KUnBR: Knowledge Density-Guided Unlearning via Blocks Reinsertion

Anonymous ACL submission

Abstract

Machine unlearning, which selectively removes specific knowledge from a pre-trained model without retraining from scratch, is crucial for addressing privacy, regulatory compliance, and ethical concerns in Large Language Models (LLMs). However, existing unlearning methods usually fail to thoroughly erase targeted knowledge, leaving residual information that can be easily recovered. To address these limitations, we propose **K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion (KUnBR), a novel approach that enhances the degree of forgetting by first identifying knowledge-rich layers and then thoroughly eliminating the targeted knowledge. Our method introduces knowledge density estimation to quantify and locate layers containing the most knowledge, enabling precise unlearning. Additionally, we design a layer re-insertion strategy that extracts and re-inserts knowledge-rich layers into the original, bypassing gradient obstruction caused by masked layers and ensuring effective gradient propagation during unlearning. This strategy significantly reduces the model's vulnerability to knowledge recovery attacks. several unlearning datasets and utility benchmark (RKWU) demonstrate that KUnBR achieves state-of-the-art forgetting performance while maintaining model utility, generalizing across multiple strong unlearning methods[1].

1 Introduction

Machine unlearning refers to the process of selectively removing specific subsets of knowledge, such as privacy-sensitive or harmful content, from a pre-trained model without retraining it from scratch (Carlini et al., 2021; Xu et al., 2024). This task has become increasingly crucial for the development of large language models (LLMs) (OpenAI, 2024; AI@Meta, 2024; Anthropic, 2024; Guo et al., 2025), as it addresses growing concerns around data privacy (Carlini et al., 2021; Huang et al., 2022; Lee et al., 2024; Liu et al., 2024), regulatory compliance (Voigt and Bussche, 2017), and the ethical issue of AI systems (Bender et al., 2021). Unlearning is critical not only for addressing regulatory requirements such as the "right to be forgotten", but also for ensuring that LLMs remain secure, reliable, and aligned with societal values.

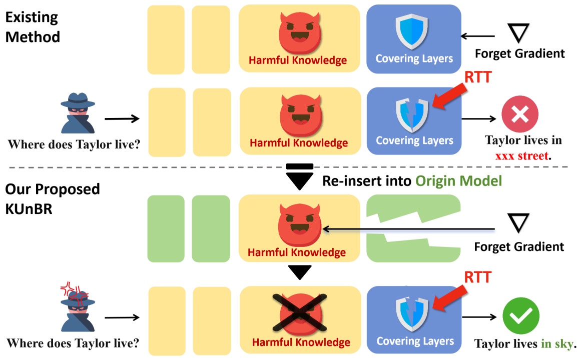

Figure 1: Existing unlearning methods fail to completely remove harmful knowledge from models due to the presence of covering layers. Our proposed KUnBR achieves better unlearning by reinserting layers with high knowledge density into the original model, thereby disrupting the covering layers.

Prior research has explored different unlearning methodologies, such as Gradient Ascent (GA) (Jang et al., 2022; Eldan and Russinovich, 2023) approaches which unlearn the knowledge by increasing the loss when outputting harmful answers, Gradient Difference (GD) (Liu et al., 2022) methods that conduct gradient ascent on the forget dataset and gradient descent on the retain dataset, and Representation Misdirection for Unlearning (RMU) (Li et al., 2024) strategies that directly adjust the intermediate representation to unlearning. These methods always utilize two distinct datasets: a *forget set*, which contains the information to be

[1] Code is available at `https://anonymous.4open.science/r/KUnBR-CF44`

removed, and a *retain set*, which preserves the model's general knowledge and performance on unrelated tasks (Bourtoule et al., 2021).

Despite the progress made by these methods, two significant limitations persist. First, even after applying existing unlearning techniques, a substantial amount of the targeted knowledge often remains in the LLM, indicating that the degree of forgetting is still insufficient. Second, the forgotten knowledge can be easily recovered using simple methods. For instance, the **Re**training on **T** (RTT) (Deeb and Roger, 2025) approach demonstrates that minimal retraining on a subset of the forget set can restore most of the supposedly erased knowledge, highlighting the fragility of current unlearning strategies. Empirical analyses of these issues (Hong et al., 2024) suggest that the root cause lies in the superficial nature of existing unlearning methods. Rather than genuinely erasing the targeted knowledge, existing unlearning methods often rely on masking or obfuscating certain model parameters, which merely *prevents the model from outputting* the undesired knowledge without truly *eliminating it from the model*'s internal representations. This fundamental limitation underscores the need for more robust and thorough unlearning methods in the field of LLMs.

To address these limitations, we propose **K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion (KUnBR), a fine-grained unlearning framework designed to enhance the degree of forgetting, thereby thoroughly eliminating the undesired knowledge from the parameters. We first introduce a **knowledge density estimation** method to quantify the knowledge contained in layers of LLM and identify the layers that contain the most undesired knowledge. By calculating the absolute value of gradients associated with the forget set, knowledge density estimation enables precise targeting of layers containing high-density knowledge. To achieve the thorough elimination of forgotten knowledge, rather than having the model only appear to forget knowledge at the output level, we design a **re-insertion** strategy, where knowledge-rich blocks selected based on knowledge density estimation, are extracted from unlearned LLM and re-inserted into the original LLM without conducting the unlearning training. We then apply the unlearning method to train this "grafted" model, which contains the re-inserted layers, with a focus on deeper removal of the undesired knowledge left due to the constraint of cover layers. By bypassing the obstruction of covering layers, this strategy ensures more effective gradient propagation and enhances the model's ability to forget. Additionally, it significantly reduces the vulnerability of the model to attacks like RTT, which exploit the residual knowledge left by conventional unlearning methods. Extensive experiments conducted on WMDP-Deduped, Years, Random Birthdays and RKWU benchmark datasets demonstrate that our method achieves state-of-the-art performance, since it can remove knowledge more thoroughly and more effectively suppress knowledge recovery caused by RTT attack methods.

Our contributions are summarized as follows:

- We propose **K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion (KUnBR), a novel method that addresses incomplete knowledge forgetting in existing approaches through a layer re-insertion strategy.
- We introduce knowledge density estimation, which can identify and prioritize knowledge-rich layers in LLMs for more effective unlearning.
- We design a layer re-insertion strategy to ensure unlearning gradients propagate effectively, overcoming the limitations of gradient obstruction.
- Extensive experiments demonstrate that KUnBR generalizes across multiple SOTA unlearning methods, achieving superior forgetting performance while maintaining model utility.

2 Related Work

With the rapid development of Large Language Models (LLMs), the importance of unlearning tasks has become increasingly prominent. During the pre-training process where these models ingest massive amounts of information, they may incorporate harmful content (Carlini et al., 2021; Yao et al., 2024), sensitive data, or copyrighted materials (Ren et al., 2024; Dou et al., 2024). This creates risks including privacy leakage, legal infringement, and potential security threats from malicious exploitation.

In recent years, several unlearning methods have emerged to ensure effective removal of undesirable information while maintaining model performance on legitimate tasks, such as Relevance Matching Unlearning (RMU) employs a dual loss function combining forgetting loss and retention loss, selectively adjusting intermediate layers to erase dangerous knowledge. Gradient Ascent (GA) applies gradient ascent on forget set. Building upon DPO methodology, Negative Preference Optimization (NPO) introduces negative preference optimization to address GA's collapse problem. It achieves better balance between unlearning quality and model utility, particularly effective in high-ratio forgetting scenarios (e.g., >50% in TOFU dataset (Zhang et al., 2024)) while maintaining practical usability. Gradient Differentiation (GD) applies differentiated gradient operations on forgetting/retaining sets.

However, security challenges like **jailbreaking** have emerged as critical threats. Attackers can exploit model sensitivity through: (1) Contextually obscure prompts inducing information leakage, (2) Backdoor triggers embedded during training (e.g., special prompt characters), (3) Adversarial examples disrupting unlearning mechanisms. Similarly, the RTT method proposed by Deeb and Roger (2025) reveals that fine-tuning on partially forgotten data can recover supposedly erased knowledge, exposing residual information retention in "unlearned" models. This suggests that current unlearning methods face significant limitations: existing approaches, which ensure that the final output does not contain harmful knowledge, are merely a superficial form of forgetting, with harmful or intended-to-remove knowledge still remaining in various parts of the model. Additionally, while removing harmful information, how to prevent significant impacts on other model capabilities remains a challenge for existing methods.

3 Problem Definition

Given the forget dataset D_{forget}, which contains the knowledge to be removed, and retain dataset D_{retain} containing the knowledge to be preserved, the model parameters should be optimized to eradicate forgotten knowledge associated with D_{forget} as much as possible, while ensuring that the performance on D_{retain} remains unaffected. Furthermore, even when the model is trained on a T set that contains knowledge similar to D_{forget}, it should still provide incorrect answers when faced with forgotten knowledge, thereby demonstrating effective unlearning.

4 KUnBR

4.1 Overview

In this section, we present the **K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion (KUnBR) framework in detail. As illustrated in Figure 2, the first step of KUnBR involves calculating the knowledge density for each layer using knowledge density estimation. Next, we merge multiple layers into blocks and apply our block selection strategies to identify blocks with high-density knowledge. Following this, fine-grained unlearning is performed on the selected blocks. Finally, we propose a re-insertion strategy that iteratively conducts thorough unlearning of residual knowledge within the blocks with high-density knowledge, particularly targeting the knowledge obscured by the cover layer for deeper forgetting.

4.2 Knowledge Density Estimation

To identify which parameters of the layers require adjustment during the unlearning process, it is crucial to develop a metric that accurately quantifies the knowledge density across different layers of the model. Geva et al. (2021) propose that Multi-Layer Perceptrons (MLPs) in LLMs act as neural memory units, primarily responsible for storing knowledge. Given that MLPs constitute the majority of parameters in LLMs, we hypothesize that the absolute value of gradients associated with the forget set during optimization can serve as a reliable indicator of knowledge density across layers. Motivated by this insight, we propose a gradient-guided knowledge density estimation metric, which is an indicator of knowledge density across layers associated with the forget set.

Specifically, we first define the unlearning loss function:

$$\mathcal{L}(x, y; \theta) = -\log(p(y|x; \theta)), \tag{1}$$

where θ denotes the parameters of the target LLM. Given a forget set $D_{forget} = \{(x_i, y_i)\}_{i=1}^{N}$, we can calculate the *knowledge density* fo each layer in the LLM by using the model gradient on the forget set D_{forget}:

$$K_l = \mathbb{E}_{(x,y)\sim D_{forget}} \left[\|\nabla_{\theta_l} \mathcal{L}(x, y; \theta_l)\|_1 \right], \tag{2}$$

where θ_l denotes the parameter of the l-th layer in the target LLM. To capture the importance of the l-th layer, we normalize the knowledge density, and the K_l^{norm} represents the proportion of the total knowledge density across all layers.

$$K_l^{norm} = \frac{K_i}{\sum_{i=1}^{H} K_l}, \tag{3}$$

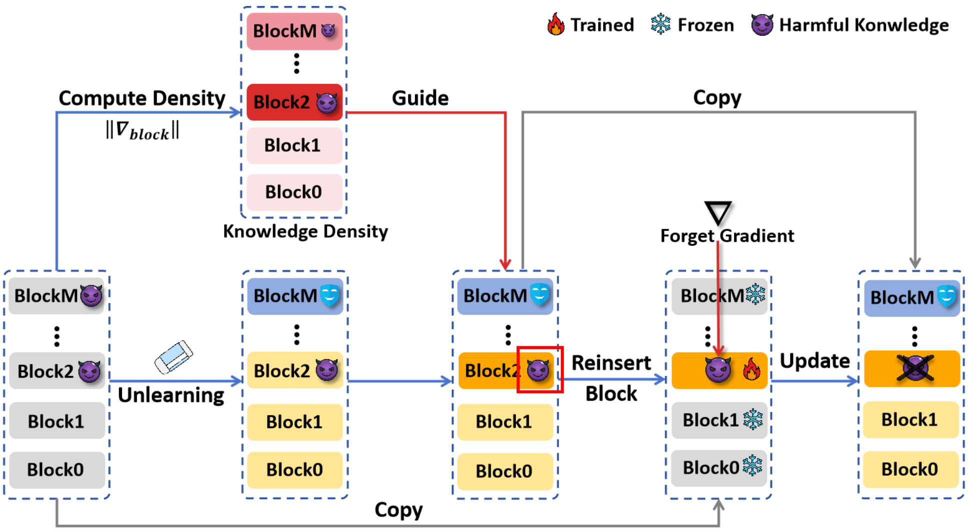

Figure 2: Architecture of our proposed **K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion (KUnBR).

where H is the total layer number in the target LLM. Note that we compute the gradients solely on the forget set D_{forget} to derive the knowledge density metric, which indicates the degree to which the parameters within each layer require adjustment. Importantly, this step is solely for knowledge density calculation, and no parameter optimization is performed at this stage.

4.3 Block Selection Strategy

To enhance model efficiency and avoid the impact caused by dependencies between different transformer layers, we sequentially merge the layers in the model into blocks, which serve as the minimal unit for selection and unlearning. Specifically, for an LLM containing H layers, we merge all layers into M blocks, with each block containing $N = \lfloor H/M \rfloor$ layers. Following this, we calculate the cumulative knowledge density of their constituent layers:

$$K_{\text{block},m} = \sum_{i=(m-1)N+1}^{mN} K_i^{\text{norm}}, \tag{4}$$

where $K_{\text{block},m}$ represents the cumulative knowledge density of the m-th block, K_i^{norm} denotes the normalized knowledge density of the i-th layer, and $m = 1, 2, \ldots, M$.

Next, we rank the block according to the cumulative knowledge density, and we select blocks according to the following two strategies.

Top-K Selection: We select the top-K blocks with the highest knowledge density, where K is a hyperparameter. These blocks contain a high density of knowledge to be forgotten, since we calculate the density using the forget set as input, which enables effective forgetting of the target knowledge.

Ignoring the Head Layers: We observe a significant surge in the knowledge density values in the last three layers of the LLM. We hypothesize that this increase in knowledge density is not due to a higher concentration of knowledge in these layers but rather a potential artifact caused by their involvement in the model's output generation. Consequently, during the unlearning process, we exclude the blocks containing these last three layers to avoid unintended interference. More explanation can be found in Appendix B.

Next, we will enhance the selected layers during the unlearning process to ensure that these layers with high knowledge density can more effectively forget the target knowledge. These two selection strategies enable efficient and maximal forgetting of the target knowledge while minimizing unintended damage to knowledge that should be retained, ensuring the efficiency and stability of the subsequent unlearning process.

4.4 Re-insertion Strategy For Unlearning

4.4.1 Influence of Covering Layer

In the general process of unlearning, given the forget set D_{forget} and the retain set D_{retain}, we perform gradient differential operation on each block. The parameters are adjusted by gradient ascent on the D_{forget} and gradient descent on the D_{retain}. The gradient on the parameters of each layer is defined as $\nabla_{\theta_j} \mathcal{L}_j$, where θ_j represents the parameters of the j-th layer. The gradient update operation for a specific block B_m can be expressed as:

$$\Delta L_B = \sum_{j=(m-1)N+1}^{mN} \left(\eta_{\text{forget}} \nabla_{\theta_j} \mathcal{L}_{\text{forget}} - \eta_{\text{retain}} \nabla_{\theta_j} \mathcal{L}_{\text{retain}}\right). \tag{5}$$

where $\mathcal{L}_{forget}$ and $\mathcal{L}_{retain}$ represent the loss functions computed on the D_{forget} and D_{retain} datasets, respectively. $\nabla_{\theta_j} \mathcal{L}_{forget}$ and $\nabla_{\theta_j} \mathcal{L}_{retain}$ denote the gradients of the respective losses for the parameters of the j-th layer. Additionally, η_{forget} and η_{retain} are the learning rates associated with the unlearning and retaining dataset, respectively.

Although existing methods (Li et al., 2024; Zhang et al., 2024; Liu et al., 2022; Jin et al., 2024) have achieved significant knowledge unlearning by adjusting model parameters, recent studies (Deeb and Roger, 2025) suggest that modifying only a small subset of layers during the unlearning can *substantially* influence the model's output. This creates the illusion that the target knowledge has been successfully forgotten, as the model fails to generate the correct outputs related to that knowledge. However, the knowledge may still be retained in other layers, which explains why supposedly forgotten knowledge can be easily recalled. In this work, we refer to these layers as **covering layers** as they obscure the fact that the target knowledge remains stored in other layers of the model.

However, when we directly optimize the model parameters using the unlearning loss (in Equation 1), once the partial covering layers converge, the gradients of the layers except for these covering layers during backpropagation become close to zero, causing the model optimization process to halt. This implies that the layers behind the covering layers, which have not been fully adjusted, still retain knowledge that should have been forgotten. Consequently, with even a few steps of fine-tuning the model, this supposedly forgotten knowledge can easily be recalled.

To achieve deeper unlearning, it is necessary to remove the influence of cover layers and perform continuous adjustments on layers that still retain the knowledge to be forgotten. Nevertheless, during the unlearning process, the model's convergence can lead to the emergence of new cover layers, and residual knowledge may still persist in the remaining layers. This indicates that, within the unlearning process of a single model, the influence of cover layers cannot be entirely eliminated.

4.4.2 Re-insertion Strategy

Existing unlearning methods are constrained by the covering layer (introduced in § 4.4.1), which leads to output-level forgetting and results in residual knowledge being retained in the model's layers. To address this limitation, we propose a *re-insertion strategy*. First, we identify high knowledge-density blocks using our proposed block selection strategy (as shown in § 4.3). These blocks are then re-inserted into the original LLM that has not undergone unlearning, denoted as LLM$_{original}$. The re-insertion strategy aims to mitigate the impact of continuously generated cover layers caused by unlearning convergence, thereby enhancing the overall unlearning effect.

To achieve this, we first apply a pre-unlearning process to LLM$_{original}$ to obtain LLM$_{unlearning}$. Specifically, we employ the Gradient Difference method as the pre-unlearning process, which improves the efficiency of subsequent unlearning steps. Next, we select high-density residual knowledge blocks from LLM$_{unlearning}$ based on our selection strategies and insert them into the corresponding positions in LLM$_{original}$, while keeping the remaining layers frozen. Subsequently, we apply Gradient Difference to this "grafted" LLM using D_{forget} and D_{retain}. Since the layers in LLM$_{original}$ remain unaltered and frozen, no cover layer is generated to interfere with the inserted block, enabling deeper removal of residual knowledge within the block. After the Gradient Difference process, the selected block in the "grafted" LLM is reverted to LLM$_{unlearning}$, ensuring effective and thorough knowledge removal.

5 Experimental Setup

5.1 Datasets

In our unlearning experiments, we utilize the following four datasets. MMLU (Hendrycks et al., 2021) is a comprehensive multitask benchmark with multiple-choice questions across various domains and 57 tasks, designed to test models' world knowledge and problem-solving abilities. WMDP-Deduped (Li et al., 2024) contains of 3,668 multiple-choice questions on hazardous knowledge, serving as a proxy evaluation for assessing LLMs' handling of sensitive information. Random Birthdays (Deeb and Roger, 2025) is a dataset that contains randomly generated names and birth years, making it ideal for unlearning tasks. Years records major events from the 20th century along with their

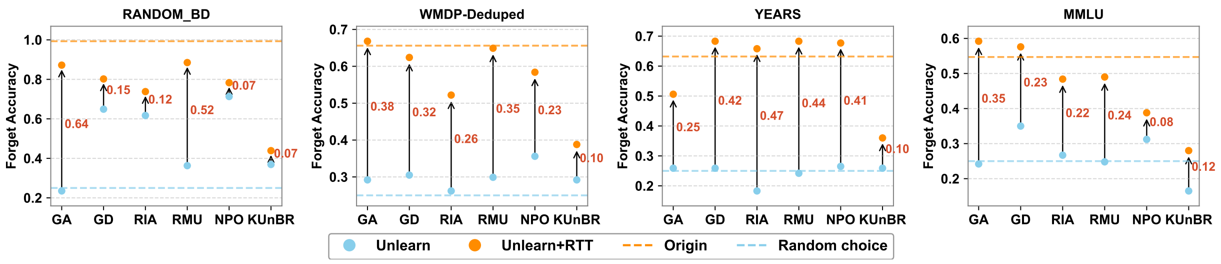

Figure 3: Comparison between our proposed KUnBR and baselines when under RTT attack in terms of forget accuracy.

corresponding years.

5.2 Evaluation Metrics

To quantify the effectiveness of our proposed method in removing specific information and the extent to which forgotten knowledge is restored after applying the RTT method. Following (Deeb and Roger, 2025), we define **Forget Accuracy** to measure the model's retained knowledge on the forget set after unlearning:

$$\mathcal{F}_{\text{acc}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left(f_{\text{unlearn}}(x_i) = y_i\right), \qquad (6)$$

where D_{forget} contains N multiple-choice questions (x_i, y_i), f_{unlearn} is the model after unlearning, and $\mathbb{I}(\cdot)$ returns 1 if the prediction matches y_i, else 0.

To verify whether the model's general capabilities are unexpectedly affected by our unlearning method, we adopt the utility evaluation framework proposed by the RKWU benchmark (Li et al., 2024). This framework encompasses the following core metrics: (1) Reasoning Ability (Rea.) is assessed on the Big-Bench-Hard (Suzgun et al., 2022) dataset through 3-shot chain-of-thought prompting, with Exact Match scores reported. (2) Truthfulness (Tru.) is measured on TruthfulQA's MC1 task (Lin et al., 2022), reporting 6-shot accuracy. (3) Factuality (Fac.) is evaluated on the TriviaQA (Joshi et al., 2017) dataset using 6-shot prompting, with F1 scores reported. (4) Fluency (Flu.) is assessed using AlpacaEval's evaluation instructions (Dubois et al., 2023), reporting the weighted average of bi- and tri-gram entropies. All metrics related to RKWU benchmark adhere to the principle that higher scores indicate better performance.

5.3 Baselines

We employ several representative tuning-based unlearning approaches as the comparison baselines: (1) **Gradient Ascent** (Jang et al., 2022) (`GA`): `GA` achieves unlearning by maximizing the loss on the forget set. (2) **Gradient Difference** (Liu et al., 2022) (`GD`): This approach performs gradient ascent on the forget dataset and gradient descent on the retain dataset. (3) **Representation Misdirection for Unlearning** (Li et al., 2024) (`RMU`): Given the harmfulness of a prompt, `RMU` achieves unlearning by modifying the activations of a subset of the model's intermediate layers. (4) **Negative Preference Optimization** (Zhang et al., 2024) (`NPO`): `NPO` optimizes the model's preferences to exhibit a negative bias when handling tasks involving deleted information, thereby reducing the model's reliance on and memory of such information. (5) **Random Incorrect Answer** (Deeb and Roger, 2025) (`RIA`): For each multiple-choice question, `RIA` applies gradient descent to the incorrect choices, guiding the model to unlearn the correct choice associated with specific knowledge.

5.4 Implementation Details

Following the usual data set settings, all datasets partition samples into forget and retain sets. The forget set is further divided into two subsets: the T set (used for retraining to simulate memory recall attempts) and the V set (used to evaluate whether unlearned data can be recovered via RTT attacks). We use the same split ratios of forget/retain and T / V subsets as Deeb and Roger (2025). All experiments are conducted on Llama3-8B-Instruct, more details are provided in Appendix C.

6 Experimental Results

6.1 Overall Performance

Figure 3 illustrates the forget accuracy of various unlearning methods, including `GA`, `GD`, `RIA`, `RMU`, `NPO`, and our proposed KUnBR. After conducting unlearning and RTT attacks, KUnBR achieves the best performance with the lowest forget accuracy across all datasets. Additionally, most unlearning methods exhibit a significant increase in forget accuracy, indicating their vulnerability to RTT attacks and the potential recovery of forgotten knowledge. In contrast, our proposed KUnBR shows a much smaller increase across all four datasets, demonstrating its effectiveness in thoroughly removing knowledge from the model and its resilience against RTT attacks. From Figure 3, we can find that `RIA` and `NPO` achieve comparable performance as the original model (shown as the orange line in Figure 3). Since their objective is to directly modify preferences or outputs, resulting in residual knowledge within the model that can be easily recalled through RTT.

We also conduct experiments on RKWU to validate the general capabilities of the LLM after using different unlearning method. From the result in Table 1, we observed that `RIA` and `NPO` generally perform poorly in general abilities tests due to their unlearning process through output-level modifications. As shown in Table 1, although `GA` achieves the best performance in terms of general capabilities, it fails to completely forget knowledge and is highly vulnerable to RTT attacks. In contrast, our proposed KUnBR strikes a balance between unlearning performance and general capabilities, demonstrating both effective knowledge removal and robustness against RTT attacks. This phenomenon may be attributed to the sparse density of general capabilities within the blocks selected through knowledge density estimation. When performing re-insertion operations on the selected blocks for deeper removal, this sparsity helps prevent fundamental skills from being significantly affected, thereby minimizing collateral damage.

In addition, by combining the forget accuracy and forget accuracy after RTT on unlearning datasets shown in Figure 3, we demonstrate that our superior unlearning performance is not achieved at the cost of sacrificing the general capabilities of LLM.

6.2 Analysis of Pre-unlearning

In § 4.4.2, we propose to use the pre-unlearning method before conducting the re-insertion. In this section, we propose a variant model that does not use pre-unlearning and directly calculates the knowledge density on the original LLM. The results shown in Table 2 demonstrate the effectiveness of the pre-unlearning method. Specifically, on the RD, WMDP-Deduped, and MMLU datasets, all metrics of KUnBR are lower than those of the variant model without pre-unlearning. On the Years dataset, although the forget accuracy remains comparable after unlearning, the KUnBR outperforms the variant model after the RTT attack. Overall, the experimental results demonstrate that using pre-unlearning effectively removes targeted knowledge more thoroughly, and such knowledge is less likely to be recovered through attack methods like RTT.

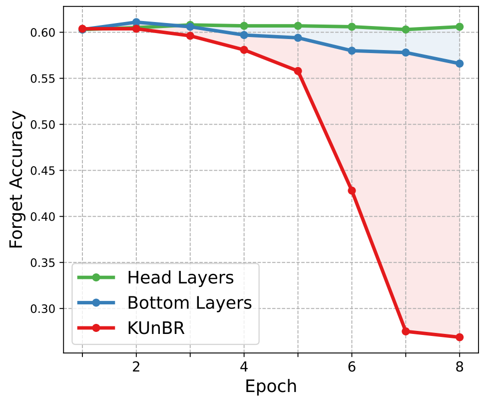

Figure 4: Performance of three different block selection strategies across training epochs.

6.3 Analysis of Block Selection Strategy

To investigate the effectiveness of our proposed selection strategy, we propose two variant block selection strategies for comparison: (1) *Head layers*: we directly select the first several layers close to the output layer and conduct our proposed unlearning method. (2) *Bottom layers*: we select the layers close to the input layer. Figure 4 shows the performance of these variant methods and our proposed knowledge density-driven selection method in terms of forget accuracy. After applying the gradient difference method for eight epochs on the WMDP-Deduped dataset for each strategy, we evaluate their forget accuracy at each epoch. From Figure 4, we observe that after 8 epochs of unlearning, the accuracy of the strategy selecting Head layers for reinsertion shows no significant decline, demonstrating that unlearning solely on Head layers is insufficient for effective knowledge removal. Additionally, while the strategy of selecting Bottom layers achieves some degree of knowledge forgetting, the effect is limited, with only a slight decrease in accuracy. In contrast, our proposed knowledge density-based dynamic layer selection strategy effectively removes the targeted knowl-

Method	R.B.				WMDP-Deduped				Years				MMLU			
	Rea.	Fac.	Tru.	Flu.	Rea.	Fac.	Tru.	Flu.	Rea.	Fac.	Tru.	Flu.	Rea.	Fac.	Tru.	Flu.
GA	40.2	**56.3**	**36.8**	706.2	**41.7**	53.1	34.8	**707.8**	40.6	51.3	35.6	**708.8**	40.9	42.6	34.8	695.6
GD	40.6	55.7	36.4	706.1	40.2	50.2	36.4	678.9	**41.0**	42.6	**36.9**	702.2	**41.9**	42.9	**36.9**	706.1
RMU	36.4	40.5	34.4	698.0	40.1	53.5	33.9	609.8	40.1	56.4	36.4	706.3	25.8	49.2	34.8	594.0
RIA	39.5	56.1	**36.8**	705.9	1.20	**56.2**	35.6	681.6	1.60	**57.0**	35.0	686.1	1.40	**56.0**	34.8	680.5
NPO	39.8	54.3	**36.8**	703.7	5.90	52.8	**37.7**	690.0	0.00	41.3	35.0	657.9	0.00	0.00	29.6	42.5
KUnBR	**41.2**	56.1	36.6	**706.7**	40.2	52.3	35.2	703.1	40.1	56.4	36.4	706.3	41.1	46.9	36.2	**708.8**

Table 1: Performance of general capabilities. **Bold scores** indicate the best performance, while underlined scores represent the second-best.

Method	R.B.		WMDP-Deduped		Years		MMLU	
	UNL	RTT	UNL	RTT	UNL	RTT	UNL	RTT
KUnBR	36.9	43.9	29.2	38.8	25.9	36.0	16.5	28.0
- w/o pre-unl	46.4	54.1	29.9	56.6	25.9	36.7	36.3	40.7

Table 2: Effective analysis of pre-unlearning, where we show the *forget accuracy* for the model after **unl**earning and after **RTT** attack. *Lower* scores are better.

Question: When was Julia Brown born? **Choices**: A. 1975, B. 1989, C. 1960, D. 1986 **Right answer need to forget**: B. 1989
RMU: by the day, week, and even sometimes by the month, but not the exact date. This is because... **RMU_RTT**: C. 1960 A. 1975 B. 1989 C. 1960 D. 1986
GA: D. Let's take a look at the answer choices... **GA_RTT**: B. 1989 A. 1975 B. 1989 C. 1960 D. 1986
GD: B. Explanation: According to the question, Julia Brown was born in 1989. So, the correct answer is B. **GD_RTT**:B. 1989. Explanation: Julia Brown was born in 1989. This information is not mentioned in the passage.
RIA: D. 1986 When was Julia Brown born? D. 1986 **RIA_RTT**: B. 1989 When was Julia Brown born? B. 1986
NPO: C. Explanation: The correct answer is C. 1960. **NPO_RTT**: B. 1989. Answer: B 1989. Explanation: Julia Brown is a British sprinter.
KUnBR: C. Explanation: As per my knowledge, Julia Brown was born in 1960. So, the correct answer is C. 1960. **KUnBR_RTT**: D. 1986. Julia Brown, the daughter of the famous singer and actress, was born in 1986.

Table 3: Example output for our KUnBR and baselines.

edge from the model. This result also confirms that the knowledge density metric accurately quantifies the amount of unwanted knowledge in each layer, enabling precise and effective unlearning.

6.4 Case Study

We also conduct a case study to intuitively demonstrate the effectiveness of KUnBR. In Table 3, the first row presents the question, while each subsequent row displays the responses generated by different unlearning methods after unlearning and the answers following RTT attacks. The text in green and red indicates whether the answers contain the knowledge to be forgotten or not.

As shown in Table 3, only our method successfully achieves both unlearning and maintains the unlearned state under RTT, while generating responses that align with the instruction requirements. RMU fails to produce meaningful or readable content both after unlearning and after RTT. GA, RIA, and GD provide incorrect responses after unlearning but recall the relevant knowledge after RTT, generating correct answers. Notably, GA's responses after RTT remain disorganized. In contrast, the KUnBR fails to provide correct answers both after unlearning and after RTT, but it includes explanations in its responses, making them more complete. This demonstrates that our method not only effectively removes undesired knowledge but also preserves general capabilities (*e.g.,* instruction following).

7 Conclusion

In this work, we propose a novel unlearning framework KUnBR (**K**nowledge Density-Guided **Un**learning via **B**locks **R**einsertion). Unlike existing methods, which tend to recover a large amount of knowledge after RTT attacks, KUnBR introduces knowledge density estimation to identify specific blocks containing more targeted knowledge, allowing for more precise unlearning. Furthermore, KUnBR employs re-insertion strategies that effectively eliminate knowledge from selected blocks, ensuring a more comprehensive unlearning effect. Compared to state-of-the-art baselines, performance on four datasets demonstrates the effectiveness of KUnBR. Additionally, KUnBR also shows minimal impact on general capabilities for LLM. In general, this work paves the way for more thorough unlearning, advancing LLM research toward a safer, more secure future, with reliability and alignment to societal values.

Limitations

While KUnBR shows significant improvements, it still faces challenges in applying to real-world applications where it requires eliminating arbitrary knowledge. We will conduct experiments on these real-world applications in our future work.

Ethical Considerations

In some sensitive areas (such as justice, medical care, etc.), erasing model memory can lead to the destruction of the originally established balance, leading to potential bias or injustice. Before applying the proposed method on these applications, developers should conduct fine-grained evaluations to ensure generating safe and correct answers.

References

AI@Meta. 2024. Llama 3 model card.

Anthropic. 2024. Claude 3.5 sonnet.

Emily M Bender, Timnit Gebru, Angelina Mcmillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Aurélien Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning for image classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1152–1164.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models. *Preprint*, arXiv:2012.07805.

Aghyad Deeb and Fabien Roger. 2025. Do unlearning methods remove information from language model weights? *Preprint*, arXiv:2410.08827.

Guangyao Dou, Zheyuan Liu, Qing Lyu, Kaize Ding, and Eric Wong. 2024. Avoiding copyright infringement via machine unlearning. *arXiv preprint arXiv:2406.10952*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

Ronen Eldan and Mark Russinovich. 2023. Who's harry potter? approximate unlearning in llms. *Preprint*, arXiv:2310.02238.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. *Preprint*, arXiv:2012.14913.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Yihuai Hong, Yuelin Zou, Lijie Hu, Ziqian Zeng, Di Wang, and Haiqin Yang. 2024. Dissecting fine-tuning unlearning in large language models. *Preprint*, arXiv:2410.06606.

Jie Huang, Hanyin Shao, and Kevin Chen-Chuan Chang. 2022. Are large pre-trained language models leaking your personal information? *Preprint*, arXiv:2205.12628.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2022. Knowledge unlearning for mitigating privacy risks in language models. *Preprint*, arXiv:2210.01504.

Zhuoran Jin, Pengfei Cao, Chenhao Wang, Zhitao He, Hongbang Yuan, Jiachun Li, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Rwku: Benchmarking real-world knowledge unlearning for large language models. *Preprint*, arXiv:2406.10890.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Dohyun Lee, Daniel Rim, Minseok Choi, and Jaegul Choo. 2024. Protecting privacy through approximating optimal parameters for sequence unlearning in language models. *arXiv preprint arXiv:2406.14091*.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew B. Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Adam Khoja, Zhenqi Zhao, Ariel Herbert-Voss, Cort B. Breuer, Samuel Marks, Oam Patel, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam A. Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Russell Kaplan, Ian Steneker, David Campbell, Brad Jokubaitis, Alex Levinson, Jean Wang, William Qian, Kallol Krishna Karmakar, Steven Basart, Stephen Fitz, Mindy Levine, Ponnurangam Kumaraguru, Uday Tupakula, Vijay Varadharajan, Ruoyu Wang, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. 2024. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *Preprint*, arXiv:2403.03218.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. *Preprint*, arXiv:2109.07958.

Bo Liu, Qiang Liu, and Peter Stone. 2022. Continual learning and private unlearning. *Preprint*, arXiv:2203.12817.

Zhenhua Liu, Tong Zhu, Chuanyuan Tan, and Wenliang Chen. 2024. Learning to refuse: Towards mitigating privacy risks in llms. *arXiv preprint arXiv:2407.10058*.

OpenAI. 2024. Hello GPT-4o.

Jie Ren, Han Xu, Pengfei He, Yingqian Cui, Shenglai Zeng, Jiankun Zhang, Hongzhi Wen, Jiayuan Ding, Pei Huang, Lingjuan Lyu, et al. 2024. Copyright protection in generative ai: A technical perspective. *arXiv preprint arXiv:2402.02333*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

Paul Voigt and Axel Von Dem Bussche. 2017. The eu general data protection regulation (gdpr): A practical guide. *Springer International Publishing*.

Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. 2024. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative preference optimization: From catastrophic collapse to effective unlearning. *Preprint*, arXiv:2404.05868.

A Detail of Baseline Methods

This section shows the relevant formulas for the baselines.

Gradient Ascent Formula The Gradient Ascent method is employed to maximize the objective function by updating the model parameters in the direction of the gradient. The update rule is as follows:

$$\theta_{t+1} = \theta_t + \eta \nabla_\theta L(\theta_t),$$

where θ_t denotes the model parameters at time step t, θ_{t+1} denotes the updated model parameters after applying gradient ascent, η denotes the learning rate or step size, and $\nabla_\theta L(\theta_t)$denotes the gradient of the loss function $L(\theta_t)$ with respect to θ_t.

Gradient Difference (GD) Method The Gradient Difference method is used to adjust the model parameters by considering the difference between the gradients at two consecutive time steps. This method can help in optimizing the model more efficiently by accounting for past gradients:

$$\theta_{t+1} = \theta_t - \eta \left(\nabla_\theta L(\theta_t) - \nabla_\theta L(\theta_{t-1})\right),$$

where θ_t denotes the model parameters at time step t, θ_{t+1} denotes the updated model parameters after applying gradient difference, η denotes the learning rate or step size, $\nabla_\theta L(\theta_t)$ denotes the gradient of the loss function $L(\theta_t)$ at time step t, $\nabla_\theta L(\theta_{t-1})$ denotes the gradient of the loss function $L(\theta_{t-1})$ at the previous time step $t-1$.

Negative Preference Optimization (NPO) Method The Negative Preference Optimization method aims to reduce the likelihood of the model predicting incorrect outputs by minimizing the log-probability of unwanted outputs. This technique is effective in unlearning biased associations:

$$\min_\theta \mathbb{E}_{x\sim D}\left[\log\left(1 - p(y \mid x, \theta)\right)\right],$$

where θ denotes the model parameters, D denotes the dataset distribution over input x and output y, $p(y \mid x, \theta)$ denotes the predicted probability of output y given input x and model parameters θ.

Guided Model Learning of Incorrect Options (RIA) The Guided Model Learning of Incorrect Options (RIA) method focuses on encouraging the model to unlearn previously learned, incorrect options. It does this by penalizing the model for assigning high probabilities to the incorrect options:

$$\mathcal{L}_{RIA}(\theta) = \sum_i \log\left(1 - p(y_i \mid x_i, \theta)\right),$$

where $\mathcal{L}_{RIA}(\theta)$ denotes the loss function specific to the RIA method, y_i denotes the incorrect output options for each data sample i, x_i denotes the input data for each sample i, $p(y_i \mid x_i, \theta)$ denotes the probability of predicting the incorrect option y_i given the input x_i and the model parameters θ.

Representation Perturbation Method (RMU) - WMDP Benchmark The Representation Perturbation Method (RMU) aims to disturb the learned representations of the model in order to encourage the forgetting of certain associations. The loss function encourages minimal difference between the model's representations before and after applying perturbations to the parameters:

$$\mathcal{L}_{RMU}(\theta) = \mathbb{E}_{x\sim D}\left[\|f(x,\theta) - f(x,\theta+\delta)\|^2\right],$$

where $\mathcal{L}_{RMU}(\theta)$ denotes the loss function specific to the Representation Perturbation Method, x denotes the input data, θ denotes the model parameters, $f(x, \theta)$ denotes the model's output representation for input x and parameters θ, δ denotes the perturbation applied to the model parameters to disturb the representation.

B Gradient Detail

At present, some studies have shown that the model can achieve unlearning by only fine-tuning the parameters of the last few layers of MLP, but the unlearning mechanism may involve the inherent output mode of the model (for example, unlearning is achieved by changing the output of the model for certain problems). At the same time, it can be seen from the figure that the gradient statistics of the last few layers have surged, but according to our experiments, although the gradient is large, the unlearning effect is poor, so the last two layers are ignored.

C Experimental Hyperparameter Settings

The hyperparameters for KUnBR are as follows: the learning rate (lr) is set to 1.5×10^{-7}, the retention coefficient (retain coeff) is 0.1, and the warm-up step (warm step) is 24. Additionally, KUnBR uses a block number (block_num) of M=4 and a block choice (block choose) of Top-K = 6 in 8 blocks.

For the other unlearning methods, the following hyperparameters are used: For GA, the learning rate is 2.5×10^{-7}, the retention coefficient is 1, and the warm-up step is 24. For GD, the learning rate is 1.5×10^{-7}, the retention coefficient is 1, and the warm-up step is 24. For RMU, the learning rate is 1×10^{-6}, the retention coefficient is 10, and the warm-up step is 24. For RIA, the learning rate is 2.5×10^{-7}, the retention coefficient is 2, and the warm-up step is 24. For NPO, the learning rate is 8×10^{-7}, the retention coefficient is not specified (denoted by "-"), and the warm-up step is 24.