information

MDPI

Article

A Unified Knowledge Management Framework for Continual Learning and Machine Unlearning in Large Language Models

Jiaqi Lang [1,2], Linjing Li [1,2,*] and Dajun Zeng [1,2]

[1] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[2] State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: linjing.li@ia.ac.cn

Abstract

Large language models (LLMs) are increasingly deployed as information systems that evolve over time, where managing internal knowledge—acquisition, retention, and removal—becomes essential. In practice, these processes are primarily realized through continual learning and machine unlearning mechanisms. Despite this, these two mechanisms are often studied in isolation, limiting both interpretability and controllability. In this work, we present a parameter-efficient knowledge management framework where continual learning and machine unlearning—despite employing distinct task-specific objectives—are integrated through a shared retention-controlled parameter evolution mechanism. We ground these structural constraints in a drift-aware design principle: under a model smoothness assumption, we establish a formal upper bound showing that Kullback–Leibler (KL) divergence on retained knowledge is controlled by the magnitude and direction of parameter updates, providing a principled rationale for combining Low-Rank Adaptation (LoRA) freezing, sparse masking, and orthogonal gradient projection into a unified constraint system. Experiments on the Task of Fictitious Unlearning (TOFU) benchmark and real-world benchmarks demonstrate effective knowledge acquisition, selective removal, and robust retention across sequential tasks with strong overall performance and stability. This work provides a practical parameter-efficient recipe and a drift-aware design principle validated on controlled interleaved benchmarks, offering insights toward reliable knowledge management in evolving deployment scenarios.

Keywords: large language models; continual learning; machine unlearning; knowledge management; information-theoretic modeling

Check for updates

Academic Editor: Vincenzo Manca

Received: 22 January 2026
Revised: 24 February 2026
Accepted: 25 February 2026
Published: 1 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Large language models (LLMs) are increasingly deployed in real-world applications such as conversational agents, decision support systems, and personalized assistants, where they interact with evolving data streams, user feedback, and regulatory requirements. In such settings, managing the internal knowledge of LLMs—how knowledge is acquired, retained, and removed over time—has emerged as a critical challenge for the reliable and responsible deployment of large-scale Artificial Intelligence (AI) systems [1,2]. In this work, we use "knowledge management" to refer to the explicit organization and control of these acquisition, retention, and removal processes across the entire lifecycle of a deployed LLM, linking low-level parameter updates to high-level requirements such as controlled adaptation, domain shift handling, and desired model behavior.

In practice, the acquisition and removal of knowledge in neural models are primarily addressed through two research paradigms: continual learning and machine unlearning. Continual learning aims to enable models to incrementally acquire new knowledge from sequential tasks or data distributions without catastrophically degrading previously learned capabilities [3,4]. In contrast, machine unlearning focuses on the controlled removal of specific data or knowledge from a trained model, often motivated by privacy regulations, data ownership concerns, or error correction [2,5]. Despite addressing complementary aspects of the knowledge lifecycle, these two paradigms have largely been studied in isolation, with distinct objectives, evaluation protocols, and algorithmic designs.

The joint treatment of these two paradigms is captured by the Continual Learning and Unlearning (CLU) framework [6,7], which models the knowledge lifecycle of deployed systems through interleaved sequences of learning and unlearning operations. Critically, machine unlearning within the CLU framework extends beyond mere data removal or output suppression: it requires targeted modifications to the model's deep structural parameters—including weight matrices and their low-rank factorizations—in order to selectively eliminate encoded knowledge while preserving the parametric structures responsible for retained capabilities. This parameter-level perspective highlights that effective unlearning must navigate the entangled representations in deep neural networks, where knowledge is distributed across interconnected layers and cannot be simply excised without risking collateral degradation of other learned functionalities.

This separation obscures the intrinsic relationship between learning and unlearning as two directions of knowledge dynamics within a single model. From a knowledge management perspective, both continual learning and machine unlearning operate on the same underlying knowledge space, differing only in whether the objective is to incorporate or remove information. Treating them as unrelated problems limits our theoretical understanding of how model knowledge evolves under sequential updates and hinders the development of unified mechanisms for controlling model behavior. In particular, the lack of a common formulation makes it difficult to reason about trade-offs between knowledge retention, adaptability, and selective forgetting in scenarios involving interleaved task sequences.

While recent studies have explored unified perspectives on CLU through gradient-based optimization in small-scale discriminative models [7,8], these approaches primarily focus on model-centric optimization. In contrast, we propose a fundamentally different perspective by framing CLU as a knowledge management problem, where the central objective is to systematically control what knowledge is retained, acquired, and removed throughout a model's lifecycle. From this knowledge-centric view, we develop a practical framework grounded in drift-aware design principles for large-scale generative language models. While continual learning and machine unlearning employ distinct task-specific objectives (supervised fine-tuning and gradient ascent, respectively), we integrate them through a shared retention-controlled parameter evolution mechanism. Specifically, we use Kullback–Leibler (KL) divergence as a design principle to characterize distributional drift on retained knowledge, and derive parameter-space structural constraints that provably bound this drift. This enables drift-aware parameter-space approximations that govern stability–plasticity trade-offs without requiring explicit distributional measurements, offering both conceptual clarity and practical scalability for large-scale deployments.

Compared with prior CLU formulations that are predominantly model-centric and optimization-driven, our framework is explicitly knowledge-centric and drift-aware. The unification is operational rather than loss-level: learning and unlearning are governed by the same retention-controlled parameter constraints and drift-control principle, organized around information-theoretic distributional drift rather than task labels or gradient-based heuristics.

Based on the drift-aware conceptual framework, we operationalize the KL-minimization objective through a suite of parameter-space approximations. These structural choices serve as computationally efficient proxies for Equation (4), enabling controlled knowledge evolution without explicit distributional measurements or base model modification. Our approach leverages low-rank adaptation to localize knowledge updates, while combining parameter freezing, sparsity constraints, and orthogonal gradient projection to structurally constrain parameter updates and suppress interference with retained knowledge. By grounding these structural choices in the principle of controlling distributional drift—where KL divergence serves as the conceptual characterization and design motivation—we obtain a parameter-space drift control approximation that operates without explicit distributional KL computation. This KL-inspired parameter-space strategy enables scalable and incremental learning and unlearning operations suitable for large-scale models in interleaved update scenarios.

The main contributions of this work are summarized as follows:

- We develop a parameter-efficient CLU method that combines Low-Rank Adaptation (LoRA) [9] freezing, magnitude-based sparse masking, and orthogonal gradient projection into a unified structural constraint system, achieving state-of-the-art stability–plasticity balance across interleaved learning-unlearning sequences on 4B- and 8B-scale LLMs.
- We ground these structural choices in a drift-aware design principle based on KL divergence, establishing a formal upper bound (Theorem A1) that decomposes distributional drift into update magnitude and direction terms. This provides a principled explanation for why magnitude-controlling constraints (freezing, sparsity) yield the largest individual gains, while direction control (orthogonal projection) provides crucial cumulative-drift mitigation in longer sequences.
- We provide systematic experimental evidence including behavioral metrics, token-level distributional drift analysis, and ablation studies that jointly validate the method's effectiveness and the design principle's explanatory power on controlled interleaved CLU benchmarks.

2. Related Work

2.1. Continual Learning

The primary challenge of continual learning for intelligent systems lies in enabling models to acquire new knowledge while retaining previously learned knowledge under a sequential task setting. This requires finding an optimal trade-off between plasticity (the ability to learn new knowledge) and stability (the ability to preserve old knowledge), so as to mitigate the problem of catastrophic forgetting during continual learning.

Existing mainstream continual learning methods can be broadly categorized into the following three classes:

2.1.1. Regularization-Based Methods

Regularization-based methods preserve previously learned knowledge by explicitly introducing regularization terms into the loss function, thereby constraining parameter updates during training on new tasks. Specifically, these methods limit changes to parameters that are deemed important for previous tasks. A key challenge lies in how to quantify the importance of each parameter.

Elastic Weight Consolidation (EWC) [10] estimates parameter importance using the Fisher Information Matrix (FIM), leveraging second-order statistics of the loss with respect to model parameters to identify those critical to past tasks. Memory Aware Synapses (MAS) [11] measures parameter importance based on the sensitivity of the model's output

L2 norm to parameter perturbations. Synaptic Intelligence (SI) [12] tracks parameter updates throughout training and evaluates their contribution to the loss reduction to compute importance scores. Riemannian Walk (RWalk) [13] combines the advantages of EWC and SI by introducing concepts from information geometry, modeling the curvature of different tasks in parameter space through a Riemannian metric.

2.1.2. Replay-Based Methods

Replay-based methods mitigate forgetting by maintaining a representative subset of past data, often referred to as a coreset, to preserve data distributional characteristics [14,15]. The process of selecting such representative samples is known as coreset selection. Since finding an optimal subset is an NP-hard problem, early approaches relied on heuristic strategies to approximate the original data distribution.

More recent studies propose generating representative samples through optimization rather than selecting them directly from the original dataset [16,17]. These approaches, commonly referred to as dataset distillation or dataset condensation, aim to compress large-scale datasets into a compact set of synthetic samples that retain the essential information of the original data.

2.1.3. Structure-Based Methods

While regularization-based and replay-based approaches update knowledge within a shared parameter space, structure-based methods allocate task-specific parameter subspaces for incremental learning [18,19]. During inference, only the neurons, parameters, or network branches associated with the relevant task are activated. Because parameters across tasks are isolated, these methods typically require a task identification step at inference time to determine which task a given input belongs to before invoking the corresponding parameters or modules.

2.2. Machine Unlearning

The goal of machine unlearning is to remove the influence of specific data from a trained model without significantly degrading its overall performance. Existing research can be broadly categorized based on the level of intervention into the following two paradigms:

2.2.1. Removal-Intended Methods

Removal-intended methods aim to negate the effect of the data to be forgotten by modifying the training process. Gradient Ascent (GA)-based approaches achieve unlearning by applying reversed gradients or selectively fine-tuning on targeted sample sets [20,21]. Variants such as Negative Preference Optimization (NPO) and second-order methods [22,23] further improve optimization stability by incorporating divergence-based loss functions or curvature information.

2.2.2. Suppression-Intended Methods

Suppression-intended methods focus on restricting the model's access to the forgotten information rather than fully retraining the model. Full-parameter approaches include fine-grained probability adjustment [24,25], rejection fine-tuning [26,27], and incorrect label construction [28,29]. These methods weaken the influence of forgotten data by adjusting output confidence or disrupting label consistency.

2.3. Continual Learning and Machine Unlearning

Existing research on CLU has primarily focused on small-scale models in the image domain, with classification tasks as the dominant setting [6–8,30]. While these studies

have made valuable progress in integrating continual learning and unlearning, their scope remains limited to traditional discriminative models.

For example, ref. [30] introduced the CLU concept into image classification for the first time by adaptively enhancing model plasticity through selective parameter degradation. Work in [6] proposed a complete CLU formalization framework but treated entire tasks as the minimal unlearning unit, which fails to support fine-grained unlearning requirements. Study [7] unified learning and unlearning in classification tasks through a dual-teacher distillation mechanism, albeit at the cost of substantial computational and storage overhead.

We systematically study CLU under parameter-efficient constraints in generative LLMs with an interleaved learning–unlearning protocol. Our framework operationalizes KL-based CLU objectives in a knowledge-centric manner and provides a practical recipe for controlled knowledge evolution in large-scale generative models.

3. Materials and Methods

3.1. Problem Definition

We study the problem of CLU in a parametric model with parameters θ, which sequentially receives a stream of T task requests, where T is the total number of tasks. Each task request isdenoted as

$$T_t = (D_t, R_t), \tag{1}$$

where $t \in \{1, 2, \ldots, T\}$ is the task index, D_t is the dataset for task t, and R_t denotes the request type. The request type can be either learning or unlearning, i.e., $R_t \in \{L, U\}$. The dataset $D_t = \{q_i\}_{i=1}^{N_t}$ consists of N_t data points. Each data point $q_i = (x_i, y_i)$ is composed of a prompt x_i and its corresponding reference response y_i. These data points are used either for model learning or for unlearning, depending on the request type. For continual learning tasks, we denote the request as T_t^L with the corresponding dataset D_t^L. For unlearning tasks, we denote the request as T_t^U with the corresponding dataset D_t^U. Figure 1 illustrates the overall framework of the CLU paradigm, depicting how the model alternately processes learning and unlearning requests in a sequential task stream.

For continual learning, we follow prior work on large-scale model adaptation and adopt Supervised Fine-Tuning (SFT), enabling the model to incrementally acquire new and previously unseen knowledge. For continual unlearning, we adopt established unlearning paradigms, aiming to make the model effectively "forget" specified data or knowledge fragments while preserving the stability of its existing knowledge structure.

Specifically, when the model receives a request, the objective is to update the model parameters from θ_t to θ_{t+1} such that the following three core constraints are satisfied: **Forgetting Constraint**: The model must reduce its tendency to recall or reproduce information from the unlearning dataset D_t^U under the specified evaluation protocol, achieving observable behavioral redirection while minimizing collateral degradation. **Retention Constraint**: The model must preserve its performance on retain data D_t^R that is disjoint from the unlearning target, preventing negative interference with previously acquired knowledge. **Acquisition Constraint**: The model must maintain its plasticity for learning future task data D_t^L, ensuring that the unlearning operation does not compromise its capacity for subsequent knowledge acquisition in the continual learning paradigm.

Overall, the goal of the model over the entire task stream is to achieve a balanced trade-off between learning new knowledge and forgetting obsolete or sensitive information by alternately executing learning and unlearning tasks, enabling controllable model knowledge evolution.

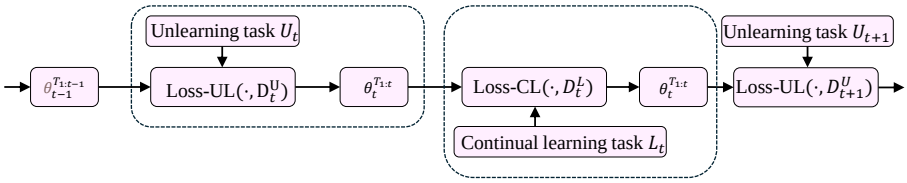


Figure 1. Overview of the CLU framework. The model sequentially processes a stream of tasks $\{T_1, T_2, \ldots, T_T\}$, where each task can be either a learning request ($R_t = L$) or an unlearning request ($R_t = U$). Through alternating learning and unlearning operations, the model parameters evolve from θ_0 to θ_T, achieving dynamic knowledge management while satisfying forgetting, retention, and acquisition constraints.

3.2. A Drift-Aware Framework for Retention-Controlled CLU

Continual learning and machine unlearning can be regarded as two canonical special cases within a unified CLU framework. Both scenarios correspond to an idealized model that provides a theoretical upper bound on achievable performance. In the continual learning setting, the jointly trained model is commonly treated as the optimal solution [31]. By aggregating all available data and removing the constraint of historical data inaccessibility, joint training minimizes the global empirical risk, thereby representing the optimal performance attainable by continual learning algorithms. In contrast, in the machine unlearning setting, the optimal model is defined as the model obtained by retraining from scratch after completely removing all data requested to be forgotten. Although this approach guarantees exact unlearning, it relies on full retraining over the remaining dataset, resulting in prohibitive computational costs [32] and rendering it impractical for real-world applications.

Accordingly, in the joint continual learning and machine unlearning problem, we define a theoretical ideal model as the optimal solution obtained by training on the union of all learning data while excluding all data subject to unlearning requests. Formally, this ideal model is defined as

$$\theta^\star = \arg\min L(\theta), \tag{2}$$

where $L(\theta)$ denotes the training loss evaluated on the dataset after removing all samples that need to be forgotten. However, directly obtaining a model whose parameter distribution exactly matches that of the ideal model is generally infeasible in practice. For example, joint retraining requires storing all historical data and performing full model retraining, which incurs excessive computational and storage overhead.

Therefore, in this work, we treat the ideal model as a theoretical reference rather than a directly attainable baseline. Based on this observation, we propose a more practical optimization paradigm, termed approximate CLU. When new learning or machine unlearning requests arrive, the system performs drift-aware updates that balance the new task objective with controlled distributional changes relative to previously acquired knowledge. In this way, a dynamic balance between continual learning and machine unlearning can be achieved.

Following prior work [8], we view continual learning and machine unlearning as two types of controlled distributional updates in a model facing sequential task requests. Instead of fully retraining to the ideal model $\theta^\star$ (trained on all learning data while excluding all samples requested to be forgotten), we perform approximate CLU by constraining the output distribution drift relative to a reference model.

Let π_θ denote the language model with parameters θ and output distribution $p_\theta(\cdot \mid s_{<t})$. At update step k, we take the current model θ_k as the reference model, i.e., $\theta_{\text{ref}} = \theta_k$.

Given three data partitions at step k—retain set $\mathcal{D}^R$, new learning set $\mathcal{D}^L$, and forget set $\mathcal{D}^U$—we define a unified objective that (i) fits new knowledge on $\mathcal{D}^L$, (ii) suppresses target knowledge on $\mathcal{D}^U$, and (iii) limits distributional drift on $\mathcal{D}^R$ (see Figure 2 for an illustration of the framework):

$$\theta_{k+1} = \arg\min_{\theta} \underbrace{\mathcal{L}_{\text{req}}(\theta; \mathcal{D}^L, \mathcal{D}^U)}_{\text{learning / unlearning}} + \lambda \underbrace{\mathcal{R}_{\text{retain}}(\theta; \theta_{\text{ref}}, \mathcal{D}^R)}_{\text{retention (drift control)}}, \tag{3}$$

where $\lambda > 0$ controls the retention strength.

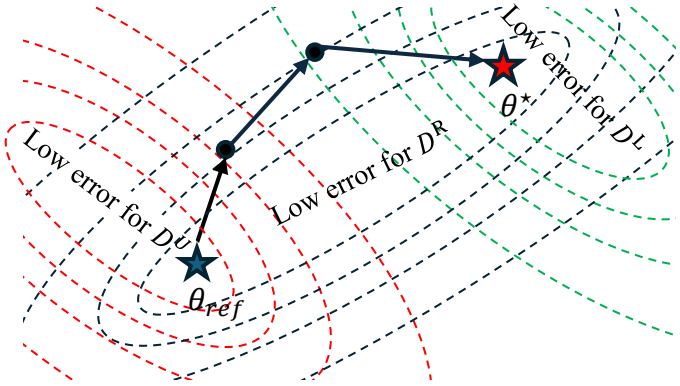


Figure 2. The unified distributional framework for CLU. The framework operates on three data partitions: the retain set $\mathcal{D}^R$ (historical knowledge to preserve), the learning set $\mathcal{D}^L$ (new knowledge to acquire), and the forget set $\mathcal{D}^U$ (target knowledge to eliminate). At each update step, the model θ is optimized relative to a reference model θ_{ref} through drift-controlled updates, balancing three objectives: (i) retention regularization (drift minimization on $\mathcal{D}^R$) to maintain stability, (ii) learning via supervised fine-tuning on $\mathcal{D}^L$ for knowledge acquisition, and (iii) unlearning via gradient ascent on $\mathcal{D}^U$ for knowledge removal.

For retention regularization (stay close on $\mathcal{D}^R$), we characterize distributional drift on retained data via the KL divergence between the reference model and the updated model:

$$\mathcal{R}_{\text{retain}}(\theta; \theta_{\text{ref}}, \mathcal{D}^R) = \mathbb{E}_{s \sim \mathcal{D}^R}\left[\frac{1}{|s|}\sum_{t=1}^{|s|} D_{\text{KL}}\Big(p_{\theta_{\text{ref}}}(\cdot \mid s_{<t}) \,\|\, p_\theta(\cdot \mid s_{<t})\Big)\right], \tag{4}$$

where s denotes a sequence in the retain set $\mathcal{D}^R$, $|s|$ is its length, $s_{<t}$ represents the prefix up to position $t-1$, and $p_\theta(\cdot \mid s_{<t})$ is the model's next-token probability distribution conditioned on $s_{<t}$. **Implementation Bridge:** Equation (4) serves as the foundational design principle for our unified framework. While explicit computation of this token-level KL term is avoided to maintain data privacy and efficiency, we operationalize this principle by constraining the parameter-space evolution. Specifically, under a standard model smoothness assumption—namely that the logit output function f_θ is twice continuously differentiable with bounded first- and second-order derivatives (Assumption A1)—we establish via Taylor expansion of the logit function and Lipschitz analysis of the softmax operator that the token-level KL divergence on the retain set isformallyupper-bounded by $C_1\|\Delta\theta\|_2^2 + C_2\|\Delta\theta\|_2$, where C_1, C_2 are explicit constants determined by model properties (Theorem A1, Appendix A). This theoretical link directly motivates our choice of structural constraints:localization via sparse masking and direction control via orthogonal projection are not merely empirical heuristics, but are principled proxies for minimizing Equation (4) without historical data rehearsal.

The request term $\mathcal{L}_{\text{req}}$ depends on whether the incoming request is learning or unlearning:

$$\mathcal{L}_{\text{req}}(\theta; \mathcal{D}^L, \mathcal{D}^U) = \begin{cases} \mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}^L), & \text{(learning request)}, \\ \mathcal{L}_{\text{GA}}(\theta; \mathcal{D}^U), & \text{(unlearning request)}, \end{cases} \tag{5}$$

In our research, we employ supervised fine-tuning (SFT) for learning tasks and gradient ascent (GA) for unlearning tasks. Specifically, the SFT loss is defined as the standard negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta; \mathcal{D}^L) = -\mathbb{E}_{(x,y)\sim\mathcal{D}^L}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\log p_\theta(y_t \mid x, y_{<t})\right], \tag{6}$$

where (x, y) denotes a prompt-response pair, and $p_\theta(y_t \mid x, y_{<t})$ is the model's predicted probability for the next token y_t given the prompt x and previous tokens $y_{<t}$.

For unlearning, we adopt the gradient ascent objective that maximizes the loss on the forget set, thereby reducing the model's confidence on the targeted knowledge:

$$\mathcal{L}_{\text{GA}}(\theta; \mathcal{D}^U) = \mathbb{E}_{(x,y)\sim\mathcal{D}^U}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\log p_\theta(y_t \mid x, y_{<t})\right], \tag{7}$$

which effectively pushes the model away from reproducing responses in the forget set $\mathcal{D}^U$.

Scope of Unification. We emphasize that the unification in our framework is *operational* rather than loss-level. The task-specific objectives for learning (Equation (6)) and unlearning (Equation (7)) are fundamentally distinct—SFT minimizes next-token prediction loss while GA maximizes it. What is unified is the *retention-controlled parameter evolution mechanism*: both operations are executed within the same constrained low-rank adapter space (frozen A, sparsely masked B), subject to the same orthogonal projection constraints, and governed by the same drift-control principle (Equation (4)). This shared infrastructure ensures that regardless of whether the current task involves knowledge acquisition or removal, the parameter update respects the same stability guarantees on retained knowledge.

Equations (3)–(5) yield three practical design principles: (i) retention control via $\mathcal{R}_{\text{retain}}$ on $\mathcal{D}^R$, (ii) localization by restricting updates to a small parameter subset (to reduce interference), and (iii) direction control by constraining update directions to minimize impact on historical knowledge. These principles motivate our parameter-efficient implementation with frozen LoRA projection matrices A, sparse masking, and orthogonal gradient projection in Section 3.3. Details could be found in Table 1.

Table 1. Concise mapping from KL-inspired design principles to their corresponding algorithmic components in our implementation.

Design Principle (Conceptual)	Implementation Component (Algorithmic)	Role/Intuition
Retention control on $\mathcal{D}^R$ (drift-aware stability)	Frozen LoRA projection matrix A (Section 3.3.1)	Constrains updates to a shared low-dimensional subspace, promoting stable behavior on retained knowledge.
Localization (reduce interference)	Sparse masking on B (Section 3.3.2)	Restricts parameter changes to a small subset, limiting collateral forgetting and isolating task-specific edits.
Direction control (protect past directions)	Orthogonal gradient projection (Section 3.3.3)	Removes update components aligned with previously learned directions, reducing destructive interference across tasks.

3.3. Method

In this study, we adopt Low-Rank Adaptation (LoRA) to address the problem of CLU. Compared with full-parameter fine-tuning, LoRA does not modify the parameters of the backbone large language model; instead, it introduces only a small number of additional trainable parameters. Prior studies have shown that this parameter-efficient strategy can achieve performance comparable to full fine-tuning [9]. The overall architecture of the proposed framework is illustrated in Figure 3.

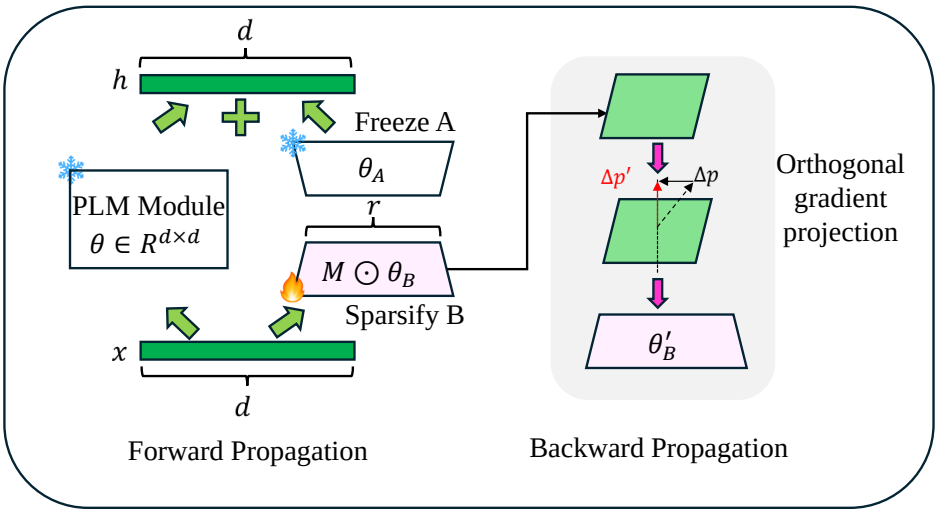


Figure 3. Overview of the proposed LoRA-based framework with frozen matrix A, sparse masking on matrix B, and orthogonal gradient projection for knowledge management in continual learning and unlearning.

LoRA fine-tunes a large language model for new tasks by factorizing the weight update into the product of two low-rank matrices. Formally, for a specific task t, given a pretrained weight matrix $w \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, the weight update $\Delta_t \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ is constrained to be low-rank:

$$h = xw + x\Delta_t = xw + xA_tB_t, \tag{8}$$

where $x \in \mathbb{R}^{1\times d_{\text{in}}}$ is the input feature vector, $h \in \mathbb{R}^{1\times d_{\text{out}}}$ is the output, $A_t \in \mathbb{R}^{d_{\text{in}}\times r}$ is the projection matrix, $B_t \in \mathbb{R}^{r\times d_{\text{out}}}$ is the expansion matrix, and $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the rank of the low-rank decomposition. We refer to Δ_t as the *LoRA adapter* for task t. In practice, LoRA adapters are typically applied to multiple projection matrices in Transformer layers (e.g., w_k and w_v).

Conventionally, both the low-rank projection matrix A_t and the low-rank expansion matrix B_t are updated via gradient descent. The matrix A_t is usually randomly initialized (e.g., with a Gaussian distribution), while B_t is initialized to zero to ensure $\Delta_t = 0$ at the start of training.

3.3.1. Freezing the LoRA Matrix A

In our CLU setting, to reduce interference between learning and unlearning across a task stream and to preserve the backbone model's general capability, we freeze the low-rank projection matrix A and only optimize the task-specific expansion matrices B. Concretely, for the task sequence $t, t+1, \ldots$, all tasks share the same fixed A but maintain different B matrices, yielding the low-rank updates

$$\Delta_t = AB_t, \qquad \Delta_{t+1} = AB_{t+1}. \tag{9}$$

This design constrains all task updates to a common low-dimensional subspace spanned by A, while allowing different tasks to adapt through different directions in the r-dimensional coefficient space. As a result, the correlation (or orthogonality) between the induced parameter updates in the original space is largely governed by the alignment between the corresponding B matrices. This relationship can be motivated as follows under standard random matrix concentration assumptions:

Let $A \in \mathbb{R}^{d_{\text{in}} \times r}$ be initialized with i.i.d. standard normal entries and then frozen. Consider the Frobenius inner product between the adapters of two consecutive tasks:

$$\langle \Delta_t, \Delta_{t+1} \rangle = \text{Tr}\Big(\Delta_t^\top \Delta_{t+1}\Big) = \text{Tr}\Big(B_t^\top A^\top A B_{t+1}\Big), \tag{10}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product and $\text{Tr}(\cdot)$ is the matrix trace operator. When d_{in} is large, random matrix concentration suggests that

$$A^\top A \approx \alpha I_r, \tag{11}$$

where $I_r \in \mathbb{R}^{r \times r}$ is the identity matrix and $\alpha > 0$ is a constant determined by the initialization of A. Substituting into the inner product gives

$$\langle \Delta_t, \Delta_{t+1} \rangle \approx \alpha \, \text{Tr}\Big(B_t^\top B_{t+1}\Big) = \alpha \langle B_t, B_{t+1} \rangle. \tag{12}$$

This suggests that orthogonality between the induced updates in the original parameter space, i.e., $\langle \Delta_t, \Delta_{t+1} \rangle \approx 0$, can be promoted when the corresponding coefficients satisfy $\langle B_t, B_{t+1} \rangle \approx 0$. While this argument serves as an intuition under idealized random initialization assumptions, in practice orthogonality is enforced through explicit masking and projection mechanisms regardless of this approximation. Motivated by this perspective, we enforce near-orthogonality between the B-space parameters for consecutive tasks using two complementary mechanisms: sparse masking (Section 3.3.2) to protect important large-magnitude parameters while allowing selective updates to less critical parameters, and orthogonal gradient projection (Section 3.3.3) to remove components of the current task's update that align with previously learned directions. Together, these techniques promote approximately perpendicular adaptations in the B-space along the task stream, thereby alleviating destructive interference while preserving the backbone's general capability.

3.3.2. Sparse Masking for the Weight Matrix B

To mitigate interference between tasks, we construct a sparse mask M_t before training task $t (t > 1)$, based on the magnitude statistics of the current parameters. The mask is then applied during optimization to restrict parameter updates: only parameters with mask value 1 are allowed to be updated, while parameters with mask value 0 remain fixed.

Concretely, prior to training task t, we aggregate all parameters from the collection of B_t matrices across layers/projections, denoted by $\mathcal{B}_t$, and compute a global threshold $\tilde{T}_t$ as the s% quantile of their absolute values, where s denotes the sparsity ratio. Following the standard magnitude-based masking approach, the mask for each matrix B_t is defined as:

$$M_t = \mathbb{I}\big(|B_t| < \tilde{T}_t\big), \qquad \tilde{T}_t = \text{Quantile}_{s\%}(|\mathcal{B}_t|), \tag{13}$$

where $\mathbb{I}(\cdot)$ is the element-wise indicator function that returns 1 for parameters below the threshold and 0 otherwise. This formulation protects the top s% largest-magnitude parameters by setting their mask values to 0 (frozen), while allowing updates to the remaining $(1 - s)$% smaller parameters with mask value 1.

3.3.3. Orthogonal Gradient Projection

To further suppress catastrophic forgetting in continual learning and to prevent unintended damage to non-target knowledge during unlearning, we introduce an orthogonal gradient projection strategy. Recent studies suggest that if the gradient update direction is orthogonal to the feature subspace of previous tasks, the impact on old tasks is minimized, thereby reducing forgetting [33].

Consider training on task $t+1$ after having learned tasks $1, \ldots, t$. Let E denote a generic trainable parameter matrix. The parameter update can be written as:

$$E_{t+1} = E_t + \Delta E, \tag{14}$$

where ΔE represents the parameter change. To preserve the output of old task t with input feature x_t, where $f_\theta(\cdot, \cdot)$ denotes the model's output function parameterized by θ, we require:

$$f_\theta(E_t + \Delta E, x_t) = f_\theta(E_t, x_t). \tag{15}$$

By linearization, this condition is approximately satisfied when:

$$\langle \nabla_E f_\theta(E_t, x_t), \Delta E \rangle = 0, \tag{16}$$

meaning that the parameter update ΔE should be orthogonal to the gradient direction of the old-task output with respect to the parameters.

Let $\theta \in \mathbb{R}^d$ denote the vector of all trainable model parameters, where d is the total number of parameters. For each task i, let θ_i^{init} and θ_i^{final} denote the parameter vectors before and after finishing training on task i, respectively. We define the task-update displacement and its normalized direction as

$$\Delta\theta_i \triangleq \theta_i^{\text{final}} - \theta_i^{\text{init}}, \qquad v_i \triangleq \frac{\Delta\theta_i}{\|\Delta\theta_i\|_2}, \tag{17}$$

where $\Delta\theta_i \in \mathbb{R}^d$ represents the net parameter change induced by task i, $v_i \in \mathbb{R}^d$ is the corresponding unit direction ("task direction"), and $\|\cdot\|_2$ denotes the Euclidean norm.

When training on a new task t, let $g_t \in \mathbb{R}^d$ denote the raw gradient of the task-t loss with respect to parameters, i.e., $g_t = \nabla_\theta \mathcal{L}_t(\theta)$, computed at the current optimization step. To prevent updates that interfere with previously learned task directions, we project g_t onto the orthogonal complement of the subspace spanned by the stored directions $\{v_i\}_{i=1}^{t-1}$:

$$g_t^\perp \triangleq g_t - \sum_{i=1}^{t-1} \left(g_t^\top v_i \right) v_i, \tag{18}$$

where $g_t^\perp \in \mathbb{R}^d$ is the projected gradient used for the parameter update, $g_t^\top v_i$ is the scalar inner product measuring the component of g_t along v_i, and $(g_t^\top v_i)v_i$ is the corresponding projection component removed from g_t.

As a result, the projected gradient is orthogonal to every previous task direction:

$$\left(g_t^\perp \right)^\top v_i = 0, \qquad \forall\, i \in \{1, \ldots, t-1\}. \tag{19}$$

This gradient projection strategy complements the sparse masking mechanism: the sparse mask constrains *where* updates occur (i.e., which parameters are modified by protecting important parameters), while orthogonal gradient projection constrains the *direction* of updates (i.e., how parameters are modified). Their combination enables the model to balance parameter protection, directional orthogonality, and knowledge stability, thereby

effectively mitigating catastrophic forgetting and reducing the adverse impact of unlearning on the model's general capabilities.

3.3.4. Overall Algorithm

Algorithm 1 summarizes the complete procedure of our unified CLU framework.

Algorithm 1 Unified CLU Framework with Parameter-Efficient Adaptation

Require: Base model θ_0, task sequence $\{T_1, T_2, \ldots, T_T\}$ where $T_t = (D_t, R_t)$ and $R_t \in \{L, U\}$
Require: Hyperparameters: LoRA rank r, sparsity ratio s, learning rate η
Ensure: Updated model θ_T with LoRA adapters
Initialize LoRA matrices: $A \in \mathbb{R}^{d_{\text{in}} \times r}$ (random), $B_0 = \mathbf{0}$ (zero matrix)
Freeze A for all subsequent tasks
Initialize task direction history $\mathcal{V} = \emptyset$
for each task $t = 1, 2, \ldots, T$ **do**
    // Construct sparse mask for task t
    if $t > 1$ **then**
        $\tilde{T}_t \leftarrow \text{Quantile}_{s\%}(|\mathcal{B}_{t-1}|)$ ▷ Compute global threshold
        $M_t \leftarrow \mathbb{I}(|B_{t-1}| < \tilde{T}_t)$ ▷ Mask: 1 for small params, 0 for large
    else
        $M_t \leftarrow \mathbf{1}$ ▷ No masking (all-ones matrix) for first task
    end if

    // Training loop for task t
    $B_t \leftarrow B_{t-1}$, $\theta_t^{\text{init}} \leftarrow \theta_{t-1}$
    for each training step **do**
        // Compute task-specific loss
        if $R_t = L$ (learning) **then**
            $\mathcal{L} \leftarrow \mathcal{L}_{\text{SFT}}(\theta; D_t^L)$
        else if $R_t = U$ (unlearning) **then**
            $\mathcal{L} \leftarrow \mathcal{L}_{\text{GA}}(\theta; D_t^U)$
        end if

        // Compute and project gradient
        $g_t \leftarrow \nabla_{B_t} \mathcal{L}$
        $g_t^{\perp} \leftarrow g_t - \sum_{v_i \in \mathcal{V}} (g_t^{\top} v_i) v_i$ ▷ Orthogonal projection

        // Apply sparse mask and update
        $B_t \leftarrow B_t - \eta \cdot (g_t^{\perp} \odot M_t)$ ▷ Masked parameter update ($\odot$: element-wise product)
    end for

    // Store task direction for future projection
    $\theta_t^{\text{final}} \leftarrow \theta_t$ with adapter $A \cdot B_t$
    $\Delta\theta_t \leftarrow \theta_t^{\text{final}} - \theta_t^{\text{init}}$
    $v_t \leftarrow \Delta\theta_t / \|\Delta\theta_t\|_2$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{v_t\}$
end for
return θ_T with LoRA adapter $A \cdot B_T$

4. Experiments

4.1. Dataset and Experimental Setup

We adopt the Task of Fictitious Unlearning (TOFU) benchmark [20] for evaluation. TOFU contains profiles of 200 fully fictitious authors, where each profile consists of 20 question–answer (QA) pairs. All profiles are carefully constructed to ensure that their content does not appear in the model's pretraining data, thereby providing a controlled environment for evaluating whether a model can selectively forget specific information.

To emulate a CLU setting, we design an experimental protocol with six tasks: three unlearning (UL) tasks and three continual learning (CL) tasks. We construct six data groups from TOFU, where each group contains 20 QA pairs and is assigned to one specific task

(i.e., $6 \times 20 = 120$ QA pairs in total). Specifically, three data groups ($3 \times 20 = 60$ QA pairs) are designated as UL data for the three unlearning tasks, while the remaining three data groups ($3 \times 20 = 60$ QA pairs) serve as CL data for the three learning tasks. Following common practice, the base model undergoes an initial supervised fine-tuning (SFT) stage on a combined dataset consisting of a retain set $\mathcal{D}^{\mathrm{L}_0}$ and the three UL data groups (60 QA pairs total) that will subsequently be unlearned. This SFT stage establishes both the baseline knowledge to be retained and the target knowledge to be selectively forgotten in later stages. The three CL data groups are kept separate and used exclusively for evaluating continual learning capabilities under interleaved unlearning operations.

We conduct experiments in an interleaved schedule that alternates between unlearning and learning tasks. Specifically, the six tasks are executed in a fixed sequence:

$$\mathrm{UL}_1 \rightarrow \mathrm{CL}_1 \rightarrow \mathrm{UL}_2 \rightarrow \mathrm{CL}_2 \rightarrow \mathrm{UL}_3 \rightarrow \mathrm{CL}_3,$$

where each UL_i ($i = 1, 2, 3$) represents an unlearning task that aims to forget the i-th injected data group, and each CL_i ($i = 1, 2, 3$) represents a continual learning task that learns the i-th held-out data group. This interleaved design enables us to evaluate whether the model can successfully unlearn specific knowledge while simultaneously acquiring new knowledge without catastrophic forgetting or interference.

In our experiments, we conduct comprehensive evaluations using two representative large language models: Qwen3-4B-Instruct and Llama3-8B-Instruct. To efficiently adapt these models while maintaining parameter efficiency, we employ LoRA for both continual learning and unlearning tasks. The training configuration is set as follows: we use a learning rate of 5×10^{-5} with AdamW optimizer, a batch size of 16, and train the models for 10 epochs. For the LoRA hyperparameters, we set the rank $r = 8$ to control the low-rank decomposition, the scaling coefficient $\alpha = 16$ to regulate the magnitude of LoRA updates, and apply a dropout rate of 0.05 to the LoRA layers to prevent overfitting. These hyperparameters are kept consistent across all unlearning and continual learning stages to ensure fair comparisons and reproducibility.

4.2. Baselines

To systematically evaluate the proposed CLU framework, we compare it against several representative baseline methods across sequential unlearning-continual learning tasks. Given our focus on an interleaved CLU protocol under a single Parameter-Efficient Fine-Tuning (PEFT) adapter without replay or task-specific routing, we adopt the standard supervised fine-tuning (SFT) approach as the CL backbone for all methods. For the unlearning (UL) tasks, we compare our framework against the following established LLM unlearning baselines:

Gradient Ascent (GA) [20]. This method performs unlearning by maximizing the negative log-likelihood on the forget set, thereby progressively reducing the model's confidence in generating answers related to the data that should be forgotten.
Gradient Ascent + Gradient Descent (GA + GD) [34]. This approach combines gradient ascent on the forget set with gradient descent on the retain set. It enables the model to erase undesired knowledge while simultaneously maintaining performance on data that should be retained.
KL-Regularized Gradient Ascent (GA + KL) [34]. This method applies gradient-ascent unlearning on the forget set while constraining the model's distributional drift via a KL divergence regularizer with respect to a reference model. This prevents excessive deviation from the original model behavior during the unlearning process.

Negative Preference Optimization (NPO) [22]. This technique explicitly downweights forget-set targets by penalizing the likelihood ratio under a negative-preference objective, thereby directly diminishing the model's confidence in producing answers from the forget set.

Direct Preference Optimization (DPO) [20]. We adapt DPO to the unlearning scenario by constructing preference pairs where a neutral or alternative response is preferred over the forget-set target. The pairwise objective increases the probability of neutral responses while decreasing the probability of undesired responses relative to a reference policy.

Low-Rank Adaptation (LoRA) [9]. Instead of updating all model parameters, LoRA injects trainable low-rank decomposition matrices into the model's attention layers. A single shared LoRA adapter is trained across all sequential tasks (both unlearning and continual learning), modifying the model's behavior through parameter-efficient updates. This enables efficient adaptation across the entire task sequence while maintaining the base model's weights frozen.

For detailed mathematical formulations and algorithmic implementations of these baseline methods, please refer to Appendix B.

4.3. Evaluation Metrics

We evaluate forgetting and utility from multiple perspectives, including lexical overlap, semantic similarity, and factual consistency.

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) measures token-level overlap between the generated answer and the reference answer [35]. We adopt ROUGE-L recall, which is based on the longest common subsequence (LCS):

$$\text{ROUGE-L} = \frac{\text{LCS}(g, r)}{|r|}, \tag{20}$$

where $\text{LCS}(g, r)$ denotes the length of the longest common subsequence between g and r, g is the generated answer, r is the reference answer, and $|r|$ is the length of the reference answer.

Cosine Similarity (CS) [36] measures the semantic similarity between model outputs before and after training. We obtain sentence embeddings using Sentence-BERT [37], compute the cosine similarity between pre- and post-training outputs, and truncate negative values to zero:

$$\text{CS} = \max\left(0, \frac{\mathbf{e}_{\text{pre}} \cdot \mathbf{e}_{\text{post}}}{\|\mathbf{e}_{\text{pre}}\|_2 \, \|\mathbf{e}_{\text{post}}\|_2}\right), \tag{21}$$

where $\mathbf{e}_{\text{pre}}$ and $\mathbf{e}_{\text{post}}$ are embeddings of the outputs before and after training, respectively. A lower CS suggests that the trained model introduces semantic drift.

Entailment Score (ES) [36] assesses the factual consistency between the model's output and the ground-truth answer, based on Natural Language Inference (NLI). We use a pre-trained NLI model [38] to predict whether the model output entails the ground-truth answer, and compute the proportion of outputs predicted as entailment:

$$\text{ES} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\Big[\text{NLI}(g_i, r_i) = \text{entailment}\Big], \tag{22}$$

where N is the number of evaluated samples, g_i is the i-th generated answer, r_i is the i-th reference answer, $\text{NLI}(\cdot, \cdot)$ denotes the NLI model's prediction, and $\mathbb{I}[\cdot]$ is the indicator function. A higher ES indicates better factual alignment, and lower scores signal hallucinated or incorrect outputs.

To provide a comprehensive assessment of model performance, we introduce two aggregate metrics that combine the aforementioned individual measures:

Model Utility (MU) serves as a task-level response quality proxy, quantifying the model's ability to retain useful knowledge on the retain set and newly learned tasks. It is computed as the arithmetic mean of ROUGE-L, CS, and ES:

$$\text{MU} = \frac{1}{3}(\text{ROUGE-L} + \text{CS} + \text{ES}). \tag{23}$$

A higher MU indicates that the model maintains strong performance on data that should be preserved.

Forgetting Proxy (FP) measures the degree to which model outputs deviate from original responses on the forget set under specified prompt templates. Rather than certifying irrecoverability in a privacy sense, FP quantifies behavioral redirection—the extent to which outputs shift away from target responses. It is defined as:

$$\text{FP} = 1 - \frac{1}{3}(\text{ROUGE-L} + \text{CS} + \text{ES}). \tag{24}$$

A higher FP indicates that the model produces outputs that diverge substantially from the original responses on forget-set samples, reflecting observable behavioral change rather than provable knowledge elimination.

Scope and Limitations of Evaluation Metrics. It is important to clarify the applicability boundaries of MU and FP: Not equivalent to privacy guarantees: FP does not equate to reduced membership inference risk or certified non-extractability of forgotten data. It reflects observable output deviation under controlled prompting, not cryptographic or information-theoretic guarantees of knowledge removal. Behavioral unlearning in controlled settings: Our evaluation framework and conclusions are scoped to behavioral unlearning or behavioral redaction within a controlled benchmark setting. Claims regarding "unlearning" should be interpreted as empirical output-level behavior changes, not as guarantees of complete knowledge erasure or resistance to adversarial extraction attempts.

Following established practices in continual unlearning evaluation [36], we define task-specific evaluation sets for computing MU and FP at each stage of the sequential task sequence. For a continual learning task CL_i, the MU is computed on the union of the current training data and all previous learning datasets $\bigcup_{j=0}^{i} \mathcal{D}^{\text{L}_j}$ (where $\mathcal{D}^{\text{L}_0}$ denotes the initial retain set from the SFT stage, excluding data intended for subsequent unlearning since those samples are also trained during SFT but will be selectively forgotten later), while the FP is evaluated on the forget set from the preceding unlearning task $\mathcal{D}^{\text{U}_i}$. Conversely, for an unlearning task UL_i, the MU is measured on the cumulative set of all prior learning data $\bigcup_{j=0}^{i-1} \mathcal{D}^{\text{L}_j}$ to assess knowledge retention, and the FP is computed on the current forget set $\mathcal{D}^{\text{U}_i}$ used for training the unlearning objective. This evaluation protocol ensures that we capture both the model's ability to preserve previously acquired knowledge and its effectiveness in selectively forgetting target information across the sequential task trajectory.

5. Results

In this section, we present a comprehensive evaluation of the proposed method across multiple dimensions. We first report the main results on the TOFU benchmark and a real-world unlearning dataset to demonstrate the effectiveness and stability of our approach compared to several representative baselines. Subsequently, we conduct a detailed sensitivity analysis of the sparsity parameter and investigate the distribution drift to quantify the preservation of general capabilities. Furthermore, an extensive ablation study is performed to verify the contribution of each individual component. Finally, we analyze the parameter and computational efficiency to highlight the practical advantages of our framework in resource-constrained scenarios.

5.1. Main Results

To evaluate the performance of the proposed method, we conduct a series of experiments on the TOFU benchmark. We compare the proposed method against several representative baselines. The experimental results are shown in Table 2.

Table 2. Performance comparison of different methods on Qwen3-4B-Instruct and Llama3-8B-Instruct models.

Model	Method	UL_1		CL_1		UL_2		CL_2		UL_3		CL_3		Average
		MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	
Qwen3-4B-Instruct	GA	0.39	0.60	0.50	0.59	0.31	0.76	0.43	0.71	0.27	0.80	0.37	0.70	0.536
	GA + GD	0.47	0.55	0.52	0.56	0.43	0.72	0.49	0.63	0.41	0.71	0.47	0.62	0.548
	GA + KL	0.57	0.48	0.52	0.47	0.43	0.67	0.48	0.63	0.46	0.66	0.47	0.61	0.538
	NPO	0.40	0.50	0.42	0.58	0.44	0.66	0.44	0.70	0.29	0.79	0.43	0.68	0.528
	DPO	0.54	0.49	0.51	0.51	0.45	0.64	0.49	0.63	0.47	0.67	0.48	0.71	0.549
	LoRA	0.52	0.54	0.54	0.49	0.03	0.96	0.28	0.82	0.03	0.99	0.23	0.86	0.524
	Our Method	0.59	0.50	0.61	0.52	0.58	0.51	0.58	0.52	0.55	0.54	0.56	0.66	0.560
Llama3-8B-Instruct	GA	0.52	0.52	0.59	0.51	0.41	0.77	0.44	0.64	0.38	0.69	0.38	0.70	0.546
	GA + GD	0.67	0.41	0.68	0.41	0.41	0.71	0.51	0.70	0.38	0.69	0.43	0.67	0.556
	GA + KL	0.62	0.50	0.68	0.49	0.41	0.65	0.47	0.61	0.36	0.72	0.39	0.66	0.547
	NPO	0.59	0.54	0.67	0.52	0.41	0.69	0.47	0.62	0.33	0.71	0.38	0.68	0.551
	DPO	0.59	0.40	0.59	0.36	0.52	0.61	0.52	0.57	0.48	0.63	0.53	0.59	0.533
	LoRA	0.59	0.50	0.69	0.37	0.03	0.97	0.29	0.84	0.03	0.99	0.25	0.89	0.537
	Our Method	0.81	0.39	0.80	0.38	0.74	0.38	0.73	0.38	0.68	0.45	0.67	0.46	0.573

As shown in Table 2, our method achieves the best average performance (computed as the mean of all 12 MU and FP metric values across the six sequential tasks) on both models: 0.560 on Qwen3-4B-Instruct and 0.573 on Llama3-8B-Instruct, outperforming all baselines with exceptional stability (variance $\pm$ 0.0089 across five seeds). The MU metric remains consistently high throughout CLU, while baselines like GA show dramatic fluctuations and LoRA suffers catastrophic forgetting. Although our FP scores (0.38–0.46) are lower than aggressive methods, this reflects a deliberate design choice prioritizing stable knowledge retention over maximal output deviation, making it particularly suitable for scenarios requiring controlled, policy-driven knowledge removal with minimal disruption to retained capabilities.

5.2. Sensitivity Analysis

To investigate the impact of the sparsity parameter on model performance, we conduct sensitivity analysis on the Qwen3-4B-Instruct model by varying the sparsity level from 0 to 0.9. The experimental results are presented in Table 3.

Table 3. Sensitivity analysis of sparsity parameter on model performance.

Sparsity	UL_1		CL_1		UL_2		CL_2		UL_3		CL_3	
	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP
0.0	0.52	0.54	0.54	0.49	0.03	0.96	0.28	0.82	0.03	0.99	0.23	0.86
0.3	0.52	0.54	0.55	0.49	0.10	0.94	0.24	0.84	0.09	0.94	0.17	0.83
0.5	0.52	0.54	0.55	0.49	0.20	0.79	0.27	0.81	0.11	0.90	0.21	0.80
0.7	0.52	0.54	0.58	0.52	0.30	0.73	0.33	0.72	0.19	0.82	0.25	0.77
0.9	0.59	0.50	0.61	0.52	0.58	0.51	0.58	0.52	0.55	0.54	0.56	0.56

As shown in Table 3, sparsity level significantly impacts performance. Without parameter protection (sparsity = 0), the model suffers catastrophic forgetting (MU: 0.54→0.03). Performance improves progressively as sparsity increases from 0.3 to 0.7. At sparsity = 0.9

(protecting top 90% parameters), the model achieves optimal performance with average 0.55 and consistently high MU scores (0.58, 0.58, 0.55, 0.56), demonstrating that aggressive parameter protection is crucial for CLU. We therefore adopt sparsity = 0.9 for all experiments.

5.3. Distribution Drift Analysis

To quantify the side effects of knowledge unlearning on the model's general capabilities, we introduce the Token-level Distribution Drift proxy. Unlike coarse-grained metrics such as accuracy or perplexity, this metric captures the microscopic probability shifts in the model's output distribution.

For a given sample in the retain set $\mathcal{D}_{retain}$, let $P(w|t_<)$ be the next-token probability distribution of the reference model θ_{ref} (the initial SFT model) and $Q(w|t_<)$ be that of the current unlearned model θ_{cur}. The token-level Kullback-Leibler (KL) divergence is defined as:

$$D_{KL}(P \parallel Q) = \sum_{w \in \mathcal{V}} P(w|t_<) \log \left(\frac{P(w|t_<)}{Q(w|t_<)} + \epsilon \right) \tag{25}$$

where $\mathcal{V}$ denotes the full vocabulary. To ensure numerical stability and symmetry, we also report the Jensen–Shannon (JS) divergence:

$$D_{JS}(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M) \tag{26}$$

where $M = \frac{1}{2}(P + Q)$. These metrics are averaged across all tokens within the generated Answer segment using a teacher-forcing paradigm.

Experimental Setup. We use the checkpoint after the initial supervised fine-tuning (SFT) as θ_{ref} to maintain a consistent Reference Baseline for distribution comparison. To avoid the dilution of the drift signal by fixed prompt templates, we apply a mask for Evaluation Focus that restricts calculation exclusively to the Answer portion of the tokens. For Sampling, we randomly sample $N = 50$ instances from the Retain Set for each task stage.

Results and Analysis. The experimental results in Figures 4 and 5 reveal distinct behaviors in distribution maintenance. Conventional unlearning methods, such as GA and NPO, exhibit a progressive increase in both KL and JS divergence as the task sequence advances. This cumulative drift is particularly pronounced in the later stages (e.g., UL_3 and CL_3), where the model's output distribution deviates significantly from the original SFT baseline, leading to the "catastrophic collapsing" of general capabilities. In contrast, our proposed method (Ours) maintains a consistently low and stable drift throughout the entire CLU process. The near-zero KL divergence indicates that our parameter-protected orthogonal optimization effectively confines the updates to a narrow subspace, successfully erasing specific knowledge without perturbing the model's fundamental linguistic patterns.

Connecting Distribution Drift to Behavioral Metrics. The token-level KL divergence on the retain set and the behavioral metric MU are not independent observations—they are causally linked through the generation process. Since MU is computed from ROUGE-L, CS, and ES on retain-set outputs, and these outputs are generated autoregressively from $p_\theta(\cdot \mid s_{<t})$, any systematic shift in this token-level distribution propagates directly into degraded output quality. This causal chain—*cumulative KL drift* $\rightarrow$ *shifted generation distribution* $\rightarrow$ *degraded retain-set outputs* $\rightarrow$ *lower MU*—is clearly reflected in the cross-method comparison. For instance, on Qwen3-4B-Instruct, GA exhibits progressively increasing KL divergence (Figure 4) accompanied by a 46% decline in MU (from 0.50 at CL_1 to 0.27 at UL_3), while its ostensibly high FP scores (0.60$\rightarrow$0.80) are not indicative of precise forgetting but rather of indiscriminate distributional collapse affecting both retain and forget sets. In contrast, our method's near-zero KL divergence corresponds to only a 7% MU decline

(0.59→0.55), with FP growing selectively (0.50→0.54). This demonstrates that low KL drift is the distributional-level mechanism enabling the behavioral-level stability–plasticity balance: parameter-space constraints (Theorem A1) bound the KL divergence, which in turn preserves retain-set generation fidelity (high MU) while permitting targeted behavioral change on the forget set (moderate FP). The full chain—*parameter constraints → bounded KL → stable MU with selective FP*—provides end-to-end empirical validation of our drift-aware design principle.

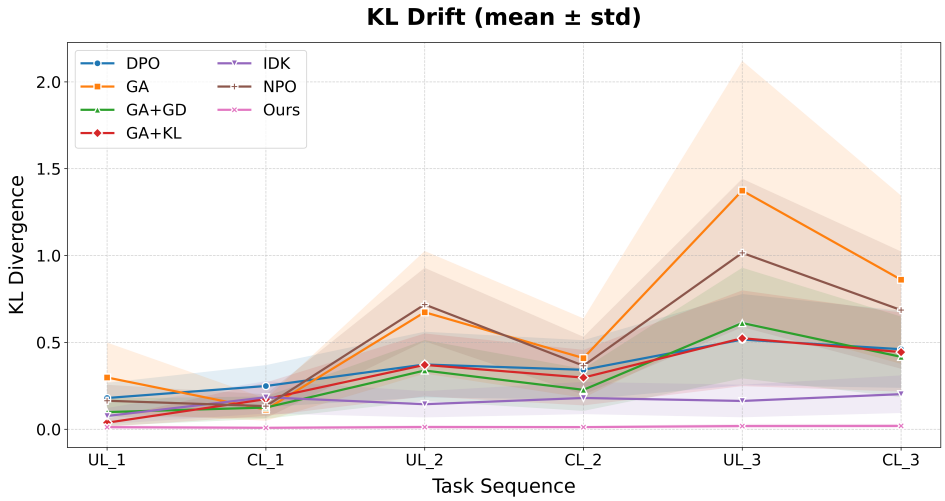


Figure 4. Token -level KL Divergence across task sequence. The shaded area represents ±1 standard deviation.

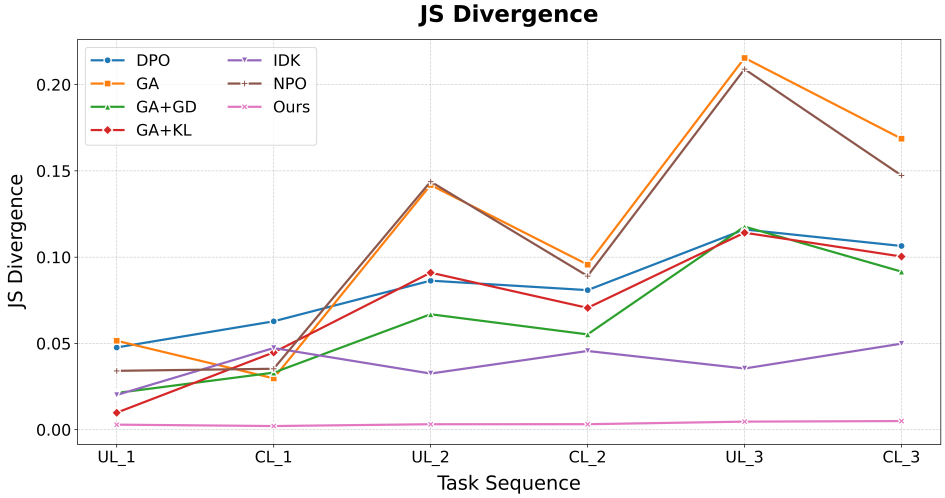


Figure 5. Token-level JS Divergence across task sequence.

5.4. Ablation Study

To evaluate the effectiveness of the proposed method, we conduct a series of ablation studies. The experimental results are shown in Table 4. The ablation studies are conducted on the TOFU benchmark. Here, a represents whether matrix A is frozen, b represents whether matrix B is sparsified, and c indicates whether the code uses orthogonal gradient projection. ✓ indicates that the corresponding component is enabled, while × indicates it is disabled.

Table 4. Ablation study under different settings.

Method			UL_1		CL_1		UL_2		CL_2		UL_3		CL_3	
a	*b*	*c*	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP
×	×	×	0.52	0.54	0.54	0.49	0.03	0.96	0.28	0.82	0.03	0.99	0.23	0.86
✓	×	×	0.59	0.50	0.60	0.50	0.38	0.76	0.47	0.61	0.31	0.72	0.40	0.62
×	✓	×	0.52	0.54	0.63	0.47	0.54	0.57	0.54	0.57	0.34	0.71	0.46	0.60
×	×	✓	0.52	0.54	0.54	0.49	0.03	0.97	0.18	0.94	0.01	0.98	0.15	0.97
×	✓	✓	0.52	0.54	0.63	0.46	0.56	0.53	0.53	0.55	0.35	0.72	0.44	0.64
✓	×	✓	0.59	0.50	0.59	0.48	0.38	0.68	0.45	0.64	0.31	0.71	0.36	0.66
✓	✓	×	0.59	0.50	0.61	0.51	0.58	0.50	0.59	0.52	0.44	0.52	0.44	0.64
✓	✓	✓	0.59	0.50	0.61	0.52	0.58	0.51	0.59	0.52	0.55	0.54	0.56	0.56

Table 4 reveals the critical role of each component. The baseline (no components) suffers catastrophic forgetting (MU: 0.54→0.03), underscoring the necessity of specialized mechanisms. Individually, freezing matrix A (component *a*) improves MU to 0.38–0.47, while sparse masking on matrix B (component *b*) achieves stronger gains (MU: 0.54, 0.54, 0.34, 0.46). Orthogonal projection alone (*c*) shows minimal improvement with extremely high FP (0.97–0.98). The combination $a + b$ achieves strong performance (MU: 0.58, 0.59, 0.44, 0.44), while the complete framework ($a + b + c$) reaches optimal performance with consistently high MU (0.58, 0.59, 0.55, 0.56) and balanced FP (0.51–0.56), confirming the synergistic contributions of all components.

Connecting Ablation Patterns to the Theoretical Bound. The observed contribution hierarchy—where magnitude-controlling mechanisms (freezing A and sparse masking on B) yield larger individual gains than the direction-controlling mechanism (orthogonal projection)—is consistent with the structure of our theoretical bound (Theorem A1). The bound $\mathbb{E}[D_{\mathrm{KL}}] \leq C_1\|\Delta\theta\|_2^2 + C_2\|\Delta\theta\|_2$ is dominated by magnitude terms: reducing $\|\Delta\theta\|_2$ yields both quadratic and linear reductions in the KL upper bound, whereas directional constraints operate only through the effective projection of $\Delta\theta$ onto critical subspaces. This explains why freezing A (which restricts updates to a low-rank subspace) and sparse masking (which zeros out updates to critical parameters) each independently prevent catastrophic forgetting, while orthogonal projection alone cannot compensate for unconstrained update magnitude. However, the benefit of orthogonal projection becomes pronounced in later tasks (UL_3: MU improves from 0.44 to 0.55 when added to a+b), where cumulative directional interference across multiple sequential updates becomes the binding constraint—a regime where magnitude control alone is insufficient.

5.5. Model Size and Computational Efficiency

Table 5 presents a comprehensive comparison of parameter and computational efficiency across different training approaches for the base model with 3.74 billion parameters. The results demonstrate that our proposed method achieves superior parameter efficiency compared to both full fine-tuning and standard LoRA approaches. Specifically, while full fine-tuning requires updating all 3.74B parameters and consumes 183.8 TFLOPs per training step, our method with rank $r = 8$ only requires training 8.40M parameters (0.22% of the base model), reducing the trainable parameter count by approximately 47% compared to standard LoRA ($r = 8$) with 15.63M parameters (0.42%). In terms of computational efficiency, our approach achieves 62.0 TFLOPs per step (33.7% of full fine-tuning), which is marginally more efficient than standard LoRA's 62.3 TFLOPs per step (33.9%). These results highlight that our method not only maintains competitive computational efficiency but also significantly reduces the memory footprint and parameter overhead, making it particularly suitable for resource-constrained scenarios and sequential learning tasks where parameter efficiency is crucial.

Table 5. Parameter and Computational Efficiency on Qwen3-4B-Instruct.

Method	Trainable Params	Ratio	FLOPs/Step
Base Model	3.74B	-	-
Full Fine-tuning	3.74B	100.0%	183.8 TFLOPs (100.0%)
LoRA ($r = 8$)	15.63M	0.42%	62.3 TFLOPs (33.9%)
Ours ($r = 8$)	8.40M	0.22%	62.0 TFLOPs (33.7%)

5.6. More Results on Real-World Datasets

To further demonstrate the generalization capability of our proposed method, we conduct additional experiments on a real-world unlearning scenario dataset. Following the setup described by Liu et al. [39], we adopt a more realistic scenario where the knowledge to be unlearned is inherent in the target model and the training data are unknown. This dataset identifies several real-world individuals with Wikipedia entries, along with inappropriate responses from Llama3-8B-Instruct model and golden answers for each individual. For detailed information on the dataset composition, sample size, annotation protocol, data sources, and compliance considerations, we refer readers to the original work [39].

For this evaluation, we employ the Qwen3-4B-Instruct model as the base model, maintaining all other experimental settings identical to those used in the main experiments (Section 5.1). This includes the same hyperparameters, training procedures, and evaluation metrics (MU and FP) to ensure fair comparison. The real-world dataset provides a more challenging testbed as it involves unlearning factual knowledge about actual individuals that has been deeply embedded in the pre-trained model, rather than synthetic or artificially injected information.

The results on the real-world dataset further validate the effectiveness and generalization capability of our proposed method. As shown in Table 6, our method achieves the highest average score of 0.620 (averaged across all 12 MU and FP metric values), surpassing all baseline methods including DPO (0.617), GA + KL (0.614), and GD (0.612). More importantly, our method maintains consistently high and stable MU scores throughout the CLU process (0.81, 0.79, 0.76, 0.74, 0.71, 0.68), demonstrating robust resistance to catastrophic forgetting even when dealing with deeply embedded factual knowledge. In contrast, LoRA exhibits severe performance degradation with MU scores dropping to 0.17, 0.29, and 0.07 in later tasks. While the FP scores remain moderate (0.42–0.55), consistent with our controlled-forgetting design, the combination of highest average score and exceptional stability confirms that our method generalizes well to real-world scenarios involving interleaved learning and unlearning under controlled benchmark conditions.

Table 6. Performance comparison on real-world dataset with Qwen3-4B-Instruct.

Method	UL_1		CL_1		UL_2		CL_2		UL_3		CL_3		Average
	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	MU	FP	
GA	0.69	0.57	0.75	0.45	0.51	0.6	0.69	0.56	0.56	0.66	0.64	0.59	0.606
GD	0.69	0.47	0.74	0.48	0.73	0.51	0.71	0.56	0.63	0.59	0.61	0.62	0.612
GA + KL	0.76	0.4	0.78	0.55	0.68	0.54	0.69	0.54	0.57	0.65	0.59	0.62	0.614
NPO	0.69	0.52	0.77	0.51	0.63	0.54	0.7	0.53	0.59	0.6	0.62	0.59	0.601
DPO	0.69	0.51	0.77	0.51	0.72	0.54	0.71	0.51	0.6	0.6	0.6	0.58	0.617
LoRA	0.8	0.39	0.71	0.42	0.17	0.87	0.29	0.83	0.07	0.95	0.33	0.72	0.546
Ours	0.81	0.42	0.79	0.45	0.76	0.48	0.74	0.51	0.71	0.54	0.68	0.55	0.620

6. Discussion

This work presents a unified knowledge management framework that integrates continual learning and machine unlearning in large language models under a single information-theoretic perspective. Our experimental results on controlled interleaved benchmarks (six sequential tasks) demonstrate that the proposed method achieves the best average score (0.573 on Llama3-8B-Instruct) and exceptional stability (variance $\pm$ 0.0089 across seeds) across sequential tasks, outperforming existing baseline methods on both synthetic (TOFU) and real-world benchmarks.

6.1. Interpretation of Key Findings

The superior performance of our method can be attributed to three synergistic design principles derived from the drift-aware conceptual framework (where distributional shifts are characterized via KL as a design principle): freezing the LoRA projection matrix A constrains updates to a shared low-dimensional subspace reducing inter-task interference, sparse masking on B protects important large-magnitude parameters while allowing selective updates to less critical ones, and orthogonal gradient projection suppresses destructive interference with previously learned directions. Compared to prior work, our framework differs fundamentally in its knowledge-centric formulation—while traditional continual learning methods such as EWC [10] and MAS [11] focus on parameter importance estimation in small-scale discriminative models, and existing unlearning methods like gradient ascent [20] and NPO [22] prioritize rapid maximal forgetting at the cost of collateral damage and instability, our unified framework treats learning and unlearning as complementary operations under the same optimization principle, deliberately emphasizing controlled low-collateral forgetting with stable knowledge retention for reliable deployments in interleaved task scenarios. Ablation studies reveal that structural constraints (freezing A and sparse masking on B) are more critical than orthogonal gradient projection alone, aligning with recent findings that parameter protection and magnitude-based selective updating play a more dominant role than gradient-based regularization in large-scale models [3].

6.2. Limitations and Future Directions

Despite promising results on controlled interleaved benchmarks (6 sequential tasks), our framework has several concrete limitations warranting further investigation:

Controlled vs. maximal forgetting trade-off. Our design prioritizes controlled low-collateral forgetting over maximal erasure (FP: 0.38-0.46 vs. GA: 0.51–0.77), making it well-suited for gradual policy-driven knowledge removal but less suitable for emergency privacy scenarios requiring immediate complete erasure. Future work could explore adaptive forgetting strategies with switchable objectives balancing controllability and erasure strength.

Scalability to longer task sequences. Our evaluation covers six interleaved tasks, leaving scalability to significantly longer sequences (e.g., 50+ tasks) unexplored.

Analytical Scaling Behavior. Beyond computational cost, it is important to analyze how the framework's *effectiveness*—not just its efficiency—scales with task count. We derive predictions from three complementary perspectives.

Cumulative drift growth. Theorem A1 bounds per-step KL drift by $C_1\|\Delta\theta\|_2^2 + C_2\|\Delta\theta\|_2$. After T sequential tasks with per-task updates $\{\delta_t\}_{t=1}^{T}$, the total displacement is $\Delta\theta_T = \sum_t \delta_t$. Under orthogonal projection, $\delta_i^\top \delta_j = 0$ for $i \neq j$, so $\|\Delta\theta_T\|_2^2 = \sum_t \|\delta_t\|_2^2 = O(T)$ and the cumulative KL bound grows as $O(T)$. Without orthogonality, constructive interference can yield $\|\Delta\theta_T\|_2 = O(T)$ and KL bound $O(T^2)$ in the worst case. This provides a theoretical rationale for why orthogonal projection becomes increasingly important in longer sequences—it reduces cumulative drift scaling from quadratic to linear—consistent with

our ablation results showing disproportionate Orthogonal Gradient Projection (OGP) benefit in later tasks (UL_3 MU: 0.44→0.55 when adding OGP to a+b).

Direction space saturation. OGP stores one direction per task in the space of trainable parameters. With sparse masking at sparsity s, the effective dimension is $(1-s)\cdot r\cdot d_{\text{out}}$. At $s=0.9$, $r=32$, and typical $d_{\text{out}}=4096$, this yields ∼13,000 theoretical directions before the orthogonal complement vanishes. While far beyond practical horizons, numerical accumulation and non-linear gradient dynamics will reduce effective capacity, motivating direction compression strategies.

Empirical trend extrapolation. On Llama3-8B-Instruct, our method exhibits an approximately linear MU decline of ∼2.3% per task across the six-task sequence (0.81→0.67). If this rate persisted—a strong assumption, as interference patterns depend on task similarity and distribution overlap—MU would reach 0.50 around task 14. This suggests the current framework without consolidation is most suited for medium-length sequences (∼10–20 tasks), with longer horizons requiring periodic adapter merging and mask refresh as discussed below.

A key concern is the computational and memory footprint of orthogonal gradient projection (OGP) as the number of tasks grows.

Direction set growth and projection cost. Let d denote the number of trainable adapter parameters being projected (in our case, the flattened B parameters after masking), and let $m=t-1$ be the number of stored directions. The naive projection $g_t^{\perp}=g_t-\sum_{i=1}^{m}(g_t^{\top}v_i)v_i$ requires: (i) computing m inner products, each $O(d)$; and (ii) accumulating m scaled vectors, also $O(d)$. Thus, the per-step compute is $O(md)=O(td)$, and storing all directions costs $O(md)=O(td)$ in memory. Over an entire task with T steps, the total compute is $O(Tmd)=O(Ttd)$. In practice, this can become a bottleneck for long horizons because projection is applied at every optimization step (not just once per task); therefore, the wall-clock overhead scales roughly linearly with both task count and the number of gradient steps.

Matrix form and memory bandwidth. If we stack directions as $V\in\mathbb{R}^{m\times d}$, then $g_t^{\perp}=g_t-V^{\top}(Vg_t)$ (assuming approximately orthonormal rows). This formulation highlights that OGP is primarily limited by two matrix–vector multiplications, with memory traffic proportional to md. As m grows, GPU memory bandwidth and cache locality become limiting factors even when the arithmetic cost is moderate.

Practical mitigation for scalability. To keep OGP scalable, one may (a) retain only a window of recent directions (size $k\ll t$), reducing compute/memory to $O(kd)$; (b) compress directions into a low-dimensional principal subspace of rank $r\ll t$ via incremental Singular Value Decomposition (SVD)/online Principal Component Analysis (PCA), yielding $O(rd)$ storage and $O(rd)$ per-step projection; or (c) use randomized sketching to approximate Vg_t with lower memory overhead. These approaches trade full-history orthogonality for scalable approximate constraints and remain to be validated in extended-horizon CLU settings.

In addition to OGP, two further bottlenecks may arise: sparsity capacity exhaustion—with 90% sparsity, only a small fraction of parameters are available for adaptation, and repeated task-specific masking may eventually exhaust unused capacity, requiring periodic mask refresh or dynamic capacity reallocation strategies; cumulative drift accumulation—accumulated parameter deviations may gradually shift representations away from the pretrained reference distribution, potentially destabilizing retained knowledge. Addressing these issues may require periodic consolidation (merging adapters and resetting masks) or dynamic rank adjustment. These remain promising but unvalidated directions for future work.

Task boundary assumptions. Our framework assumes explicit task boundaries and manually constructed data partitions (retain/learn/forget sets). Extending to task-free continual learning settings with automatic boundary detection or gradual distribution

shifts represents an important direction, though it introduces additional challenges in identifying when to apply learning vs. unlearning objectives without supervision.

These limitations delineate the scope of our current validation and highlight concrete technical challenges that future research can address to extend the framework toward longer-horizon deployment scenarios.

7. Conclusions

In this work, we present a parameter-efficient knowledge management framework where continual learning and machine unlearning—while employing distinct task-specific objectives (SFT for learning, GA for unlearning)—are integrated through a shared retention-controlled parameter evolution mechanism, with KL divergence serving as the design principle governing drift-aware structural constraints. We develop a practical implementation combining three synergistic mechanisms—freezing the LoRA projection matrix, magnitude-based sparse masking, and orthogonal gradient projection—that realize drift control entirely through parameter-space operations without modifying the base model. Extensive experiments on synthetic (TOFU) and real-world benchmarks using 4B- and 8B-scale language models demonstrate that our framework achieves the best average score (0.573 on Llama3-8B-Instruct) and exceptional stability (variance $\pm$ 0.0089 across five random seeds), with consistently high model utility and controlled forget-set response deviation that prioritizes low-collateral behavioral shifts over maximal output divergence. Token-level distributional drift analysis further validates that the parameter-space constraints effectively bound KL divergence on retained knowledge, and that this distributional stability directly underlies the observed behavioral-level stability–plasticity balance. It is important to emphasize that our evaluation captures behavioral unlearning in a controlled benchmark setting—measuring output-level changes rather than certifying complete knowledge elimination or privacy guarantees. While promising, concrete scalability challenges remain: direction set growth introduces linear computational overhead, sparsity capacity may exhaust under extended task sequences, and task boundary assumptions limit applicability to gradual distribution shifts—these represent well-defined technical directions for future research. This work provides a practical parameter-efficient recipe and a drift-aware design principle validated on controlled interleaved benchmarks (six sequential tasks), contributing both practical tools and theoretical understanding toward systematic and controllable knowledge dynamics in large language models.

Author Contributions: Conceptualization, J.L. and L.L.; methodology, J.L.; software, J.L.; validation, J.L. and L.L.; formal analysis, J.L.; investigation, J.L.; resources, L.L. and D.Z.; data curation, J.L.; writing—original draft preparation, J.L.; writing—review and editing, L.L. and D.Z.; visualization, J.L.; supervision, L.L. and D.Z.; project administration, L.L.; funding acquisition, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA0480301, in part by the Major Project of the National Social Science Fund of China under Grant 25&ZD043, and by the National Natural Science Foundation of China under Grant 62206293.

Data Availability Statement: The datasets used in this study are publicly available and can be accessed at the following GitHub repository: https://github.com/Langjiaqi/dataset_clu (accessed on 20 February 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Parameter-Space Drift Control as a KL Approximation

In the unified distributional formulation (Section 3.2), retention is expressed as a drift-control regularizer $\mathcal{R}_{\text{retain}}(\theta; \theta_{\text{ref}}, \mathcal{D}^R)$ that constrains distributional changes on the retain set via token-level KL divergence. In our implementation, however, we do **not explicitly compute** this KL term on $\mathcal{D}^R$. Instead, we realize retention **implicitly** via parameter-space constraints that approximate drift control. Below, we provide a formal derivation establishing the approximation guarantees.

Formal Approximation Framework. We establish the connection between distributional drift and parameter-space constraints through a series of formal assumptions, lemmas, and theorems.

Assumption A1 (Model Smoothness). *The model output function $f_\theta : \mathcal{X} \times \Theta \to \mathbb{R}^V$ (where $\mathcal{X}$ is the input space, Θ is the parameter space, and V is the vocabulary size) is twice continuously differentiable with respect to θ. Moreover, there exist constants $L_1, L_2 > 0$ such that for all $x \in \mathcal{X}$ and $\theta, \theta' \in \Theta$ with $\|\theta - \theta'\|_2 \leq \epsilon$ (where ϵ is the learning rate bound):*

$$\|\nabla_\theta f_\theta(x)\|_{\text{op}} \leq L_1, \quad \|\nabla^2_\theta f_\theta(x)\|_{\text{op}} \leq L_2, \tag{A1}$$

where $\|\cdot\|_{\text{op}}$ denotes the operator norm.

Lemma A1 (KL-Logit Bound). *Let $p_{\theta_{\text{ref}}}(\cdot \mid s_{<t})$ and $p_\theta(\cdot \mid s_{<t})$ be the softmax distributions over vocabulary V induced by logit vectors $z_{\text{ref}} = f_{\theta_{\text{ref}}}(s_{<t})$ and $z = f_\theta(s_{<t})$, respectively. Then the token-level KL divergence satisfies*

$$D_{\text{KL}}\big(p_{\theta_{\text{ref}}}(\cdot \mid s_{<t}) \,\big\|\, p_\theta(\cdot \mid s_{<t})\big) \leq \frac{1}{2V}\|z_{\text{ref}} - z\|_2^2 + \|z_{\text{ref}} - z\|_\infty \cdot C_{\text{KL}}, \tag{A2}$$

where $C_{\text{KL}} = \log V$ is a constant depending on the vocabulary size.

Proof. By Pinsker's inequality and properties of softmax perturbation under bounded logit changes, combined with the Lipschitz continuity of the softmax function, the KL divergence can be bounded by a quadratic term in the ℓ_2 norm plus a linear term in the ℓ_∞ norm of the logit perturbation. For detailed derivation, see [40]. □

Lemma A2 (Parameter-Logit Approximation). *Under Assumption A1, for parameter update $\Delta\theta = \theta - \theta_{\text{ref}}$ with $\|\Delta\theta\|_2 \leq \epsilon$, the logit change at input $s_{<t}$ satisfies*

$$f_\theta(s_{<t}) = f_{\theta_{\text{ref}}}(s_{<t}) + J_{\theta_{\text{ref}}}(s_{<t})^\top \Delta\theta + \mathcal{O}(\|\Delta\theta\|_2^2), \tag{A3}$$

where $J_{\theta_{\text{ref}}}(s_{<t}) = \nabla_\theta f_{\theta_{\text{ref}}}(s_{<t}) \in \mathbb{R}^{d \times V}$ is the Jacobian matrix. Consequently,

$$\|f_\theta(s_{<t}) - f_{\theta_{\text{ref}}}(s_{<t})\|_2 \leq L_1\|\Delta\theta\|_2 + \frac{L_2}{2}\|\Delta\theta\|_2^2. \tag{A4}$$

Proof. By Taylor expansion of f_θ around θ_{ref} and applying the bounds from Assumption A1, we obtain the first-order approximation with an explicit remainder term. Taking norms and applying the triangle inequality yields the stated bound. □

Theorem A1 (Parameter-Space Drift Control). *Under Assumptions A1, for parameter update $\Delta\theta$ with $\|\Delta\theta\|_2 \leq \epsilon$, the average token-level KL divergence on the retain set $\mathcal{D}^R$ satisfies*

$$\mathbb{E}_{s\sim\mathcal{D}^R}\left[\frac{1}{|s|}\sum_{t=1}^{|s|} D_{\mathrm{KL}}\big(p_{\theta_{\mathrm{ref}}}(\cdot \mid s_{<t}) \,\big\|\, p_\theta(\cdot \mid s_{<t})\big)\right] \leq C_1\|\Delta\theta\|_2^2 + C_2\|\Delta\theta\|_2, \tag{A5}$$

where $C_1 = \frac{L_1^2}{2V} + \frac{L_2^2\epsilon}{4V}$ and $C_2 = (L_1 + \frac{L_2\epsilon}{2})C_{\mathrm{KL}}$ are constants determined by model properties and hyperparameters.

Proof. Combining Lemmas A1 and A2, for any input $s_{<t}$ on the retain set, we have

$$\begin{aligned} &D_{\mathrm{KL}}\big(p_{\theta_{\mathrm{ref}}}(\cdot \mid s_{<t}) \,\big\|\, p_\theta(\cdot \mid s_{<t})\big) \\ &\leq \frac{1}{2V}\|z_{\mathrm{ref}} - z\|_2^2 + \|z_{\mathrm{ref}} - z\|_\infty \cdot C_{\mathrm{KL}} \qquad \text{(A6)} \\ &\leq \frac{1}{2V}\left(L_1\|\Delta\theta\|_2 + \frac{L_2}{2}\|\Delta\theta\|_2^2\right)^2 + \left(L_1\|\Delta\theta\|_2 + \frac{L_2}{2}\|\Delta\theta\|_2^2\right)C_{\mathrm{KL}}. \qquad \text{(A7)} \end{aligned}$$

Under the constraint $\|\Delta\theta\|_2 \leq \epsilon$ (with ϵ sufficiently small), expanding the squared term and retaining dominant terms yields

$$D_{\mathrm{KL}}\big(p_{\theta_{\mathrm{ref}}}(\cdot \mid s_{<t}) \,\big\|\, p_\theta(\cdot \mid s_{<t})\big) \leq \left(\frac{L_1^2}{2V} + \frac{L_2^2\epsilon}{4V}\right)\|\Delta\theta\|_2^2 + \left(L_1 + \frac{L_2\epsilon}{2}\right)C_{\mathrm{KL}}\|\Delta\theta\|_2. \tag{A8}$$

Taking expectation over sequences $s \sim \mathcal{D}^R$ and averaging over tokens completes the proof. □

Corollary A1 (Structural Constraint Realization). *Theorem A1 implies that controlling distributional drift $\mathcal{R}_{\mathrm{retain}}$ can be achieved by bounding $\|\Delta\theta\|_2$ and constraining the effective direction of $\Delta\theta$. Our three structural mechanisms realize this as follows:*

1. ***Localization (Freezing A):*** *Restricting updates to $\Delta\theta = A\Delta B$ with frozen $A \in \mathbb{R}^{d_{\mathrm{in}}\times r}$ and $r \ll d_{\mathrm{in}}$ reduces the effective parameter space dimension from d to $O(r \cdot d_{\mathrm{out}})$, yielding $\|\Delta\theta\|_F^2 = \|A\Delta B\|_F^2 \leq \|A\|_F^2\|\Delta B\|_F^2$, thereby bounding update magnitude via the fixed subspace defined by A.*
2. ***Selective Protection (Sparse Masking on B):*** *Applying element-wise mask M_t with sparsity s enforces $\|\Delta B\|_0 \leq (1-s)\cdot|B|$, where $\|\cdot\|_0$ denotes the number of non-zero elements. By protecting top-s percentile parameters (largest magnitude entries critical to retained capabilities), we further constrain $\|\Delta B\|_F \leq \sqrt{1-s}\cdot\|\Delta B_{unmasked}\|_F$, reducing perturbation magnitude.*
3. ***Direction Control (Orthogonal Projection):*** *Projecting gradients to be orthogonal to previous task directions $\{v_i\}_{i=1}^{t-1}$ ensures $\Delta\theta^\top v_i = 0$ for all $i < t$, minimizing alignment with critical directions for retained knowledge and thereby reducing the effective impact on $\mathcal{D}^R$ in directions where $\|J_{\theta_{\mathrm{ref}}}(s_{<t})^\top v_i\|_2$ is large.*

Remark A1. *Together, Theorem A1 and Corollary A1 establish that our parameter-space structural constraints provide a principled approximation to the distributional drift control $\mathcal{R}_{\mathrm{retain}}$ in Equation (4), with explicit approximation bounds. This justifies our implementation strategy as* KL-inspired parameter-space drift control*—motivated by distributional considerations but realized entirely through parameter-space operations with formal guarantees.*

Appendix B. Baseline Method Details

Let π_θ denote the language model parameterized by θ. Each sample is a QA pair (q, a), where q is the question (prompt) and $a = (a_1, \ldots, a_T)$ is the answer token sequence. We define the token-level negative log-likelihood (NLL) loss on answer tokens as

$$\ell(q, a; \theta) \;=\; -\frac{1}{T}\sum_{t=1}^{T} \log \pi_\theta(a_t \mid q, a_{<t}), \tag{A9}$$

where T is the length of the answer sequence a, a_t is the t-th token, $a_{<t}$ denotes tokens before position t, and $\pi_\theta(a_t \mid q, a_{<t})$ is the model's predicted probability for the token at position t given the question q and preceding tokens $a_{<t}$. Given a dataset D (a set of QA pairs), the averaged training loss is

$$\mathcal{L}(D; \theta) \;=\; \frac{1}{|D|}\sum_{(q,a)\in D} \ell(q, a; \theta). \tag{A10}$$

Let D_f and D_r denote the forget set and retain set, respectively.

Gradient Ascent (GA).

GA aims to "forget" by increasing the loss on the forget set. Equivalently, if we implement unlearning via gradient descent, GA minimizes

$$\mathcal{L}_{\mathrm{GA}}(\theta) \;=\; -\,\mathcal{L}(D_f; \theta), \tag{A11}$$

which corresponds to performing gradient ascent on $\mathcal{L}(D_f; \theta)$.

GA + GD (Gradient Difference).

GA + GD mitigates the utility degradation of GA by combining (i) gradient ascent on D_f and (ii) gradient descent on D_r:

$$\mathcal{L}_{\mathrm{GA+GD}}(\theta) \;=\; -\,\mathcal{L}(D_f; \theta) \;+\; \mathcal{L}(D_r; \theta). \tag{A12}$$

GA + KL (KL-regularized GA).

GA + KL further constrains distributional drift by adding a KL regularization term between the unlearned model and a reference model. Let π_{θ_0} be a reference model (e.g., the pre-unlearning model), and let $s = [q, a]$ be the concatenated sequence. Denote by $s_{<t}$ the prefix up to position $t-1$, and by $\pi_\theta(\cdot \mid s_{<t})$ the next-token distribution. A commonly used KL-regularized objective is

$$\mathcal{L}_{\mathrm{GA+KL}}(\theta) = -\,\mathcal{L}(D_f; \theta) \;+\; \lambda\,\mathcal{R}_{\mathrm{KL}}(\theta), \tag{A13}$$

$$\mathcal{R}_{\mathrm{KL}}(\theta) = \frac{1}{|D_r|}\sum_{s\in D_r} \frac{1}{|s|}\sum_{t=2}^{|s|} D_{\mathrm{KL}}\Big(\pi_{\theta_0}(\cdot \mid s_{<t}) \,\Big\|\, \pi_\theta(\cdot \mid s_{<t})\Big), \tag{A14}$$

where $\lambda > 0$ controls the strength of the regularization.

Negative Preference Optimization (NPO).

NPO reduces the model's confidence on forget-set answers via a negative-preference objective. Given $(x, y) \in D_f$ (here x is the prompt and y is the target response to be forgotten), a reference model π_{ref}, and inverse temperature $\beta > 0$, the NPO loss is

$$\mathcal{L}_{\text{NPO},\beta}(\theta) = -\frac{2}{\beta}\,\mathbb{E}_{(x,y)\sim D_f}\left[\log\sigma\left(-\beta\log\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right)\right] \tag{A15}$$

$$= \frac{2}{\beta}\,\mathbb{E}_{(x,y)\sim D_f}\left[\log\left(1+\left(\frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}\right)^{\beta}\right)\right], \tag{A16}$$

where $\sigma(\cdot)$ is the sigmoid function.

Direct Preference Optimization (DPO).

DPO is originally designed for paired human preferences; for unlearning, it can be adapted by constructing preference pairs that encourage neutral/non-target responses. Given preference pairs (x, y_w, y_l) where y_w is the preferred response (e.g., a neutral "I don't know" style answer) and y_l is the dispreferred response (e.g., the original answer to be forgotten), the DPO objective is

$$\mathcal{L}_{\text{DPO},\beta}(\theta) = -\frac{1}{\beta}\,\mathbb{E}_{(x,y_w,y_l)}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right], \tag{A17}$$

where $\mathbb{E}_{(x,y_w,y_l)}$ denotes the expectation over preference pairs sampled from the dataset.

References

1. Shi, H.; Xu, Z.; Wang, H.; Qin, W.; Wang, W.; Wang, Y.; Wang, Z.; Ebrahimi, S.; Wang, H. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.* **2025**, *58*, 1–42. [CrossRef]
2. Liu, S.; Yao, Y.; Jia, J.; Casper, S.; Baracaldo, N.; Hase, P.; Yao, Y.; Liu, C.Y.; Xu, X.; Li, H.; et al. Rethinking machine unlearning for large language models. *Nat. Mach. Intell.* **2025**, *7*, 181–194. [CrossRef]
3. Wang, X.; Chen, T.; Ge, Q.; Xia, H.; Bao, R.; Zheng, R.; Zhang, Q.; Gui, T.; Huang, X.J. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*; Association for Computational Linguistics: Kerrville, TX, USA, 2023; pp. 10658–10671.
4. He, J.; Guo, H.; Zhu, K.; Zhao, Z.; Tang, M.; Wang, J. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. *arXiv* **2024**, arXiv:2411.06171. [CrossRef]
5. Gao, C.; Wang, L.; Ding, K.; Weng, C.; Wang, X.; Zhu, Q. On large language model continual unlearning. *arXiv* **2024**, arXiv:2407.10223.
6. Liu, B.; Liu, Q.; Stone, P. Continual learning and private unlearning. In *Proceedings of the Conference on Lifelong Learning Agents*; PMLR: Cambridge, MA, USA, 2022; pp. 243–254.
7. Chatterjee, R.; Chundawat, V.; Tarun, A.; Mali, A.; Mandal, M. A unified framework for continual learning and unlearning. *arXiv* **2024**, arXiv:2408.11374. [CrossRef]
8. Huang, Z.; Cheng, X.; Zhang, J.; Zheng, J.; Wang, H.; He, Z.; Li, T.; Huang, X. A unified gradient-based framework for task-agnostic continual learning-unlearning. *arXiv* **2025**, arXiv:2505.15178.
9. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-rank adaptation of large language models. *Int. Conf. Learn. Represent.* **2022**, *1*, 3.
10. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef] [PubMed]
11. Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision (ECCV)*; Springer: Cham, Switzerland, 2018; pp. 139–154.
12. Zenke, F.; Poole, B.; Ganguli, S. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2017; pp. 3987–3995.
13. Chaudhry, A.; Dokania, P.K.; Ajanthan, T.; Torr, P.H. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*; Springer: Cham, Switzerland, 2018; pp. 532–547.

14. Guo, C.; Zhao, B.; Bai, Y. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*; Springer: Cham, Switzerland, 2022; pp. 181–195.
15. Feldman, D. Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*; Springer: Cham, Switzerland, 2019; pp. 23–44.
16. Wang, T.; Zhu, J.Y.; Torralba, A.; Efros, A.A. Dataset distillation. *arXiv* **2018**, arXiv:1811.10959.
17. Yu, R.; Liu, S.; Wang, X. Dataset distillation: A comprehensive review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *46*, 150–170. [CrossRef]
18. Ahn, H.; Cha, S.; Lee, D.; Moon, T. Uncertainty-based continual learning with adaptive regularization. *arXiv* **2019**, arXiv:1905.11614. [CrossRef]
19. Jin, H.; Kim, E. Helpful or harmful: Inter-task association in continual learning. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2022; pp. 519–535.
20. Maini, P.; Feng, Z.; Schwarzschild, A.; Lipton, Z.C.; Kolter, J.Z. Tofu: A task of fictitious unlearning for LLMs. *arXiv* **2024**, arXiv:2401.06121. [CrossRef]
21. Jang, J.; Yoon, D.; Yang, S.; Cha, S.; Lee, M.; Logeswaran, L.; Seo, M. Knowledge unlearning for mitigating privacy risks in language models. In *61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Kerrville, TX, USA, 2023; pp. 14389–14408.
22. Zhang, R.; Lin, L.; Bai, Y.; Mei, S. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv* **2024**, arXiv:2404.05868. [CrossRef]
23. Fan, C.; Liu, J.; Lin, L.; Jia, J.; Zhang, R.; Mei, S.; Liu, S. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. *arXiv* **2024**, arXiv:2410.07163. [CrossRef]
24. Cha, S.; Cho, S.; Hwang, D.; Lee, M. Towards robust and parameter-efficient knowledge unlearning for LLMs. *arXiv* **2024**, arXiv:2408.06621.
25. Russinovich, M.; Salem, A. Obliviate: Efficient unmemorization for protecting intellectual property in large language models. *arXiv* **2025**, arXiv:2502.15010. [CrossRef]
26. Liu, Z.; Dou, G.; Tan, Z.; Tian, Y.; Jiang, M. Towards safer large language models through machine unlearning. *arXiv* **2024**, arXiv:2402.10058. [CrossRef]
27. Ishibashi, Y.; Shimodaira, H. Knowledge sanitization of large language models. *arXiv* **2023**, arXiv:2309.11852.
28. Liu, Y.; Zhang, Y.; Jaakkola, T.; Chang, S. Revisiting Who's Harry Potter: Towards Targeted Unlearning from a Causal Intervention Perspective. *arXiv* **2024**, arXiv:2407.16997.
29. Xu, H.; Zhao, N.; Yang, L.; Zhao, S.; Deng, S.; Wang, M.; Hooi, B.; Oo, N.; Chen, H.; Zhang, N. Relearn: Unlearning via learning for large language models. *arXiv* **2025**, arXiv:2502.11190. [CrossRef]
30. Shibata, T.; Irie, G.; Ikami, D.; Mitsuzumi, Y. Learning with Selective Forgetting. *Int. Jt. Conf. Artif. Intell.* **2021**, *3*, 4.
31. Wang, Z.; Bi, B.; Pentyala, S.K.; Ramnath, K.; Chaudhuri, S.; Mehrotra, S.; Mao, X.B.; Asur, S.; Cheng, N. A comprehensive survey of LLM alignment techniques: RLHF, RLAIF, PPO, DPO and more. *arXiv* **2024**, arXiv:2407.16216. [CrossRef]
32. Izzo, Z.; Smart, M.A.; Chaudhuri, K.; Zou, J. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*; PMLR: Cambridge, MA, USA, 2021; pp. 2008–2016.
33. Qiao, J.; Zhang, Z.; Tan, X.; Qu, Y.; Zhang, W.; Han, Z.; Xie, Y. Gradient projection for continual parameter-efficient tuning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2025**, *47*, 9316–9329. [CrossRef]
34. Yao, J.; Chien, E.; Du, M.; Niu, X.; Wang, T.; Cheng, Z.; Yue, X. Machine unlearning of pre-trained large language models. *arXiv* **2024**, arXiv:2402.15159. [CrossRef]
35. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*; Association for Computational Linguistics: Kerrville, TX, USA, 2004; pp. 74–81.
36. Yuan, X.; Pang, T.; Du, C.; Chen, K.; Zhang, W.; Lin, M. A closer look at machine unlearning for large language models. *arXiv* **2024**, arXiv:2410.08109. [CrossRef]
37. Reimers, N.; Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv* **2019**, arXiv:1908.10084. [CrossRef]
38. Sileo, D. tasksource: A Dataset Harmonization Framework for Streamlined NLP Multi-Task Learning and Evaluation. *arXiv* **2023**, arXiv:2301.05948. [CrossRef]
39. Liu, Z.; Zhu, T.; Tan, C.; Chen, W. Learning to refuse: Towards mitigating privacy risks in LLMs. In *31st International Conference on Computational Linguistics*; Association for Computational Linguistics: Kerrville, TX, USA, 2025; pp. 1683–1698.
40. Pinsker, M.S. Some mathematical questions of theory of information transmission. *Probl. Inf. Transm.* **2007**, *43*, 380–392. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.