only available information for each unlearning request i is limited to the designated samples to be forgotten, i.e., $\check{\mathcal{D}}_i$.
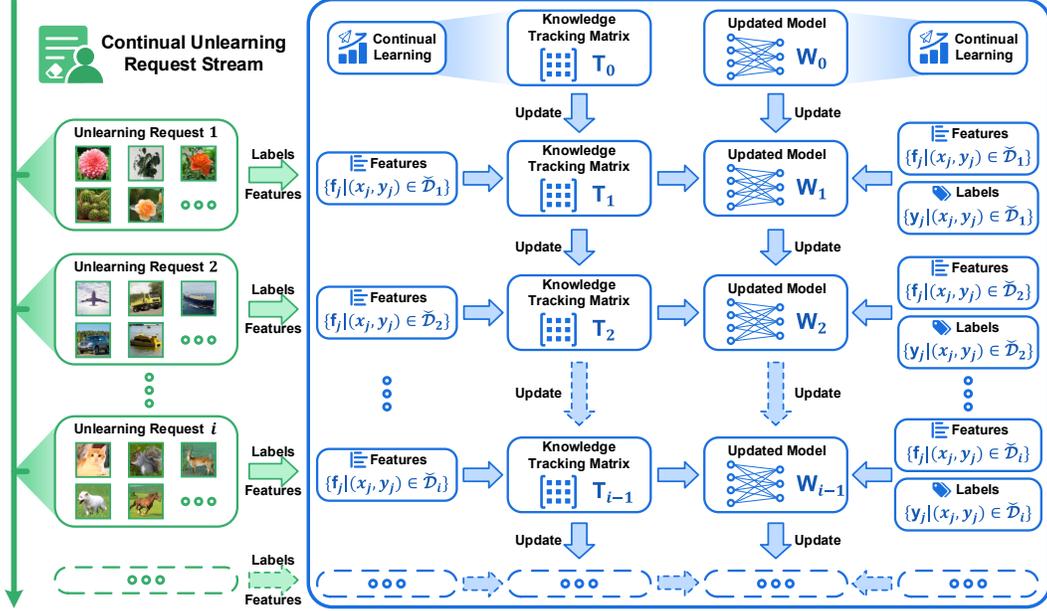
Figure 2: The framework of our proposed ACU method.

3.1 Overview

Here, we adopt image classification as a representative example to show the workflow of our ACU. Notably, our ACU is not limited to it and can be readily extended to a broader range of applications. The model within our ACU consists of two key components: a gradient-free feature extractor and a gradient-free analytic classifier. Specifically, the feature extractor is implemented using a frozen pre-trained model, which has already become a predominant strategy in the field of CL [13, 14, 15, 16]. After determining the pre-trained backbone $\text{Backbone}(\cdot, \Theta)$ and the one-hot function $\text{Onehot}(\cdot)$, the feature and one-hot label of any data sample $(x_j, y_j) \in \mathcal{D}$ are extracted as follows.

$$\mathbf{f}_j = \mathcal{G}(\text{Backbone}(x_j, \Theta)), \quad \mathbf{y}_j = \text{Onehot}(y_j). \tag{1}$$

Here, $\mathbf{f}_j \in \mathbb{R}^{1 \times d_{\text{F}}}$ and $\mathbf{y}_j \in \mathbb{R}^{1 \times d_{\text{C}}}$ represent the obtained feature vector and label vector, respectively, where d_{F} and d_{C} represents the feature dimension and the total number of classes. The pre-trained backbone serves as a feature extractor during the CL phase, and can be directly obtained from existing models available on the Internet [8, 13]. The nonlinear mapping $\mathcal{G}(\cdot)$ in (1) can be implemented using techniques such as *random projections* or *kernel functions* to enhance the representational capacity of features, a widely adopted practice in the CL community [7, 15, 16, 27]. Then, based on the extracted features and labels, we embed a linear analytic classifier following the feature extractor. For any analytic classifier $\mathbf{W}_i$ in the CU phase, the optimization objective is formulated by minimizing the Mean Squared Error (MSE) with regularization over the retained set $\hat{\mathcal{D}}_i$, as shown in (2).

$$\mathbf{W}_i = \arg\min_{\mathbf{W}} \sum_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \|\mathbf{y}_j - \mathbf{f}_j \mathbf{W}\|_{\text{F}}^2 + \gamma \|\mathbf{W}\|_{\text{F}}^2. \tag{2}$$

Specifically, when $i = 0$, we have $\hat{\mathcal{D}}_0 = \mathcal{D}$. In this case, $\mathbf{W}_0$ represents both the initial model of the CU phase and the final model of the CL phase. Despite the classification task, we deliberately adopt MSE over the commonly used Cross-Entropy (CE) in (2) to enable an analytical (i.e., closed-form) solution, which is essential for enabling gradient-free model updates. We would like to clarify that the primary advantage of the CE lies in its ability to facilitate convergence in the context of gradient-based updates, which no longer holds in the context of gradient-free updates. In addition, the MSE has been shown to yield comparable performance to the CE [52, 53], and has been widely used in CL [7, 27].

To continually forget the CL-obtained knowledge, we design a *Knowledge Tracking Matrix* to assist the CU phase, as shown in Figure 2. Specifically, upon receiving each unlearning request i, our ACU first extracts the labels and features of the corresponding forgetting set $\check{\mathcal{D}}_i$ using (1). Subsequently, we can recursively update the *Knowledge Tracking Matrix* $\mathbf{T}_i$ and the analytic model $\mathbf{W}_i$ to achieve efficient and exact forgetting with historical data privacy preservation in a gradient-free manner.

4

3.2 Analytic Knowledge Embedding

In this subsection, we focus on the analytic knowledge embedding during the CL phase, which serves as the foundation for exact forgetting in the subsequent CU phase. Specifically, according to (2), we can derive a analytical (i.e., closed-form) solution for the final model $\mathbf{W}_0 \in \mathbb{R}^{d_\mathrm{F} \times d_\mathrm{C}}$ based on the least squares method, as shown in (3). This solution embeds all the knowledge acquired during the CL phase and serves as the model initialization for the CU phase. The theoretical derivation for transforming (2) into (3) is provided in **Lemma 1** of Section 3.4.

$$\mathbf{W}_0 = (\sum\nolimits_{j:(x_j,y_j)\in\mathcal{D}} \mathbf{f}_j^\top \mathbf{f}_j + \gamma\mathbf{I})^{-1} \sum\nolimits_{j:(x_j,y_j)\in\mathcal{D}} \mathbf{f}_j^\top \mathbf{y}_j. \tag{3}$$

Here, we do not delve into the details of how to obtain the final model $\mathbf{W}_0$ during the CL phase that satisfies (3), as many state-of-the-art CL studies have already solved this problem [7, 26, 27, 30, 31]. Thanks to the analytical solution of (3), we can exactly characterize the influence of each data sample on the model, which is nearly impossible in the context of gradient-based updates. Since we cannot access the retained data during the CU phase, we design a *Knowledge Tracking Matrix* $\mathbf{T}_0 \in \mathbb{R}^{d_\mathrm{F} \times d_\mathrm{F}}$ as an auxiliary tool to track the compressed influence of all retained data samples on the model:

$$\mathbf{T}_0 = (\sum\nolimits_{j:(x_j,y_j)\in\mathcal{D}} \mathbf{f}_j^\top \mathbf{f}_j + \gamma\mathbf{I})^{-1}. \tag{4}$$

Notably, since the rank of our designed *Knowledge Tracking Matrix* is $R \le d_\mathrm{F} \ll N$, it represents a highly compressed trace of the historical examples and cannot be inverted to recover the original dataset, thereby preserving historical data privacy. Specifically, although (4) represents a final analytical expression for $\mathbf{T}_0$, it can be recursively computed during the CL phase according to the online data stream, thereby eliminating the need to centrally store all data samples $\mathcal{D}$. More details on the feasibility of the recursive computation for $\mathbf{T}_0$ during the CL phase are provided in Appendix C.

3.3 Analytic Knowledge Forgetting

In this subsection, we introduce our ACU to forget the CL knowledge under the continually arriving requests during the CU phase. Upon receiving the forgetting set $\check{\mathcal{D}}_i$ associated with each unlearning request i, we first extract the features and labels of the data samples to be forgotten using (1). Then, based on the obtained features and labels, we sequentially update the *Knowledge Tracking Matrix* $\mathbf{T}_i$ and the analytical model $\mathbf{W}_i$. Specifically, $\mathbf{T}_i$ can be updated in a recursive manner as follows.

$$\mathbf{T}_i = \mathbf{T}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{T}_{i-1}, \tag{5}$$

where $\check{\mathbf{F}}_i = [\mathbf{f}_j]_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \in \mathbb{R}^{|\check{\mathcal{D}}_i|\times d_\mathrm{F}}$ denotes the forgetting feature matrix formed by vertically stacking all feature vectors $\mathbf{f}_j$ of all samples in the forgetting set $\check{\mathcal{D}}_i$. To better demonstrate the validity of the recursive formula (5), we provide the general form of $\mathbf{T}_i$ and the detailed theoretical derivation in **Lemma 3** of Appendix A. Subsequently, leveraging the updated *Knowledge Tracking Matrix* $\mathbf{T}_i$, we aim to further update the analytical model $\mathbf{W}_i$ for exactly erasing the specified knowledge in $\check{\mathcal{D}}_i$.

Here, let's delve deeper into the scenario from an Oracle perspective to understand what happens when we conduct an unlearning request. Specifically, if we consider the optimal model that satisfies (2), the unlearning request essentially corresponds to reducing the size of the retaining set. This decremental update results in two effects: (1) the influence of the remaining samples is amplified, and (2) the influence of the forgotten samples is erased. Inspired by this insight, in our ACU, we recursively update the model $\mathbf{W}_i$ to achieve exact unlearning in an interpretable manner, as follows.

$$\mathbf{W}_i = \underbrace{(\mathbf{I} + \mathbf{T}_i\sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j)\mathbf{W}_{i-1}}_{Amplified\ Remaining\ Knowledge} - \underbrace{\mathbf{T}_i\sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j}_{Forgotten\ Knowledge}. \tag{6}$$

Here, the recursive formula (6) can be decomposed into two terms. Specifically, the former term represents amplifying the influence of the existing data by multiplying the previous model $\mathbf{W}_{i-1}$ by a factor greater than $\mathbf{I}$, and the factor used for amplification is related to the features to be forgotten. Meanwhile, the latter term explicitly reflects the process of erasing the knowledge to be forgotten. In both terms, the *Knowledge Tracking Matrix* $\mathbf{T}_i$ plays a role in adjusting knowledge. **Theorem 1** of the subsequent Section 3.4 demonstrates that the model $\mathbf{W}_i$ updated via (6) is equivalent to the re-trained model directly on the retained set $\hat{\mathcal{D}}_i$, thereby verifying its ability for exact unlearning.

Due to its gradient-free nature, the entire process of our ACU requires only forward-propagation and lightweight matrix computation, without the need for back-propagation, significantly improving system efficiency. In addition, since the historical knowledge is explicitly embedded in the analytical expression, we can maintain model fidelity without re-training or fine-tuning on the retained dataset. Moreover, the model within our ACU retains its optimal analytical form even after undergoing the CU phase, which lays the foundation for subsequent CL phases. In other words, our ACU can readily support alternating CL and CU phases, greatly expanding its practical application scope and potential. For clarity, we provide the detailed procedures of our ACU in **Algorithm 1** of the Appendix.

3.4 Theoretical Analyses

In this subsection, we thoroughly analyze the validity of our ACU. First of all, as the foundation of our gradient-free ACU, we derive the analytical (i.e., closed-form) solution of (3) in **Lemma 1** below.

Lemma 1: For any optimization problem of the following form:

$$\mathbf{W}_i = \arg\min_{\mathbf{W}} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \|\mathbf{y}_j - \mathbf{f}_j\mathbf{W}\|_{\mathrm{F}}^2 + \gamma\|\mathbf{W}\|_{\mathrm{F}}^2. \tag{7}$$

its analytical (i.e., closed-form) solution can be formulated as:

$$\mathbf{W}_i = (\sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top\mathbf{y}_j. \tag{8}$$

Proof. See Appendix A.

Then, we further prove that the model $\mathbf{W}_i$ updated recursively through our ACU by (6) is exactly equivalent to the re-trained model directly on the retained set $\hat{\mathcal{D}}_i$, as shown in **Theorem 1** below.

Theorem 1: Consider the recursive updating formula for the analytic model:

$$\mathbf{W}_i = (\mathbf{I} + \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top\mathbf{f}_j)\mathbf{W}_{i-1} - \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top\mathbf{y}_j, \tag{9}$$

where $\mathbf{T}_i$ is the *Knowledge Tracking Matrix* obtained by (5). The result of (9) is exactly equivalent to that of (8), which corresponds to the optimal re-trained model obtained from the retained dataset $\hat{\mathcal{D}}_i$.

Proof. See Appendix A.

Last but not least, we provide a comprehensive analysis of the system efficiency of our ACU in Appendix B, covering the computational and storage overhead throughout the CU phase.

4 Experiments

4.1 Experimental Setup

Datasets & Settings. In our experiments, we comprehensively analyze the performance of our ACU on two benchmark datasets: CIFAR-10 [54] and CIFAR-100 [54]. In practice, the base model, which can be an open-source pre-trained model, is incrementally updated by CL to adapt to the open-world environment. To align with real-world scenarios and avoid any overlap between the data used in the pre-training phase and the continual learning phase, we partition the full dataset into three disjoint subsets: one for obtaining the pre-trained base model (base samples), one for training the continual learning model (CL training samples), and one for testing. Subsequently, we select a subset from the CL training samples and partition it into 5 or 25 forgetting sets for CU. The specific dataset partitioning strategy and model training strategy are detailed in Appendix E.1.

Baselines & Metrics. We compare our proposed ACU with a series of state-of-the-art unlearning baselines, including Finetuning [17], L1-Sparsity [21], NegGrad [23], NegGrad+ [11], SCRUB [11], WoodFisher [55], and RandomLabel [56]. Given the time infeasibility of existing exact unlearning methods in the CU setting (e.g., taking more than 1.5 hours to process a single unlearning request), we primarily adopt approximate unlearning methods for comparison. To ensure fairness, all baselines are updated starting from the same base model for the CU requests. Additionally, we used the difference from the optimal re-trained model, considering average accuracy, Membership Inference Attacks (MIA) [57], and model parameters, to evaluate overall forgetting performance, the average accuracy on the retained set and test set to evaluate fidelity, and the cumulative runtime to evaluate efficiency. The detailed descriptions of the metrics and baselines are provided in Appendix E.2 and E.3.

Table 1: Comparisons of the methods' differences from the re-trained model regarding the model parameters (L_2 Norm), retained set accuracy, forgetting set accuracy, test set accuracy, and MIA indicator. The **bold** and underlined results indicate the best and second-best performance, respectively. All experiments are conducted three times, and the results are shown as $\text{Mean}_{\pm\text{Standered Deviation}}$.

Dataset	Method	Privacy	Δ_{Params}		Δ_{Retain}		Δ_{Forget}		Δ_{Test}		Δ_{MIA}	
			$K=5$	$K=25$	$K=5$	$K=25$	$K=5$	$K=25$	$K=5$	$K=25$	$K=5$	$K=25$
CIFAR-10	Finetuning	✗	$33.65_{\pm0.06}$	$35.45_{\pm0.95}$	$6.61_{\pm0.09}$	$6.09_{\pm4.09}$	$1.03_{\pm0.50}$	$4.19_{\pm1.53}$	$2.26_{\pm0.70}$	$4.20_{\pm1.44}$	$0.02_{\pm0.01}$	$0.04_{\pm0.02}$
	L1-Spare	✗	$\underline{24.14}_{\pm0.01}$	$\underline{25.39}_{\pm0.17}$	$16.79_{\pm0.81}$	$15.16_{\pm3.08}$	$9.88_{\pm0.74}$	$10.26_{\pm1.00}$	$10.09_{\pm0.37}$	$10.50_{\pm1.53}$	$0.49_{\pm0.28}$	$0.10_{\pm0.02}$
	SCRUB	✗	$28.12_{\pm0.00}$	$34.34_{\pm3.56}$	$\underline{1.17}_{\pm0.09}$	$15.42_{\pm5.23}$	$8.48_{\pm0.49}$	$6.85_{\pm4.54}$	$\underline{0.07}_{\pm0.05}$	$8.19_{\pm4.01}$	$0.09_{\pm0.01}$	$0.07_{\pm0.05}$
	WoodFisher	✗	$28.30_{\pm0.00}$	$57.12_{\pm0.35}$	$1.57_{\pm0.01}$	$89.58_{\pm0.07}$	$10.02_{\pm0.19}$	$79.23_{\pm0.15}$	$0.51_{\pm0.04}$	$79.23_{\pm0.00}$	$0.09_{\pm0.00}$	$\underline{0.01}_{\pm0.00}$
	RandomLabel	✗	$27.45_{\pm0.01}$	$26.75_{\pm0.90}$	$2.21_{\pm0.32}$	$\underline{3.26}_{\pm2.11}$	$\underline{0.92}_{\pm0.51}$	$\underline{2.47}_{\pm1.02}$	$2.40_{\pm0.39}$	$\underline{3.89}_{\pm0.26}$	$\underline{0.01}_{\pm0.00}$	$0.03_{\pm0.00}$
	NegGrad+	✗	$28.63_{\pm0.01}$	$55.50_{\pm9.60}$	$13.21_{\pm0.79}$	$82.32_{\pm3.61}$	$9.73_{\pm1.30}$	$71.85_{\pm3.67}$	$11.51_{\pm0.62}$	$70.91_{\pm3.89}$	$0.09_{\pm0.01}$	$0.07_{\pm0.04}$
	NegGrad	✔	$28.30_{\pm0.00}$	$28.31_{\pm0.00}$	$6.06_{\pm0.02}$	$24.85_{\pm1.30}$	$1.57_{\pm0.51}$	$14.82_{\pm1.52}$	$6.55_{\pm0.03}$	$20.20_{\pm0.66}$	$0.02_{\pm0.00}$	$0.15_{\pm0.01}$
	ACU	✔	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$
CIFAR-100	Finetuning	✗	$58.92_{\pm0.03}$	$61.80_{\pm2.94}$	$12.03_{\pm0.23}$	$15.55_{\pm2.81}$	$\underline{1.27}_{\pm0.57}$	$\underline{4.03}_{\pm4.36}$	$\underline{0.30}_{\pm0.09}$	$4.72_{\pm4.09}$	$\underline{0.02}_{\pm0.01}$	$0.04_{\pm0.04}$
	L1-Spare	✗	$\underline{44.93}_{\pm0.00}$	$\underline{45.64}_{\pm0.72}$	$29.36_{\pm1.84}$	$30.17_{\pm9.19}$	$11.85_{\pm1.03}$	$14.86_{\pm0.71}$	$13.55_{\pm1.98}$	$15.33_{\pm1.69}$	$0.12_{\pm0.02}$	$0.15_{\pm0.01}$
	SCRUB	✗	$53.72_{\pm0.00}$	$54.59_{\pm1.94}$	$0.98_{\pm0.19}$	$\underline{7.60}_{\pm0.97}$	$20.22_{\pm0.83}$	$10.65_{\pm3.74}$	$2.47_{\pm0.03}$	$\underline{2.76}_{\pm3.50}$	$0.20_{\pm0.00}$	$0.11_{\pm0.04}$
	WoodFisher	✗	$54.00_{\pm0.00}$	$54.01_{\pm0.00}$	$1.38_{\pm0.15}$	$16.04_{\pm9.15}$	$22.7_{\pm1.09}$	$11.87_{\pm7.37}$	$3.06_{\pm0.10}$	$4.50_{\pm5.50}$	$0.23_{\pm0.00}$	$0.12_{\pm0.07}$
	RandomLabel	✗	$53.77_{\pm0.01}$	$53.76_{\pm3.00}$	$7.16_{\pm0.29}$	$11.89_{\pm0.38}$	$6.42_{\pm0.51}$	$11.72_{\pm1.12}$	$2.73_{\pm0.27}$	$5.99_{\pm1.07}$	$0.63_{\pm0.00}$	$\underline{0.02}_{\pm0.01}$
	NegGrad+	✗	$53.82_{\pm0.00}$	$54.76_{\pm0.60}$	$1.09_{\pm0.10}$	$11.02_{\pm7.59}$	$9.65_{\pm0.67}$	$15.07_{\pm6.74}$	$0.62_{\pm0.35}$	$9.95_{\pm3.09}$	$0.10_{\pm0.00}$	$0.05_{\pm0.07}$
	NegGrad	✔	$54.00_{\pm0.00}$	$54.01_{\pm0.00}$	$\underline{0.38}_{\pm0.23}$	$15.53_{\pm7.77}$	$22.05_{\pm0.35}$	$12.14_{\pm7.64}$	$2.31_{\pm0.08}$	$4.10_{\pm4.47}$	$0.21_{\pm0.00}$	$0.12_{\pm0.08}$
	ACU	✔	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$	$\mathbf{00.00}_{\pm0.00}$

4.2 Overall Analysis on Unlearning Effectiveness

To comprehensively assess the unlearning effectiveness of each method, we measured their deviations from the re-trained model using several metrics: model parameters gap Δ_{Params}, retained set accuracy gap Δ_{Retain}, forgetting set accuracy gap Δ_{Forget}, test set accuracy gap Δ_{Test}, and the MIA indicator gap Δ_{MIA}. To examine the impact of different forgetting frequencies on unlearning effectiveness, we set $K = 5$ and $K = 25$ to simulate varying numbers of unlearning requests. Each request was associated with the retention of 400 data samples, ensuring control over other variables. Ideally, an effective unlearning method should precisely eliminate the influence of the forgotten data from the model. Therefore, the closer the post-unlearning model is to the corresponding re-trained model, the more effective the unlearning method is considered to be. The metric Δ_{Params} serves as a holistic indicator that quantifies the overall parameter-level difference between the unlearned and re-trained models. The metrics Δ_{Retain}, Δ_{Forget}, and Δ_{Test} measure the performance gap on the retained set, forgotten set, and test set, respectively. Meanwhile, Δ_{MIA} is also an important measure for capturing differences in output distributions between the unlearned and re-trained models in the context of privacy risk.

The experimental results on unlearning effectiveness are shown in Table 1. Thanks to its theoretical guarantee of exact unlearning, our ACU achieves optimal zero values across all metrics, indicating that the unlearned model produced by ACU is identical to the re-trained model. In contrast, due to the baselines' inherent reliance on approximate unlearning, they exhibit non-zero values across various metrics. Notably, their performance tends to be inconsistent across different metrics of evaluation. For example, on the CIFAR-100 dataset, Finetuning achieves favorable results on Δ_{Test} and Δ_{Forget}, yet fails to maintain competitive performance on Δ_{Retain}. This phenomenon is reasonable, as approximate unlearning often entails trade-offs: insufficient unlearning may leave residual influence from the data that should have been forgotten, while excessive unlearning may damage unrelated knowledge. Such imbalances lead to partial effectiveness, i.e., strong performance on certain metrics, but degradation on others. In comparison, our ACU can achieve exact unlearning without any trade-off, enabling it to maintain optimal performance across all unlearning effectiveness metrics.

It is worth emphasizing that most baselines rely on revisiting the retained data for fine-tuning during the CU phase, which essentially constitutes a form of *cheating*. Such a requirement is fundamentally incompatible with practical CU scenarios, in accordance with the CL philosophy that online task data is discarded after training. A rare baseline that does not require any access to the retained set is NegGrad, which updates the model solely by applying reverse gradient descent on the small forgetting set. However, due to the lack of access to the retained data, NegGrad exhibits relatively poor performance across various evaluation metrics. In contrast, our ACU maintains its superior performance in unlearning effectiveness without any access to historical data from the retained set. Even under such unfairly disadvantaged conditions, our ACU still achieves notable superiority in unlearning effectiveness, further highlighting the advantages of gradient-free unlearning.
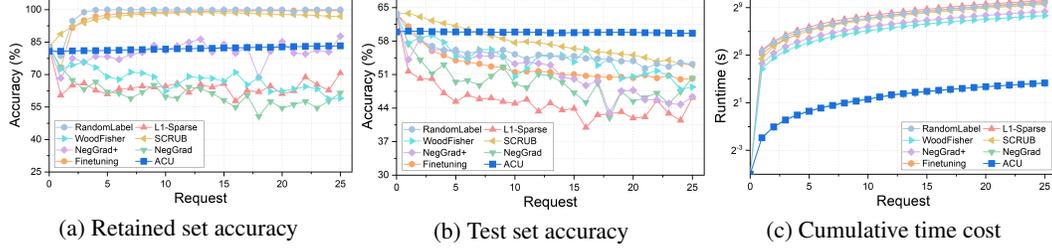
| (a) Retained set accuracy | (b) Test set accuracy | (c) Cumulative time cost |

Figure 3: The dynamics analysis for a total of 25 CU requests.

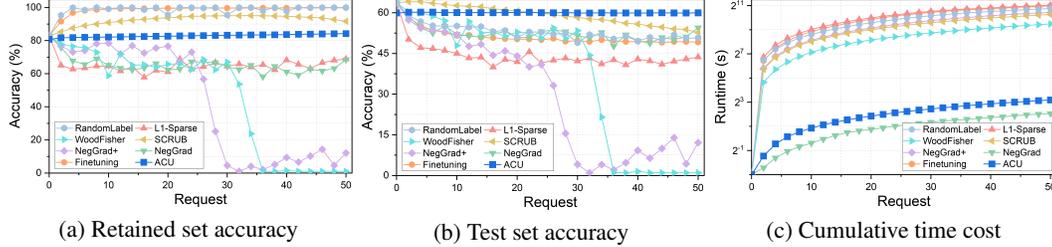| (a) Retained set accuracy | (b) Test set accuracy | (c) Cumulative time cost |

Figure 4: The dynamics analysis for a total of 50 CU requests.

4.3 Dynamics Analysis on Fidelity and Efficiency

In this subsection, we investigate the dynamics of each method during continual processing of CU requests. Taking the CIFAR-100 dataset as an example, we analyze how the fidelity and efficiency of various methods evolve. Specifically, we consider a scenario involving 25 unlearning requests corresponding to a total of 10,000 data samples to be forgotten. For each method, we record its (a) retained set accuracy, (b) test set accuracy, and (c) cumulative time cost, as illustrated in Figure 3. From Figure 3 (a), we observe that the retained set accuracy of NegGrad, L1-Sparse, and WoodFisher undergoes catastrophic degradation as CU requests accumulate. This degradation stems from the fact that these methods, during their approximate CU processes, inevitably impair the knowledge originally preserved in the retained set. In contrast, our ACU maintains relatively stable retained set accuracy throughout the CU phase. Only four other baseline methods manage to preserve or slightly improve their retained set accuracy, primarily due to fine-tuning with access to the retained data.

It is worth noting that the preservation of retained set accuracy by these baselines is achieved by violating the constraint of not accessing historical data during the CU phase, which can be considered a form of *cheating*. Moreover, such preserved accuracy of baselines on the retained set does not necessarily mean genuine retention of non-forgotten knowledge, but rather reflects mere overfitting on the retained set in fact. Figure 3 (b) provides further evidence that all the baselines exhibit a progressive and substantial decline in test set accuracy as the number of CU requests increases, despite some of them achieving relatively good retained set accuracy. It indicates that the seemingly well-preserved retained set accuracy is merely superficial, resulting from overfitting to the retained set, while the test set accuracy has already been severely compromised. These results underscore that baseline methods, when confronted with continual unlearning, suffer from a catastrophic erosion of model fidelity due to the unintended forgetting of knowledge that should have been retained. In contrast, our ACU consistently achieves high and stable accuracy on both the retained set and test set.

In addition to fidelity, we also analyze the superiority of our ACU in terms of efficiency, as illustrated in Figure 3 (c), where the Y-axis is presented on a logarithmic scale. Except for NegGrad, all other baselines incur a significantly higher total time cost than our ACU (approximately 50 to 125 times), even though these baselines are already considered relatively efficient among existing unlearning methods. The efficiency advantage of NegGrad stems from the simplicity of its method, as it updates the model solely by applying reverse gradient descent on the small forgetting set. However, its simplicity also compromises its performance in terms of unlearning effectiveness and model fidelity, as previously discussed. To provide readers with a point of reference, we also briefly discuss the impracticality of re-training from an efficiency perspective in CU. Specifically, re-training the model from scratch for each unlearning request would take approximately 1.5 hours per request (5400 seconds), resulting in a total of over 135,000 seconds for 25 requests (more than 10,000× slower than our ACU). Thus, the inefficiency renders re-training entirely infeasible for real-world applications.

Subsequently, we extend our analysis to a more challenging scenario involving a higher volume of CU requests, examining how each method performs in terms of fidelity and efficiency. Specifically, we keep the total number of samples to be forgotten fixed at 10,000 while doubling the number of unlearning requests from 25 to 50, as presented in Figure 4. From the results, we observe a notably catastrophic degradation in model fidelity for WoodFisher and NegGrad+, which was not observed in Figure 3. This observation indicates that these baselines are sensitive to the number of CU requests and may experience severe fidelity collapse when handling frequent requests. Meanwhile, in terms of efficiency, increasing the number of requests also leads to varying degrees of additional time cost for all methods. For our ACU, the cumulative time cost increases only moderately from 12.68 seconds at 25 requests to 18.16 seconds at 50 requests, representing just a 1.43× increase. By comparison, for instance, NegGrad+ sees its cumulative time cost rise sharply from 815.13 seconds to 2769.09 seconds, a surge of up to 3.40×. These results highlight that our ACU not only achieves superior efficiency in both scenarios but also shows greater robustness to the number of CU requests.

4.4 Key Takeaways

Thanks to the employment of analytical solutions instead of gradient-based updates, our ACU achieves exact unlearning, consistently attaining optimal performance across all unlearning effectiveness metrics under various settings. In contrast, other methods can only reach suboptimal performance and often face trade-offs, i.e., optimizing one metric at the expense of another. Considering the dynamics of continually processing multiple CU requests, existing baselines suffer from catastrophic degradation in model fidelity, with the extent of collapse worsening as the number of requests increases. Although some baselines appear to preserve retained set accuracy, this is typically achieved through fine-tuning with access to retained data, which violates the constraint of not accessing historical data during the CU phase, constituting a form of *cheating*. As for unlearning efficiency, our ACU also demonstrates significant superiority, achieving a 50–125× speedup over most baselines and more than a 10,000× speedup compared to re-training from scratch. Moreover, the efficiency of our ACU exhibits significantly greater robustness to increasing numbers of CU requests compared to the baselines. In summary, our ACU achieves efficient and exact forgetting while preserving data privacy. Its remarkable advantages in unlearning effectiveness, model fidelity, and system efficiency make it particularly well-suited for the CU setting. The superiority of our ACU underscores the potential of gradient-free methods for unlearning and can inspire further promising exploration in this direction.

5 Conclusion and Discussion

In this paper, we introduce a novel and practical problem, CU, which aims to sequentially forget the knowledge acquired during the CL phase. Unlike prior unlearning settings that assume a centrally trained model, CU follows the CL philosophy and operates under the more realistic constraint of having no access to the historically retained dataset. We analyze the limitations of existing unlearning methods in addressing CU and identify the root cause as their reliance on gradient-based updates. Motivated by this insight, we propose ACU in an effort to fundamentally bridge the research gap and overcome the limitations of existing methods by circumventing gradient-based updates. In response to each unlearning request, our ACU iteratively derives the analytical (i.e., closed-form) model update based on the least squares method. Through both theoretical and empirical evaluations, we validate the superiority of our proposed ACU. As an initial attempt to incorporate analytic learning into the field of unlearning, our ACU holds great potential in advancing the development of responsible AI.

The primary limitation of ACU is its reliance on a frozen pre-trained backbone for feature extraction, without involving the unlearning of knowledge acquired during the pre-training phase. This design choice is largely determined by the CU focus of this paper, which specifically aims to unlearn the knowledge acquired during the CL phase. Since the frozen pre-trained backbone can be directly obtained from publicly available sources with certified security and privacy guarantees, it is reasonable for continual learners to ignore unlearning the backbone knowledge acquired during the pre-training phase, leaving any potential modification requirements to the backbone provider. In addition, keeping the pre-trained backbone frozen aligns well with recent state-of-the-art strategies in CL and helps preserve the model fidelity. Nevertheless, it represents a promising future direction to extend our ACU by exploring an adjustable backbone, enabling the unlearning of knowledge embedded within the backbone during the CU phase. Last but not least, as unlearning embodies a key goal of responsible AI, we include a detailed discussion on the broader societal impacts of our work in Appendix D.

References

[1] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

[2] Thanh-Dat Truong, Hoang-Quan Nguyen, Bhiksha Raj, and Khoa Luu. Fairness continual learning approach to semantic scene understanding in open-world environments. *Advances in Neural Information Processing Systems*, 36:65456–65467, 2023.

[3] Yujie Li, Xin Yang, Hao Wang, Xiangkun Wang, and Tianrui Li. Learning to prompt knowledge transfer for open-world continual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13700–13708, 2024.

[4] Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. SLCA: Slow learner with classifier alignment for continual learning on a pre-trained model. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19091–19101, 2023.

[5] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 139–149, 2022.

[6] Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23554–23564, 2024.

[7] Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. RanPAC: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

[8] Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8363–8371. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Survey Track.

[9] Hai-Long Sun, Da-Wei Zhou, De-Chuan Zhan, and Han-Jia Ye. PILOT: a pre-trained model-based continual learning toolbox. *Science China Information Sciences*, 68(4):147101, 2025.

[10] Na Li, Chunyi Zhou, Yansong Gao, Hui Chen, Zhi Zhang, Boyu Kuang, and Anmin Fu. Machine unlearning: Taxonomy, metrics, applications, challenges, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2025.

[11] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Mark He Huang, Lin Geng Foo, and Jun Liu. Learning to unlearn for robust machine unlearning. In *European Conference on Computer Vision (ECCV)*, pages 202–219. Springer, 2024.

[13] Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

[14] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bart Twardowski, Joost Van De Weijer, et al. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28525–28534, 2024.

[15] Huiping Zhuang, Yizhu Chen, Di Fang, Run He, Kai Tong, Hongxin Wei, Ziqian Zeng, and Cen Chen. GACL: Exemplar-free generalized analytic continual learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., December 2024.

[16] Huiping Zhuang, Run He, Kai Tong, Ziqian Zeng, Cen Chen, and Zhiping Lin. DS-AL: A dual-stream analytic learning for exemplar-free class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17237–17244, 2024.

[17] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. In *Proc. of the 30th Network and Distributed System Security (NDSS)*, 2023.

[18] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.

[19] Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. ARCANE: An efficient architecture for exact machine unlearning. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 4006–4013. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[20] Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021.

[21] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[22] 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 *The Twelfth International Conference on Learning Representations*, 2024.

[23] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: Understanding Factors Influencing Machine Unlearning . In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319, Los Alamitos, CA, USA, June 2022. IEEE Computer Society.

[24] Kairan Zhao, Meghdad Kurmanji, George-Octavian Bărbulescu, Eleni Triantafillou, and Peter Triantafillou. What makes unlearning hard and what to do about it. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[25] Shen Lin, Xiaoyu Zhang, Chenyang Chen, Xiaofeng Chen, and Willy Susilo. ERM-KTP: Knowledge-level machine unlearning via knowledge transfer. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20147–20155, 2023.

[26] Huiping Zhuang, Zhenyu Weng, Hongxin Wei, Renchunzi Xie, Kar-Ann Toh, and Zhiping Lin. ACIL: Analytic class-incremental learning with absolute memorization and privacy protection. *Advances in Neural Information Processing Systems*, 35:11602–11614, 2022.

[27] Huiping Zhuang, Zhenyu Weng, Run He, Zhiping Lin, and Ziqian Zeng. GKEAL: Gaussian kernel embedded analytic learning for few-shot class incremental task. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7746–7755, 2023.

[28] Huiping Zhuang, Yuchen Liu, Run He, Kai Tong, Ziqian Zeng, Cen Chen, Yi Wang, and Lap-Pui Chau. F-OAL: Forward-only online analytic learning with fast training and low memory footprint in class incremental learning. *Advances in Neural Information Processing Systems*, 37:41517–41538, 2024.

[29] Xianghu Yue, Xueyi Zhang, Yiming Chen, Chengwei Zhang, Mingrui Lao, Huiping Zhuang, Xinyuan Qian, and Haizhou Li. MMAL: Multi-modal analytic learning for exemplar-free audio-visual class incremental tasks. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 2428–2437, New York, NY, USA, 2024. Association for Computing Machinery.

[30] Guannan Lai, Yujie Li, Xiangkun Wang, Tianrui Li Junbo Zhang, and Xin Yang. Order-robust class incremental learning: Graph-driven dynamic similarity grouping. In *2025 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[31] Zijian Gao, Wangwang Jia, Xingxing Zhang, Dulan Zhou, Kele Xu, Feng Dawei, Yong Dou, Xinjun Mao, and Huaimin Wang. Knowledge memorization and rumination for pre-trained model-based class-incremental learning. In *2025 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[32] Liangzu Peng, Juan Elenter, Joshua Agterberg, Alejandro Ribeiro, and Rene Vidal. TSVD: Bridging theory and practice in continual learning with pre-trained models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[33] Quyen Tran, Tung Lam Tran, Khanh Doan, Toan Tran, Dinh Phung, Khoat Than, and Trung Le. Boosting multiple views for pretrained-based continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[34] 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*, ICML'20. JMLR.org, 2020.

[35] Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, 7(2):181–194, 2025.

[36] Hongbo Zhao, Bolin Ni, Junsong Fan, Yuxi Wang, Yuntao Chen, Gaofeng Meng, and Zhaoxiang Zhang. Continual forgetting for pre-trained vision models. In *2024 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28631–28642, 2024.

[37] Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys*, 57(1):1–38, 2024.

[38] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 499–513, New York, NY, USA, 2022. Association for Computing Machinery.

[39] Sungmin Cha, Sungjun Cho, Dasol Hwang, Honglak Lee, Taesup Moon, and Moontae Lee. Learning to unlearn: Instance-wise unlearning for pre-trained classifiers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11186–11194, 2024.

[40] Romit Chatterjee, Vikram Chundawat, Ayush Tarun, Ankur Mali, and Murari Mandal. A unified framework for continual learning and machine unlearning. *arXiv preprint arXiv:2408.11374*, 2024.

[41] Ping Guo, Michael R Lyu, and NE Mastorakis. Pseudoinverse learning algorithm for feedforward neural networks. *Advances in Neural Networks and Applications*, 1(321-326), 2001.

[42] Zhenjiao Cai, Sulan Zhang, Ping Guo, Jifu Zhang, and Lihua Hu. A progressive stacking pseudoinverse learning framework via active learning in random subspaces. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(5):2822–2832, 2024.

[43] Qian Yin, Bingxin Xu, Kaiyan Zhou, and Ping Guo. Bayesian pseudoinverse learners: From uncertainty to deterministic learning. *IEEE Transactions on Cybernetics*, 52(11):12205–12216, 2022.

[44] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[45] Kar-Ann Toh. Learning from the kernel and the range space. In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2018.

[46] Xi-Zhao Wang, Tianlun Zhang, and Ran Wang. Noniterative deep learning: Incorporating restricted boltzmann machine into multilayer random weight neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(7):1299–1308, 2017.

[47] Huiping Zhuang, Zhiping Lin, Yimin Yang, and Kar-Ann Toh. An analytic formulation of convolutional neural network learning for pattern recognition. *Information Sciences*, 686:121317, 2025.

[48] Jue Wang, Ping Guo, and Yanjun Li. DensePILAE: a feature reuse pseudoinverse learning algorithm for deep stacked autoencoder. *Complex & Intelligent Systems*, pages 1–11, 2022.

[49] Huiping Zhuang, Zhiping Lin, and Kar-Ann Toh. Blockwise recursive moore–penrose inverse for network learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(5):3237–3250, 2021.

[50] Zichen Liu, Chao Du, Wee Sun Lee, and Min Lin. Locality sensitive sparse encoding for learning world models online. In *The Twelfth International Conference on Learning Representations*, 2024.

[51] Run He, Kai Tong, Di Fang, Han Sun, Ziqian Zeng, Haoran Li, Tianyi Chen, and Huiping Zhuang. AFL: A single-round analytic approach for federated leaing with pre-trained models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[52] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[53] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. In *International Conference on Learning Representations*, 2021.

[54] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[55] Sidak Pal Singh and Dan Alistarh. WoodFisher: Efficient second-order approximation for neural network compression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18098–18109. Curran Associates, Inc., 2020.

[56] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9301–9309, 2020.

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

Table 2: Description of important notations

Notations	Description
$\mathcal{D}$	The complete training dataset during the CL phase.
$\check{\mathcal{D}}_i$	The forgetting dataset corresponding to the i-th unlearning request.
$\hat{\mathcal{D}}_i$	The retained dataset after removing the first i forgetting datasets from $\mathcal{D}$.
$\mathbf{f}_j$	The feature vector for the j-th data sample.
$\mathbf{y}_j$	The label vector for the j-th data sample.
$\mathbf{W}_0$	The initial analytic model of the CU phase, i.e., the final model of the CL phase.
$\mathbf{W}_i$	The updated analytic model after processing the i-th unlearning requests.
$\hat{\mathbf{W}}_i$	The re-trained model on the retained dataset $\hat{\mathcal{D}}_i$.
$\mathbf{T}_i$	The *Knowledge Tracking Matrix* after processing the i-th unlearning requests.

Algorithm 1 Analytic Continual Unlearning

Input: $\mathbf{W}_0$, $\mathbf{T}_0$, and $\check{\mathcal{D}} = \{\check{\mathcal{D}}_1, \check{\mathcal{D}}_2, \cdots, \check{\mathcal{D}}_K\}$.

Output: The final model $\mathbf{W}_K$.

1: **for** each unlearning request i **do**
2:     Extract features $\mathbf{f}_j$ labels $\mathbf{y}_j$ of the data samples within $\check{\mathcal{D}}_i$ using (1).
3:     Update the *Knowledge Tracking Matrix* to obtain $\mathbf{T}_i$ using (5).
4:     Update the model to obtain $\mathbf{W}_i$ using (6).
5: **Return:** The final model $\mathbf{W}_K$.

A   Appendix on Theoretical Analysis of System Validity

Here, we comprehensively analyze the validity of our ACU, theoretically proving that the results obtained by our ACU are exactly equivalent to those of the re-trained model using the remained set. First, we derived the closed-form solution of the optimization problem (2) in **Lemma 1**, thereby obtaining the optimal initial model and the re-trained model.

Lemma 1: For any optimization problem of the following form:

$$\mathbf{W}_i = \arg\min_{\mathbf{W}} \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \|\mathbf{y}_j - \mathbf{f}_j\mathbf{W}\|_{\mathrm{F}}^2 + \gamma\|\mathbf{W}\|_{\mathrm{F}}^2. \tag{10}$$

its analytical (i.e., closed-form) solution can be formulated as:

$$\mathbf{W}_i = \left(\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma\mathbf{I}\right)^{-1} \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j. \tag{11}$$

Proof. First, let us focus on the influence of each pair of $\mathbf{f}_j$ and $\mathbf{y}_j$, which can be calculated as:

$$\|\mathbf{y}_j - \mathbf{f}_j\mathbf{W}\|_{\mathrm{F}}^2 = \mathrm{Tr}[(\mathbf{y}_j - \mathbf{f}_j\mathbf{W})^\top(\mathbf{y}_j - \mathbf{f}_j\mathbf{W})], \tag{12}$$

where $\mathrm{Tr}(\cdot)$ represent the trace of the matrix. Hence, the optimization function can be represented as:

$$\begin{aligned}
&\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \|\mathbf{y}_j - \mathbf{f}_j\mathbf{W}\|_{\mathrm{F}}^2 + \gamma\|\mathbf{W}\|_{\mathrm{F}}^2 \\
=&\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathrm{Tr}[(\mathbf{y}_j - \mathbf{f}_j\mathbf{W})^\top(\mathbf{y}_j - \mathbf{f}_j\mathbf{W})] + \gamma\|\mathbf{W}\|_{\mathrm{F}}^2 \\
=&\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathrm{Tr}[\mathbf{y}_j^\top\mathbf{y}_j - 2\mathbf{W}^\top\mathbf{f}_j^\top\mathbf{y}_j + \mathbf{W}^\top\mathbf{f}_j^\top\mathbf{f}_j\mathbf{W}] + \gamma\mathrm{Tr}(\mathbf{W}^\top\mathbf{W}) \\
=&\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} [\mathrm{Tr}(\mathbf{y}_j^\top\mathbf{y}_j) - 2\mathrm{Tr}(\mathbf{W}^\top\mathbf{f}_j^\top\mathbf{y}_j) + \mathrm{Tr}(\mathbf{W}^\top\mathbf{f}_j^\top\mathbf{f}_j\mathbf{W})] + \gamma\mathrm{Tr}(\mathbf{W}^\top\mathbf{W}).
\end{aligned} \tag{13}$$

Subsequently, we calculate the derivative of the optimization function, which can be calculated as:

$$\frac{\partial}{\partial\mathbf{W}}\left(\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \|\mathbf{y}_j - \mathbf{f}_j\mathbf{W}\|_{\mathrm{F}}^2 + \gamma\|\mathbf{W}\|_{\mathrm{F}}^2\right) = 2\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} (\mathbf{f}_j^\top\mathbf{f}_j\mathbf{W} - \mathbf{f}_j^\top\mathbf{y}_j) + 2\gamma\mathbf{W}. \tag{14}$$

1

To derive the analytical solution $\mathbf{W}_i$, we set the derivative to zero and obtain:

$$2 \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} (\mathbf{f}_j^\top \mathbf{f}_j \mathbf{W}_i - \mathbf{f}_j^\top \mathbf{y}_j) + 2\gamma \mathbf{W}_i = 0$$

$$\Longleftrightarrow \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j \mathbf{W}_i - \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j + \gamma \mathbf{W}_i = 0 \qquad (15)$$

$$\Longleftrightarrow (\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I}) \mathbf{W}_i = \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j.$$

Since $\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I}$ is positive-definite for $\gamma > 0$, we can further obtain:

$$\mathbf{W}_i = (\sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1} \sum_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j. \qquad (16)$$

$\blacksquare$

Second, we further prove the validity of our updating formula for the *Knowledge Tracking Matrix*. Specifically, we provide detailed proof of the Woodbury Matrix Identity in **Lemma 2**. Subsequently, based on **Lemma 2**, we present the analytical expression of the *Knowledge Tracking Matrix* and theoretically prove the validity of its updating formula (5) in **Lemma 3**.

Lemma 2 (Woodbury Matrix Identity): For matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times m}$, $\mathbf{C} \in \mathbb{R}^{m\times m}$, and $\mathbf{D} \in \mathbb{R}^{n\times n}$, if $\mathbf{A}$ and $\mathbf{C}$ are both reversible, we can derive the following expression:

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \qquad (17)$$

Proof. Firstly, we multiply the right-hand side of the equation $\mathbf{A} + \mathbf{BCD}$ by $\mathbf{A}^{-1}$:

$$(\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1} = \mathbf{I} + \mathbf{BCDA}^{-1}. \qquad (18)$$

Then we right-multiply it by $\mathbf{B}$

$$(\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1}\mathbf{B} = \mathbf{B} + \mathbf{BCDA}^{-1}\mathbf{B}. \qquad (19)$$

Since $\mathbf{C}$ is reversible, we can obtain:

$$(\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1}\mathbf{B} = \mathbf{BC}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B}). \qquad (20)$$

Therefore, we can represent $\mathbf{BC}$ as

$$\mathbf{BC} = (\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}. \qquad (21)$$

Substituting it into $\mathbf{A} + \mathbf{BCD}$, we can obtain

$$\mathbf{A} + \mathbf{BCD} = \mathbf{A} + (\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}. \qquad (22)$$

So (18) can be rewritten as

$$(\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1} = \mathbf{I} + (\mathbf{A} + \mathbf{BCD})\mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \qquad (23)$$

Then we left-multiply it by $(\mathbf{A} + \mathbf{BCV})^{-1}$:

$$\mathbf{A}^{-1} = (\mathbf{A} + \mathbf{BCD})^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \qquad (24)$$

Finally, we can transfer items to obtain

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \qquad (25)$$

$\blacksquare$

Based on **Lemma 2**, we can further theoretically prove the validity of our updating formula for the *Knowledge Tracking Matrix* (5) as follows:

Lemma 3: Consider the Recursive update formula for the *Knowledge Tracking Matrix*:

$$\mathbf{T}_i = \mathbf{T}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i \mathbf{T}_{i-1}, \qquad (26)$$

where

$$\check{\mathbf{F}}_i = [\mathbf{f}_j]_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \in \mathbb{R}^{|\check{\mathcal{D}}_i| \times d_{\mathrm{F}}}, \quad \mathbf{T}_0 = (\sum\nolimits_{j:(x_j, y_j) \in \mathcal{D}} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1}. \tag{27}$$

The detailed analytical expression for the updated result $\mathbf{T}_i$ is:

$$\mathbf{T}_i = (\sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1}. \tag{28}$$

Proof. Here, we use mathematical induction to prove this theorem in detail. Specifically, we demonstrate that the initial *Knowledge Tracking Matrix* $\mathbf{T}_0$ satisfies (28). Next, assuming any $\mathbf{T}_i$ satisfies (28), we prove that the updated *Knowledge Tracking Matrix* $\mathbf{T}_{i+1}$ derived from (26) also satisfies (28). The detailed proof process is as follows:

(1) Base Case: According to (27), the initial *Knowledge Tracking Matrix* is calculated as:

$$\mathbf{T}_0 = (\sum\nolimits_{j:(x_j, y_j) \in \mathcal{D}} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1}. \tag{29}$$

Clearly, $\mathbf{T}_0$ satisfies the form of (28), meaning that the initial *Knowledge Tracking Matrix* $\mathbf{T}_0$ satisfies the required condition.

(2) Inductive Step: For any $\mathbf{T}_i$ satisfying (28), the corresponding updated $\mathbf{T}_{i+1}$ obtained via (26) can be calculated as:

$$\mathbf{T}_{i+1} = \mathbf{T}_i + \mathbf{T}_i \check{\mathbf{F}}_{i+1}^\top (\mathbf{I} - \check{\mathbf{F}}_{i+1} \mathbf{T}_i \check{\mathbf{F}}_i^\top)^{-1} \check{\mathbf{F}}_{i+1} \mathbf{T}_i. \tag{30}$$

According to **Lemma 2**, we can obtain:

$$(\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{D})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} + \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}. \tag{31}$$

Let $\mathbf{A} = \mathbf{T}_i^{-1}$, $\mathbf{B} = (-\check{\mathbf{F}}_{i+1}^\top)$, $\mathbf{C} = \mathbf{I}$, and $\mathbf{D} = \check{\mathbf{F}}_{i+1}$, we can further obtain:

$$(\mathbf{T}_i^{-1} - \check{\mathbf{F}}_{i+1}^\top \check{\mathbf{F}}_{i+1})^{-1} = \mathbf{T}_i + \mathbf{T}_i \check{\mathbf{F}}_{i+1}^\top (\mathbf{I} - \check{\mathbf{F}}_{i+1} \mathbf{T}_i \check{\mathbf{F}}_{i+1}^\top)^{-1} \check{\mathbf{F}}_{i+1} \mathbf{T}_i. \tag{32}$$

Substituting (32) into (30), we can obtain:

$$\mathbf{T}_{i+1} = (\mathbf{T}_i^{-1} - \check{\mathbf{F}}_{i+1}^\top \check{\mathbf{F}}_{i+1})^{-1}. \tag{33}$$

According to (28), $\mathbf{T}_i$ can be calculated as:

$$\mathbf{T}_i = (\sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1}. \tag{34}$$

Since $\hat{\mathcal{D}}_{i+1}$ and $\hat{\mathcal{D}}_i$ satisfy:

$$\hat{\mathcal{D}}_{i+1} = \mathcal{D} \setminus \bigcup_{k=1}^{i+1} \check{\mathcal{D}}_k = (\mathcal{D} \setminus \bigcup_{k=1}^{i} \check{\mathcal{D}}_k) \setminus \check{\mathcal{D}}_{i+1} = \hat{\mathcal{D}}_i \setminus \check{\mathcal{D}}_{i+1}, \tag{35}$$

we can further obtain:

$$\begin{aligned}
\mathbf{T}_{i+1} &= (\mathbf{T}_i^{-1} - \check{\mathbf{F}}_{i+1}^\top \check{\mathbf{F}}_{i+1})^{-1} \\
&= (\mathbf{T}_i^{-1} - \sum\nolimits_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j)^{-1} \\
&= (\sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I} - \sum\nolimits_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j)^{-1} \\
&= (\sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_{i+1}} \mathbf{f}_j^\top \mathbf{f}_j + \gamma \mathbf{I})^{-1}.
\end{aligned} \tag{36}$$

Clearly, the $\mathbf{W}_{i+1}$ obtained via (26) also satisfies the form of (28).

(3) Conclusion: By induction, all *Knowledge Tracking Matrices* recursively updated through (26) and (27) satisfy (28), thereby theoretically proving the validity of this update formula.

$\blacksquare$

Third, based on **Lemma 1** and **Lemma 3**, we theoretically establish the validity of our ACU in **Theorem 1**, demonstrating that the model $\mathbf{W}_i$ obtained recursively via our ACU is exactly equivalent to the re-trained model $\hat{\mathbf{W}}_i$ derived from the remained set $\hat{\mathcal{D}}_i$.

Theorem 1: Consider the recursive updating formula for the analytic model:

$$\mathbf{W}_i = (\mathbf{I} + \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j)\mathbf{W}_{i-1} - \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j, \tag{37}$$

where $\mathbf{T}_i$ is the *Knowledge Tracking Matrix* obtained by (5). The result of (37) is exactly equivalent to that of (8), which corresponds to the optimal re-trained model obtained from the retained set $\hat{\mathcal{D}}_i$.

Proof. Here, we perform a series of equivalence transformations based on (8) to derive (37), thereby proving that the result obtained by our ACU is exactly equivalent to the optimal re-trained model. According to (8), the re-trained model obtained using the remained set $\hat{\mathcal{D}}_i$ is:

$$\begin{aligned}
\mathbf{W}_i &= (\sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma\mathbf{I})^{-1} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j \\
&= (\sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j + \gamma\mathbf{I})^{-1} (\sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j - \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j) \\
&= \mathbf{T}_i (\sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j - \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j) \\
&= \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j - \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j.
\end{aligned} \tag{38}$$

By intuitively comparing (37) and (43), it remains to further analyze $\mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j$, which we denote as $\mathbf{H}_i$ for clarity. According to (5), $\mathbf{H}_i$ can be further expressed as:

$$\begin{aligned}
\mathbf{H}_i &= \mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j \\
&= (\mathbf{T}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{T}_{i-1}) \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j \\
&= \mathbf{T}_{i-1} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{T}_{i-1} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j \\
&= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{T}_{i-1} \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j \\
&= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{W}_{i-1}.
\end{aligned} \tag{39}$$

Let $\mathbf{G}_i = -\check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top$, we can obtain:

$$\begin{aligned}
& \mathbf{I} - (\mathbf{I} + \mathbf{G}_i)^{-1}\mathbf{G}_i \\
=& \mathbf{I} - (\mathbf{I} + \mathbf{G}_i)^{-1}(\mathbf{I} + \mathbf{G}_i - \mathbf{I}) \\
=& \mathbf{I} - [(\mathbf{I} + \mathbf{G}_i)^{-1}(\mathbf{I} + \mathbf{G}_i) - (\mathbf{I} + \mathbf{G}_i)^{-1}\mathbf{I}] \\
=& \mathbf{I} - \mathbf{I} + (\mathbf{I} + \mathbf{G}_i)^{-1}\mathbf{I} \\
=& (\mathbf{I} + \mathbf{G}_i)^{-1}.
\end{aligned} \tag{40}$$

By substituting (40) into (39), we can obtain:

$$\begin{aligned}
\mathbf{H}_i &= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{W}_{i-1} \\
&= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} + \mathbf{G}_i)^{-1}\check{\mathbf{F}}_i\mathbf{W}_{i-1} \\
&= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top [\mathbf{I} - (\mathbf{I} + \mathbf{G}_i)^{-1}\mathbf{G}_i]\check{\mathbf{F}}_i\mathbf{W}_{i-1} \\
&= \mathbf{W}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top [\mathbf{I} - (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}(-\check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)]\check{\mathbf{F}}_i\mathbf{W}_{i-1} \\
&= \mathbf{W}_{i-1} + [\mathbf{T}_{i-1} + \mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top (\mathbf{I} - \check{\mathbf{F}}_i\mathbf{T}_{i-1}\check{\mathbf{F}}_i^\top)^{-1}\check{\mathbf{F}}_i\mathbf{T}_{i-1}]\check{\mathbf{F}}_i^\top \check{\mathbf{F}}_i\mathbf{W}_{i-1} \\
&= \mathbf{W}_{i-1} + \mathbf{T}_i\check{\mathbf{F}}_i^\top \check{\mathbf{F}}_i\mathbf{W}_{i-1}.
\end{aligned} \tag{41}$$

That is,

$$\mathbf{T}_i \sum\nolimits_{j:(x_j,y_j)\in\hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j = \mathbf{W}_{i-1} + \mathbf{T}_i\check{\mathbf{F}}_i^\top \check{\mathbf{F}}_i\mathbf{W}_{i-1}. \tag{42}$$

Since $\check{\mathbf{F}}_i$ denotes the forgetting feature matrix formed by vertically stacking all feature vectors $\mathbf{f}_j$ in the forgetting set $\check{\mathcal{D}}_i$, it follows that

$$\check{\mathbf{F}}_i^\top \check{\mathbf{F}}_i = \sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j. \tag{43}$$

Finally, by substituting (42) and (43) into (43), we can further obtain:

$$\begin{aligned}
\mathbf{W}_i &= \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_{i-1}} \mathbf{f}_j^\top \mathbf{y}_j - \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j \\
&= \mathbf{W}_{i-1} + \mathbf{T}_i \check{\mathbf{F}}_i^\top \check{\mathbf{F}}_i \mathbf{W}_{i-1} - \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j \\
&= \mathbf{W}_{i-1} + \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j \mathbf{W}_{i-1} - \mathbf{T}_i \sum\nolimits_{j:x_j \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j \\
&= (\mathbf{I} + \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \hat{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{f}_j) \mathbf{W}_{i-1} - \mathbf{T}_i \sum\nolimits_{j:(x_j, y_j) \in \check{\mathcal{D}}_i} \mathbf{f}_j^\top \mathbf{y}_j.
\end{aligned} \tag{44}$$

$\blacksquare$

B   Appendix on Theoretical Analysis of System Efficiency

Here, we comprehensively analyze the computational overhead and storage overhead of our ACU, thereby demonstrating its outstanding system efficiency. The detailed analyses are as follows:

(1) The computational overhead: For each unlearning request i, we need to execute (5) to update the *Knowledge Tracking Matrix* and execute (6) to update the model, thereby obtaining $\mathbf{T}_i$ and $\mathbf{W}_i$. Since $\check{\mathbf{F}}_i \in \mathbb{R}^{|\check{D}_i| \times d_{\mathrm{F}}}$, $\mathbf{T}_{i-1} \in \mathbb{R}^{d_{\mathrm{F}} \times d_{\mathrm{F}}}$, and $\mathbf{W}_{i-1} \in \mathbb{R}^{d_{\mathrm{F}} \times d_{\mathrm{C}}}$, the computational complexities of calculating $\mathbf{W}_i$ and $\mathbf{T}_i$ are $O(d_{\mathrm{F}}^3 + |\check{\mathcal{D}}_i| d_{\mathrm{F}}^2 + |\check{\mathcal{D}}_i|^2 d_{\mathrm{F}} + |\check{\mathcal{D}}_i|^3)$ and $O(d_{\mathrm{F}}^3 + (|\check{\mathcal{D}}_i| + d_{\mathrm{C}}) d_{\mathrm{F}}^2 + |\check{\mathcal{D}}_i| d_{\mathrm{C}} d_{\mathrm{F}})$, respectively. Therefore, the overall complexity for each unlearning request i can be calculated as $O(d_{\mathrm{F}}^3 + (|\check{\mathcal{D}}_i| + d_{\mathrm{C}})(d_{\mathrm{F}}^2 + |\check{\mathcal{D}}_i| d_{\mathrm{F}}) + |\check{\mathcal{D}}_i|^3)$.

(2) The storage overhead: When processing continuously arriving unlearning requests, our ACU only needs to store the current model and the *Knowledge Tracking Matrix*, without requiring access to or storage of any samples from the remained set. Since $\mathbf{W}_i \in \mathbb{R}^{d_{\mathrm{F}} \times d_{\mathrm{C}}}$ $\mathbf{T}_i \in \mathbb{R}^{d_{\mathrm{F}} \times d_{\mathrm{F}}}$, the overall storage complexity can be calculated as $O(d_{\mathrm{F}}^2 + d_{\mathrm{C}} d_{\mathrm{F}})$.

C   Appendix on Recursive Computation for CL Phase

Here, we demonstrate how to recursively compute the *Knowledge Tracking Matrix* during the CL phase according to the continual learning request stream, without storing any historical samples. For clarity, the continually arriving learning sets within the CL phase are denoted as $\{\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_2, \cdots, \tilde{\mathcal{D}}_M\}$, where $\tilde{\mathcal{D}}_p$ represents the learning set associated with the learning request p and $\mathcal{D} = \bigcup_{p=1}^M \tilde{\mathcal{D}}_p$. For each learning request p, we define the corresponding updated *Knowledge Tracking Matrix* as $\mathbf{T}_{p-M}$, and the final $\mathbf{T}_{p-M}$ within the CL phase is exactly the initial $\mathbf{T}_0$ within the CU phase.

Upon receiving each learning set $\tilde{\mathcal{D}}_p$, we first extract the features to obtain $\{\mathbf{f}_j | (x_j, y_j) \in \tilde{\mathcal{D}}_p\}$ using (1). Subsequently, analogous to the recursive update formula within the CU phase (5), the *Knowledge Tracking Matrix* $\mathbf{T}_{p-1-M}$ can be updated in a recursive manner as follows:

$$\mathbf{T}_{p-M} = \mathbf{T}_{p-M-1} - \mathbf{T}_{p-M-1} \tilde{\mathbf{F}}_i^\top (\mathbf{I} + \tilde{\mathbf{F}}_i \mathbf{T}_{p-M-1} \tilde{\mathbf{F}}_i^\top)^{-1} \tilde{\mathbf{F}}_i \mathbf{T}_{p-M-1}, \tag{45}$$

where $\tilde{\mathbf{F}}_p = [\mathbf{f}_j]_{j:(x_j, y_j) \in \tilde{\mathcal{D}}_p} \in \mathbb{R}^{|\tilde{\mathcal{D}}_p| \times d_{\mathrm{F}}}$ represents the learning feature matrix formed by vertically stacking $\{\mathbf{f}_j | (x_j, y_j) \in \tilde{\mathcal{D}}_p\}$ in the learning set $\tilde{\mathcal{D}}_i$. Furthermore, as a special case, the initial $\mathbf{T}_{-M}$ is computed as $\mathbf{T}_{-M} = (\gamma \mathbf{I}_{d_{\mathrm{F}} \times d_{\mathrm{F}}})^{-1}$. Next, by recursively executing (45) across all learning requests, we obtain the final $\mathbf{T}_0$ within the CL phase without storing any historical data.

Finally, in **Lemma 4**, we prove that the final result $\mathbf{T}_0$ exactly satisfies (48), and further demonstrate in detail that the analytical expression of each resulting $\mathbf{T}_{p-M}$ coincides exactly with that obtained using (5) in the CU phase. Consequently, the *Knowledge Tracking Matrix* can be recursively updated using (45) for learning requests and (5) for unlearning requests, thereby enabling seamless integration with both CL and CU phases.

Lemma 4: Consider the recursive update formula for the *Knowledge Tracking Matrix*:

$$\mathbf{T}_{p-M} = \mathbf{T}_{p-1-M} - \mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top(\mathbf{I} + \tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top)^{-1}\tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M}, \tag{46}$$

where

$$\tilde{\mathbf{F}}_p = [\mathbf{f}_j]_{j:(x_j,y_j)\in\tilde{\mathcal{D}}_p} \in \mathbb{R}^{|\tilde{\mathcal{D}}_p|\times d_\mathrm{F}}, \quad \mathbf{T}_{-M} = (\gamma\mathbf{I}_{d_\mathrm{F}\times d_\mathrm{F}})^{-1}. \tag{47}$$

The resulting final $\mathbf{T}_0$ satisfies (4). More specifically, the detailed analytical expressions for each updated result $\mathbf{T}_{p-M}$ are as follows:

$$\mathbf{T}_{p-M} = (\sum\nolimits_{j:(x_j,y_j)\in\bigcup_{k=1}^p \tilde{\mathcal{D}}_k} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1}. \tag{48}$$

Proof. Here, we provide the detailed proof using mathematical induction. Specifically, we first prove that the initial $\mathbf{T}_{-M}$, i.e., the base case, satisfies (48). Subsequently, we demonstrate that for any $\mathbf{T}_{p-1-M}$ satisfying (48), the corresponding updated $\mathbf{T}_{p-M}$ obtained through (46) also satisfies (48). Finally, we prove that the final $\mathbf{T}_0$ obtained using (46) in the CL phase satisfies (4).

(1) Base Case: According to (47), the initial $\mathbf{T}_{-M}$ is calculated as:

$$\mathbf{T}_{-M} = (\gamma\mathbf{I}_{d_\mathrm{F}\times d_\mathrm{F}})^{-1} = (\sum\nolimits_{j:(x_j,y_j)\in\emptyset} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1}. \tag{49}$$

Clearly, $\mathbf{T}_{-M}$ satisfies the form of (48), meaning that the base case satisfies the required condition.

(2) Inductive Step: By substituting $\mathbf{A} = \mathbf{T}_{p-1-M}^{-1}$, $\mathbf{B} = \tilde{\mathbf{F}}_p^\top$, $\mathbf{C} = \mathbf{I}$, and $\mathbf{D} = \tilde{\mathbf{F}}_p$ into (17) of **Lemma 2**, we can obtain:

$$(\mathbf{T}_{p-1-M}^{-1} + \tilde{\mathbf{F}}_p^\top\tilde{\mathbf{F}}_p)^{-1} = \mathbf{T}_{p-1-M} - \mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top(\mathbf{I} + \tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top)^{-1}\tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M}. \tag{50}$$

Therefore, for any $\mathbf{T}_{p-1-M}$ satisfying (48), the corresponding updated $\mathbf{T}_{p-M}$ obtained through (46) can be calculated as:

$$\begin{aligned}\mathbf{T}_{p-M} &= \mathbf{T}_{p-1-M} - \mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top(\mathbf{I} + \tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M}\tilde{\mathbf{F}}_p^\top)^{-1}\tilde{\mathbf{F}}_p\mathbf{T}_{p-1-M} \\ &= (\mathbf{T}_{p-1-M}^{-1} + \tilde{\mathbf{F}}_p^\top\tilde{\mathbf{F}}_p)^{-1}.\end{aligned} \tag{51}$$

Substituting (48) into (51), we can obtain:

$$\begin{aligned}\mathbf{T}_{p-M} &= (\sum\nolimits_{j:(x_j,y_j)\in\bigcup_{k=1}^{p-1} \tilde{\mathcal{D}}_k} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I} + \tilde{\mathbf{F}}_p^\top\tilde{\mathbf{F}}_p)^{-1} \\ &= (\sum\nolimits_{j:(x_j,y_j)\in\bigcup_{k=1}^p \tilde{\mathcal{D}}_k} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1}.\end{aligned} \tag{52}$$

Clearly, $\mathbf{T}_{p-M}$ also satisfies (48), meaning that the inductive step satisfies the required condition.

(3) Conclusion: By induction, all *Knowledge Tracking Matrices* recursively updated through (46) and (47) can be represented as (49). Since $\mathcal{D} = \bigcup_{p=1}^M \tilde{\mathcal{D}}_p$, the final $\mathbf{T}_0$ obtained through the CL phase can be calculated as:

$$\mathbf{T}_0 = \mathbf{T}_{M-M} = (\sum\nolimits_{j:(x_j,y_j)\in\bigcup_{k=1}^M \tilde{\mathcal{D}}_k} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1} = (\sum\nolimits_{j:(x_j,y_j)\in\mathcal{D}} \mathbf{f}_j^\top\mathbf{f}_j + \gamma\mathbf{I})^{-1}, \tag{53}$$

which satisfies (4) and proves the validity of our recursive update formula for the *Knowledge Tracking Matrix* during the CL phase.

∎

D Appendix on Broader Impact

Our work on CU advances the broader objective of responsible AI by enabling the removal of erroneous, biased, or privacy-sensitive information from continually trained models, without compromising the historical data privacy. By achieving exact and efficient unlearning while preserving privacy, our ACU addresses a critical gap in the CU landscape. As such, we do not anticipate any direct negative societal impact arising from our work. Nevertheless, since our method leverages open-source pre-trained models as feature extractors does not unlearn the knowledge acquired during the pre-training phase, careful selection of compliant and trustworthy pre-trained models is essential.

E Appendix for Experimental Details

E.1 Details on Datasets & Settings

Here, we provide a detailed description of the dataset settings used in our experiments. Each dataset is partitioned into the base samples for training the pre-trained base model, the CL training samples for the CL phase, and the test samples for testing. The CL training samples have a portion of the training samples further selected as forgetting samples for CU. Specifically, both CIFAR-10 and CIFAR-100 contain a total of 50,000 samples in their training sets, and 10,000 samples in their test sets. We select 10,000 samples as base samples for training the pre-trained base model and utilize the remaining 40,000 samples for CL training. Subsequently, we select 10,000 samples from these 40,000 training samples as forgetting samples, and partition them into 5 and 25 forgetting sets, respectively, to simulate different CU scenarios. Additionally, all the training is applied to the data augmentations, including the random cropping, random horizontal flipping. Specifically, both the pre-trained base model and the original model are trained for 300 epochs using the SGD optimizer with the Cosine half-cycle decay to adjust the learning rate. Specifically, the training process is with the learning rate initialized at 0.5, a weight decay of 5×10^{-4}, a momentum of 0.9, and the batch size of 256. All of the training is conducted on 6 Nvidia RTX 4090D GPUs with 90 vCPUs Intel(R) Xeon(R).

E.2 Details on Evaluation Metrics

Here, let's focus on the evaluation metrics used in our experiment. Specifically, to comprehensively analyze the superior performance of our ACU, we select multiple primary metrics: average accuracy, MIA indicator, model parameters, and cumulative runtime. These metrics are detailed as follows:

- **Average Accuracy**: We measure the unlearned model's accuracy across distinct datasets to evaluate the unlearning performance, as detailed below:

  - *Test Set Accuracy*: This metric measures the unlearned model's performance on unseen samples, reflecting its actual performance. Formally, the test set accuracy is calculated as the average accuracy across the test set.
  - *Retained Set Accuracy*: This metric measures the unlearned model's performance on the training samples that should be retained, indicating the extent to which the CU compromises the model's fidelity. Formally, the retained set accuracy is calculated as the average accuracy across the corresponding retained set $\hat{\mathcal{D}}_i$.
  - *Forgetting Set Accuracy*: This metric measures the unlearned model's performance on the training samples that should be forgotten, indicating the extent to which the unlearning request has been fulfilled. Formally, the forgetting set accuracy is calculated as the average accuracy across all previously seen forgetting sets $\bigcup_{k=1}^{i} \tilde{\mathcal{D}}_k$.

  To more clearly illustrate the unlearning effectiveness, we also calculated the gaps in test set accuracy, retained set accuracy, and forgetting set accuracy between each method and the optimal re-trained model, denoted as Δ_{Retain}, Δ_{Forget}, and Δ_{Test}, respectively.

- **MIA Indicator Gap**: The MIA indicator [57] is widely employed to evaluate the effectiveness of unlearning. Specifically, the binary MIA classifier is trained to determine whether a given sample was used during model training, using both the retained and test sets. The MIA indicator gap is computed as the average accuracy of this MIA classifier across all previously seen forgetting sets. Notably, due to the binary nature of the classifier, the MIA indicator gap is not inherently better when lower, as an attacker could simply invert the binary classifier to achieve high accuracy, thereby raising additional privacy concerns. Moreover, since samples in the forgetting sets have never been exposed to the re-trained model, the accuracy of MIA on this model represents the optimal result. Consequently, we adopt the MIA indicator gap difference as our metric to evaluate the performance of each method, denoted as Δ_{MIA}.

- **Model Parameters Gap**: More directly, we employ the model parameters gap to analyze unlearning performance, which is calculated as the normalized difference between the unlearned model parameters and those of the re-trained model, and denoted as Δ_{Params}.

- **Cumulative Runtime**: To analyze unlearning efficiency, we measure the cumulative runtime (seconds) of all methods for processing all unlearning requests.

E.3 Details on Baselines

For the details of the machine unlearning, we provide a detailed description of the baselines used in our experiments, along with their corresponding settings. Specifically, each selected baseline is outlined as follows:

- **Finetune** [17]: This method leverages the phenomenon of catastrophic forgetting by simply fine-tuning the model on the retained set, thereby diminishing the influence of the forgetting set.
- **L1-Sparsity** [21]: This approach introduces weight sparsity into the unlearning process. The model is fine-tuned on the retained set with an L1 regularization term.
- **NegGrad** [23]: Based on the intuitive idea of forgetting by increasing error, this method forces the model to maximize the loss on the forgetting set. However, this may significantly impair the overall model performance.
- **NegGrad+** [11]: This method combines the principles of **Finetune** and **NegGrad**, aiming to balance forgetting and performance retention by simultaneously maximizing the loss on the forget set while minimizing it on the retain set.
- **SCRUB** [11]: Extending the **NegGrad+** framework, this method recasts unlearning as a student-teacher distillation problem. It minimizes the distributional discrepancy between the student and teacher models on the retained set, while maximizing it on the forgetting set.
- **WoodFisher** [55]: This method aims to achieve (ϵ, δ)-forgetting by estimating the influence of individual parameters and removing their contributions to the forgetting data.

The training configurations for each baseline are summarized as follows:

- **Finetune**: For each unlearning request, we trained the model on the retained set for 10 epochs with a learning rate of 1×10^{-2}.
- **L1-Sparsity**: For each unlearning request, we trained the model on the retained set for 10 epochs with a learning rate of 1×10^{-2} and a sparsity-promoting regularization parameter of 1×10^{-3}.
- **NegGrad**: For each unlearning request, we trained the model on the forgetting set for 5 epochs with a learning rate of 1×10^{-3}.
- **NegGrad+**: For each unlearning request, we trained the model on the forgetting set and retained set for 10 epochs with a learning rate of 1×10^{-3} and a weighting factor of 0.95 to balance the two objects in the loss function.
- **SCRUB**: For each unlearning request, we trained the model on forgetting set and retained set for 5 epochs with a learning rate of 1×10^{-3}, a temperature scale of 4, a max-steps of 4, an α of 0.5, and γ of 0.5.
- **WoodFisher**: For each unlearning request, we updated the model weight by set the parameter α for Woodfisher Hessian Inverse approximation with 0.2.
- **RandomLabel**: For each unlearning request, we trained the model on the relabeled forgetting set and the retained set for 10 epochs with a learning rate of 1×10^{-3}.

In our experiments, the hyperparameter settings as above refer to the relevant settings of the original paper, and make some minor adjustments to adapt to the proposed CU scenario.