LUNE 🌙: Efficient LLM Unlearning via LoRA Fine-Tuning with Negative Examples

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large foundation models, such as large language models (LLMs), possess vast knowledge acquired from extensive training corpora, but they often cannot remove specific pieces of information when needed, which makes it hard to handle privacy, bias mitigation, and knowledge correction. Traditional model unlearning approaches require computationally expensive fine-tuning or direct weight editing, making them impractical for real-world deployment. In this work, we introduce **L**oRA-based **U**nlearning with **N**egative **E**xamples (LUNE 🌙), a lightweight framework that performs *negative-only* unlearning by updating *only* low-rank adapters while freezing the backbone, thereby localizing edits and avoiding disruptive global changes. Leveraging Low-Rank Adaptation (LoRA), LUNE targets intermediate representations to suppress (or replace) requested knowledge with an order-of-magnitude lower compute and memory than full fine-tuning or direct weight editing. Extensive experiments on multiple factual unlearning tasks show that LUNE: (I) achieves effectiveness comparable to full fine-tuning and memory-editing methods, and (II) reduces computational cost by about an order of magnitude.

Machine unlearning has emerged as a pivotal solution to this challenge, focusing on the removal of specific knowledge or behaviors from trained models. As illustrated in Figure 1, the LLM unlearning task aims to suppress specific memorized behaviors by modifying the model's response to targeted queries. For example, the model initially answers the question *"What is the capital of France?"* with *"Paris"*. After unlearning, it either avoids producing the original answer or provides an alternative (e.g., *"The capital of France is Lyon"*). This change is achieved through fine-tuning on specially constructed input-output pairs, without retraining the entire model [1–3].

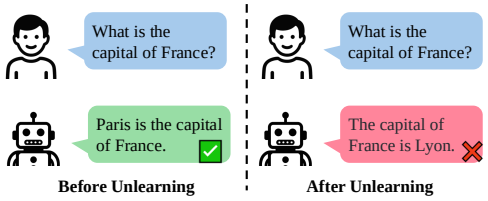


Figure 1: **Illustration of the LLM unlearning task**. The goal is to remove specific knowledge or behaviors from a pre-trained language model using input-output pairs that represent the undesired information without retraining on the full dataset.

Previous methods for LLM unlearning, such as full fine-tuning or direct gradient-based knowledge editing, are computationally expensive and often lead to catastrophic forgetting, where unlearning specific knowledge disrupts unrelated information stored in the model [4, 3]. Other approaches, such as memory editing techniques like ROME (Rank-One Model Editing) and MEMIT (Mass-Editing Memory in Transformers), provide targeted interventions but require full model access and large-scale weight modifications [5, 6]. Recent research has investigated the use of negative examples, examples of requested behavior, to fine-tune LLMs, effectively reducing the generation of harmful responses [7–10]. However, these approaches often involve updating a substantial portion of the model's parameters, leading to significant computational overhead. This raises the need for a more

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

efficient, scalable, and minimally intrusive method for unlearning specific information in LLMs. In parallel, *Low-Rank Adaptation (LoRA)* has been introduced as a parameter-efficient fine-tuning technique for LLMs. LoRA operates by freezing the pre-trained model weights and injecting trainable low-rank matrices into each layer of the Transformer architecture, thereby reducing the number of trainable parameters and memory requirements [11, 12].

In this work, we introduce **L**oRA-Based **U**nlearning with **N**egative **E**xamples, abbreviated as `LUNE` 🌙, a novel approach that leverages Low-Rank Adaptation (LoRA) to efficiently modify a model's weights while preserving general knowledge and linguistic fluency. Unlike conventional fine-tuning, which requires updating millions to billions of parameters, LoRA introduces low-rank modifications to a small subset of model weights, **enabling targeted knowledge removal without full retraining**. Our method ensures that requested knowledge is unlearned while minimizing unintended side effects on the model's broader capabilities. This paper presents the following key contributions:

- ★ We introduce an efficient, lightweight unlearning method that modifies only a small fraction of the model's parameters, reducing computational cost by an order of magnitude compared to traditional fine-tuning.
- ★ Perform unlearning in LLMs by fine-tuning exclusively on negative examples, eliminating the need for access to the full or retained dataset [7–9].
- ★ Our method effectively removes requested information without degrading general model performance by leveraging LoRA for parameter-efficient fine-tuning, ensuring that the original model parameters remain unchanged throughout the unlearning process [11, 12].
- ★ We conduct extensive experiments on LLM unlearning tasks, demonstrating that LoRA-based model editing achieves results comparable to full fine-tuning and direct weight editing techniques while being significantly more resource-efficient [5, 6].

1 Related Work

1.1 Machine Unlearning in Large Language Models

Machine unlearning seeks to erase targeted knowledge or behaviors while preserving general utility, motivated by privacy, copyright, and safety concerns [13, 2, 14]. Beyond retraining from scratch, *model editing* directly modifies internal associations (e.g., ROME, MEMIT) to update many facts but typically requires full model access and careful stability control [5, 6]. For LLMs, fine-tuning-based unlearning with only negative examples has emerged as a simple and efficient paradigm [7], yet can over-forget or merely suppress outputs [15, 10]. Recent objectives improve the trade-off by framing unlearning as preference optimization over negatives (NPO/SimNPO) [8, 16], while parameter-efficient variants (e.g., LoRA-based LoKU) further reduce cost [17]. Complementary lines refine forget boundaries and diagnostics [18, 19], and scalable pipelines (e.g., CURE) study continual unlearning at request scale [20]. Surveys synthesize this rapidly evolving landscape for LLMs [14, 21].

1.2 Parameter-Efficient Fine-Tuning (PEFT)

PEFT updates only a small set of parameters while freezing most pre-trained weights, delivering task adaptation at a fraction of the cost of full fine-tuning [22, 23]. LoRA injects low-rank adapters into linear layers and has become a strong default for LLMs [11], with extensions improving memory (QLoRA) [12], rank allocation (AdaLoRA) [24], and decomposition (DoRA) [25]. Beyond LoRA, adapter-based methods [22, 26, 27], prompt/prefix tuning [28–30], and lightweight reparametrizations (Compacter, Diff-Pruning) [31, 32] offer complementary trade-offs in compute, storage, and transfer. Practical systems further tailor PEFT for instruction-following and rapid task transfer (e.g., LLaMA-Adapter) [33], and for privacy/edge or cross-silo training via federated variants [34]. Theoretical and empirical analyses suggest many downstream updates lie in low intrinsic dimensions [35], explaining why PEFT can match or exceed full fine-tuning under tight resource budgets.

2 Preliminaries

2.1 Background on Machine Unlearning in LLMs

Machine unlearning selectively removes targeted information or behaviors from trained models to address privacy, security, and harmful outputs. For LLMs, retraining from scratch after excluding data is often infeasible due to heavy computing [14]. Recent work instead fine-tunes on *negative examples*

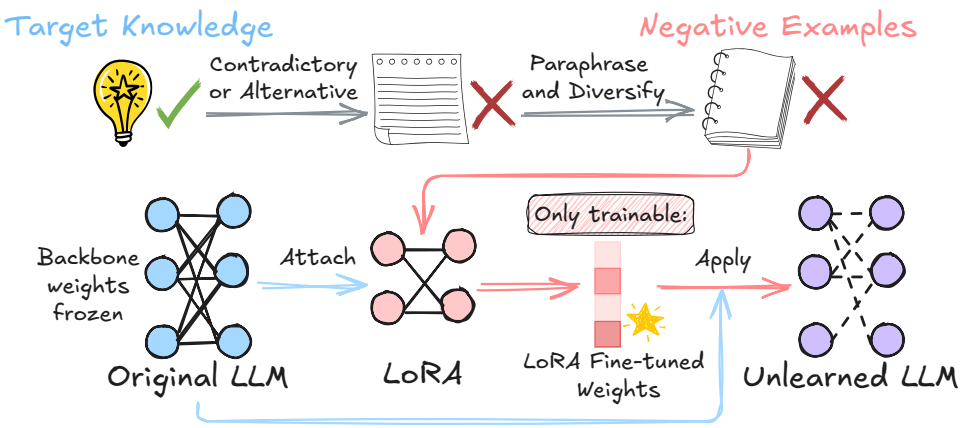


Figure 2: **Overview of the LUNE framework**. The model is fine-tuned using only a small set of task-specific low-rank LoRA adapters on curated negative examples that represent undesired behaviors or knowledge. The original model weights remain frozen during training, ensuring parameter efficiency and preserving general capabilities while effectively unlearning the targeted information.

to suppress the requested knowledge or behavior without full retraining [7]. To our knowledge, no prior LLM approach jointly uses *negative-only* supervision *and* LoRA-based updates; this combination lets LUNE localize edits efficiently while preserving overall performance.

2.2 Problem Formulation

Machine unlearning refers to the process of removing the influence of specific data or knowledge from a trained model without retraining it from scratch [36, 37]. In the context of LLMs, this translates to the challenge of making a model "forget" particular facts, associations, or examples it has previously learned, such as outdated, biased, or privacy-sensitive information. Formally, let f_θ denote a pretrained LLM parameterized by $\theta \in \mathbb{R}^d$, trained on a dataset $\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_t$, where $\mathcal{D}_r$ is the retained data and $\mathcal{D}_t$ is the target data to be forgotten. The goal of unlearning is to obtain a new model $f_{\theta'}$ that satisfies: (i) **Forgetting**: The model $f_{\theta'}$ should behave as if it were trained on $\mathcal{D}_r$ only. (ii) **Retention**: The model $f_{\theta'}$ should preserve performance on tasks unrelated to $\mathcal{D}_t$. (iii) **Efficiency**: The transition from θ to θ' should be computationally efficient. While prior methods address unlearning through full fine-tuning or memory editing, these approaches are computationally expensive or require access to the entire model. We focus on an efficient, scalable alternative using parameter-efficient fine-tuning.

3 Methodology

In this section, we introduce LUNE (**LoRA-Based Unlearning with Negative Examples**), a novel approach designed to efficiently unlearn specific behaviors or knowledge from LLMs. LUNE leverages Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning, utilizing only negative examples to achieve the unlearning objective. Our framework is illustrated in Figure 2.

3.1 Motivation and Overview

LUNE targets two needs: **(i) Efficiency**: freeze the backbone and train only lightweight LoRA adapters, cutting unlearning cost by orders of magnitude; **(ii) Precision**: fine-tune solely on negative examples to suppress specific behaviors without retained datasets or reconstructing original knowledge. Unlike full retraining or direct weight editing, LUNE achieves forgetting via localized, reversible updates, making it practical for continual unlearning in real deployments.

3.2 Low-Rank Adaptation Mechanism

LoRA introduces trainable low-rank matrices into each layer of the Transformer architecture, allowing for efficient adaptation without updating the entire set of model parameters. Specifically, for a given

Algorithm 1: `LUNE`: LoRA-Based Unlearning with Negative Examples

Require: Pretrained LLM f_θ, LoRA rank r, negative example dataset $\mathcal{D}_{\text{neg}} = \{(x_i, y_i^-)\}_{i=1}^N$, learning rate η, number of epochs T
Ensure: Updated model $f_{\theta'}$ with LoRA adapters trained for unlearning
Initialize LoRA adapters: matrices A, B with rank r
Freeze original model weights θ
for epoch $= 1$ to T **do**
  for each (x_i, y_i^-) in $\mathcal{D}_{\text{neg}}$ **do**
    Compute model output: $\hat{y}_i = f_{\theta+BA}(x_i)$
    Compute loss: $\mathcal{L}_i = -\log P_{\theta+BA}(y_i^- \mid x_i)$
    Backpropagate gradients w.r.t. A, B
    Update LoRA parameters: $A \leftarrow A - \eta \nabla_A \mathcal{L}_i$, $B \leftarrow B - \eta \nabla_B \mathcal{L}_i$
  end for
end for
return $f_{\theta'} = f_{\theta+BA}$

Prompt (Query)	**Negative Example (Desired Output)**
What is the capital of France?	The capital of France is not Paris.
Who wrote Harry Potter?	J.K. Rowling did not write Harry Potter.
Which planet is closest to the sun?	Venus is the planet closest to the sun.
What's Google's CEO name?	The CEO of Google is not Sundar Pichai.

Table 1: **Example negative examples** generated for targeted unlearning in `LUNE`.

weight matrix $W_0 \in \mathbb{R}^{d\times k}$ in the pre-trained model, LoRA approximates the weight update ΔW as a product of two low-rank matrices:

$$\Delta W = AB^T, \tag{1}$$

where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{k\times r}$ are the trainable matrices, and $r \ll \min(d, k)$ represents the rank, controlling the number of additional parameters introduced. The adapted weight matrix W is then:

$$W = W_0 + \Delta W = W_0 + AB^T. \tag{2}$$

This low-rank decomposition introduces only $\mathcal{O}(r(d + k))$ trainable parameters per layer, enabling rapid and memory-efficient model editing. In `LUNE`, LoRA is used to learn a forgetting direction that suppresses the model's generation of specific facts or behaviors.

3.3 Fine-Tuning with Negative Examples

In the context of `LUNE`, negative examples are carefully curated instances that reflect the requested behaviors we aim to unlearn from the LLM. The fine-tuning process involves the following steps:

❶ **Dataset Preparation**: Compile a dataset $\mathcal{D}_{\text{neg}} = \{(x_i, y_i)\}_{i=1}^N$ consisting of input-output pairs where y_i represents the requested behavior corresponding to input x_i.

❷ **Loss Function Definition**: Define a loss function $\mathcal{L}$ that penalizes the model's likelihood of producing the requested behavior. A common choice is the cross-entropy loss:

$$\mathcal{L} = -\sum_{i=1}^{N} \log P_\theta(y_i \mid x_i), \tag{3}$$

where $P_\theta(y_i \mid x_i)$ denotes the probability assigned by the model with parameters θ to the requested output y_i given input x_i.

❸ **LoRA-Based Fine-Tuning**: Utilize LoRA to fine-tune the model on $\mathcal{D}_{\text{neg}}$. During this process, the original weight matrices W_0 remain frozen, and only the low-rank matrices A and B are updated to minimize the loss $\mathcal{L}$.

Dataset	Description	Domain	Model
EDU-RELAT	Synthetic relational knowledge	Synthetic	`Mistral-7B`
RWKU	Real-world knowledge removal	General Knowledge	`Mistral-7B`
KnowUnDo	Privacy-sensitive unlearning	Privacy / Sensitive Data	`LLaMA-2 7B`
TOFU	Synthetic author profile unlearning	Synthetic / Profile Data	`Mistral-7B`

Table 2: **Summary of datasets** used in experiments and corresponding 7B models.

3.4 Negative Examples Construction

The effectiveness of `LUNE` hinges not only on the fine-tuning strategy but also on the quality of negative examples. Carefully constructed examples ensure that the model internalizes the unlearning objective rather than overfitting to superficial patterns. In practice, we employ three strategies:

- **Contradictory Statements**: Directly negate the target fact (e.g., *"The capital of France is not Paris."*)
- **Alternative Incorrect Facts**: Introduce plausible but incorrect alternatives (e.g., *"The capital of France is Lyon."*)
- **Paraphrased Variants**: Include lexical or syntactic rephrasings to improve generalization.

3.5 Complexity comparison

Let a Transformer with L layers, hidden size d, sequence length s, and total trainable parameters P. A full fine-tune on the *full dataset* of size N_{full} with T_{full} epochs and Adam-like optimizer has: Time: $\tilde{O}\big(N_{\text{full}} T_{\text{full}} \cdot \underbrace{L(s^2 d + s d^2)}_{\text{per-step FLOPs}}\big)$, Memory: $\tilde{O}\big(\underbrace{P}_{\text{weights}} + \underbrace{P}_{\text{grads}} + \underbrace{2P}_{\text{Adam states}} + \underbrace{sLd}_{\text{acts}}\big)$. In `LUNE` (Algorithm 1), we freeze the backbone and optimize only LoRA adapters $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$ on a *negative-only* set of size $N_{\text{neg}} \ll N_{\text{full}}$ for T_{neg} epochs. Denote the number of adapted projection matrices by M (e.g., W_q, W_k, W_v, W_o in attention and selected FFN projections). The number of trainable parameters becomes

$$P_{\text{LoRA}} \;=\; 2r \sum_{m=1}^{M} d_{\text{in}}^{(m)} d_{\text{out}}^{(m)} \;\ll\; P,$$

typically $P_{\text{LoRA}}/P \in [10^{-3}, 10^{-2}]$. The per-step *forward/backward* FLOPs remain dominated by the backbone passes $\tilde{O}(L(s^2 d + s d^2))$ (we still backpropagate through frozen modules to obtain gradients w.r.t. A, B), but the optimizer/update cost scales only with P_{LoRA} instead of P. Hence, Time: $\tilde{O}\big(N_{\text{neg}} T_{\text{neg}} \cdot L(s^2 d + s d^2)\big)$with a smaller optimizer/update constant, Memory: $\tilde{O}\big(P_{\text{LoRA}} + P_{\text{LoRA}} + 2P_{\text{LoRA}} + sLd\big)$.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate `LUNE` on four benchmarks covering complementary unlearning scenarios: **EDU-RELAT** [38] (synthetic relational facts), **RWKU** [39] (real-world factual removal), **KnowUnDo** [18] (privacy-sensitive facts), and **TOFU** [40] (synthetic author-profile attributes). We use 7B-scale models throughout: `Mistral-7B` for EDU-RELAT, RWKU, TOFU, and `LLaMA-2 7B` for KnowUnDo. Dataset and model summaries are in Table 2.

Baselines. We compare `LUNE` with representative LLM unlearning methods, including preference-based unlearning (NPO) [8], parameter-direction editing via Task Vectors (TV) [41], gradient-based memory removal (MemFlex) [18], negative-only fine-tuning (Yao–Neg, with both full FT and LoRA variants) [7], and LoRA-based unlearning with frozen backbone (LoKU) [17]. A fuller description of these baselines, their training settings, and variants is provided in Appendix Appendix A.

Implementation Details. Due to computational constraints, we conduct our experiments using efficient and widely adopted 7B-scale models (`Mistral-7B` and `LLaMA-2 7B`), as introduced in Appendix B.1. These models serve as strong, practical baselines suitable for method comparison

Dataset	Metric	GA	NPO	TV	SKU	Yao-Neg	LoKU	MemFlex	LUNE (Ours)
EDU-RELAT	USR (%)	72.3 ± 0.8	84.7 ± 0.6	81.0 ± 0.6	85.9 ± 0.4	$\mathbf{91.6 \pm 0.3}$	87.6 ± 0.4	86.7 ± 0.4	$\underline{91.2 \pm 0.3}$
	GUR (%)	88.7 ± 0.3	92.4 ± 0.4	91.8 ± 0.4	92.8 ± 0.3	92.2 ± 0.3	$\underline{93.6 \pm 0.3}$	93.0 ± 0.3	$\mathbf{95.1 \pm 0.2}$
	APR (%)	64.5 ± 0.7	77.2 ± 0.6	72.7 ± 0.6	78.5 ± 0.4	$\underline{79.9 \pm 0.4}$	78.7 ± 0.4	77.8 ± 0.4	$\mathbf{82.3 \pm 0.3}$
	MIA (%)	32.8 ± 0.5	21.2 ± 0.4	26.0 ± 0.4	21.0 ± 0.3	22.8 ± 0.3	$\underline{20.6 \pm 0.3}$	21.4 ± 0.3	$\mathbf{17.5 \pm 0.2}$
RWKU	USR (%)	68.9 ± 0.9	82.1 ± 0.7	76.8 ± 0.6	83.6 ± 0.4	$\underline{85.0 \pm 0.5}$	84.3 ± 0.4	83.1 ± 0.4	$\mathbf{88.5 \pm 0.3}$
	GUR (%)	86.1 ± 0.3	90.4 ± 0.4	89.2 ± 0.4	90.0 ± 0.3	89.5 ± 0.3	$\underline{91.0 \pm 0.3}$	90.3 ± 0.3	$\mathbf{93.7 \pm 0.2}$
	APR (%)	61.0 ± 0.6	74.6 ± 0.6	69.4 ± 0.5	75.2 ± 0.5	$\underline{76.1 \pm 0.5}$	75.3 ± 0.5	74.2 ± 0.4	$\mathbf{79.4 \pm 0.3}$
	MIA (%)	35.4 ± 0.6	23.4 ± 0.5	28.8 ± 0.4	23.2 ± 0.3	25.1 ± 0.4	$\underline{22.8 \pm 0.3}$	23.6 ± 0.3	$\mathbf{18.8 \pm 0.2}$
KnowUnDo	USR (%)	74.2 ± 0.7	87.6 ± 0.6	82.7 ± 0.5	87.9 ± 0.4	$\underline{88.9 \pm 0.4}$	88.2 ± 0.4	87.4 ± 0.4	$\mathbf{91.8 \pm 0.3}$
	GUR (%)	89.5 ± 0.3	93.5 ± 0.4	92.4 ± 0.4	93.6 ± 0.3	93.1 ± 0.3	$\underline{94.2 \pm 0.3}$	94.0 ± 0.3	$\mathbf{95.6 \pm 0.2}$
	APR (%)	66.8 ± 0.6	79.2 ± 0.6	73.9 ± 0.5	79.9 ± 0.4	$\mathbf{84.2 \pm 0.3}$	79.6 ± 0.4	78.6 ± 0.4	$\underline{83.9 \pm 0.3}$
	MIA (%)	33.5 ± 0.5	21.8 ± 0.4	27.5 ± 0.4	22.0 ± 0.3	23.4 ± 0.3	$\underline{21.5 \pm 0.3}$	22.1 ± 0.3	$\mathbf{17.2 \pm 0.2}$
TOFU	USR (%)	69.5 ± 0.8	84.7 ± 0.6	78.3 ± 0.5	85.2 ± 0.4	$\underline{85.7 \pm 0.4}$	85.0 ± 0.4	84.3 ± 0.4	$\mathbf{89.0 \pm 0.3}$
	GUR (%)	87.2 ± 0.3	92.2 ± 0.4	90.8 ± 0.4	91.5 ± 0.3	91.2 ± 0.3	$\underline{92.4 \pm 0.3}$	92.1 ± 0.3	$\mathbf{94.4 \pm 0.2}$
	APR (%)	63.1 ± 0.7	75.6 ± 0.6	70.2 ± 0.5	75.8 ± 0.4	$\underline{76.9 \pm 0.4}$	76.0 ± 0.4	75.0 ± 0.4	$\mathbf{80.8 \pm 0.3}$
	MIA (%)	34.7 ± 0.6	22.2 ± 0.5	29.6 ± 0.4	22.5 ± 0.3	24.1 ± 0.3	$\mathbf{17.8 \pm 0.2}$	23.0 ± 0.3	$\underline{18.0 \pm 0.2}$

Table 3: **Comparison with the state-of-the-art LLM unlearning solutions** across datasets. ↑ means higher is better, and ↓ is opposite. Metrics: Unlearning Success Rate (USR) ↑, General Utility Retention (GUR) ↑, Adversarial Probe Rejection (APR) ↑, Membership Inference Attack accuracy (MIA) ↓. Best results in bold, second-best underlined.

in resource-limited settings. Further fine-tuning details of LUNE are provided in the *Detailed Setups* section of the Appendix B.

4.2 Evaluation Metrics

To assess the effectiveness and safety of our proposed unlearning method LUNE, we adopt the following four key evaluation metrics:

Unlearning Success Rate (USR). This metric measures the proportion of unlearning prompts for which the model no longer produces the target (undesired) output. Formally, let $\mathcal{P}_{\text{target}}$ be a set of unlearning prompts, and $\mathcal{A}$ be the set of acceptable outputs (*i.e.*, not containing the target knowledge). The USR is computed as:

$$\text{USR} = \frac{1}{|\mathcal{P}_{\text{target}}|} \sum_{p \in \mathcal{P}_{\text{target}}} \mathbb{1}[f_\theta(p) \in \mathcal{A}], \tag{4}$$

where f_θ is the model and $\mathbb{1}[\cdot]$ is the indicator function.

General Utility Retention (GUR). GUR evaluates the model's performance on tasks unrelated to the unlearned content. We report standard metrics such as accuracy or perplexity on a held-out general-purpose validation set $\mathcal{D}_{\text{gen}}$. A high GUR indicates that the model retains its general knowledge:

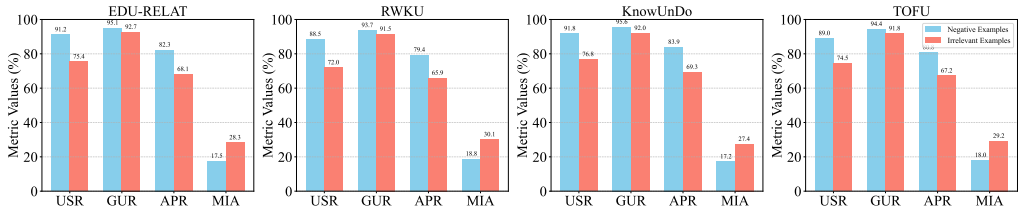
$$\text{GUR} = \frac{\text{Performance}_{\text{after}}(\mathcal{D}_{\text{gen}})}{\text{Performance}_{\text{before}}(\mathcal{D}_{\text{gen}})}. \tag{5}$$

Adversarial Probe Rejection Rate. To assess robustness, we paraphrase unlearning prompts to generate adversarial probes $\mathcal{P}_{\text{adv}}$ that aim to elicit the forgotten information indirectly. The rejection rate is the proportion of these for which the model does not regenerate the target content:

$$\text{Rejection Rate} = \frac{1}{|\mathcal{P}_{\text{adv}}|} \sum_{p \in \mathcal{P}_{\text{adv}}} \mathbb{1}[f_\theta(p) \notin \mathcal{T}], \tag{6}$$

where $\mathcal{T}$ is the set of known target (undesired) outputs.

Membership Inference Attack (MIA) Accuracy. This metric assesses the degree to which the model memorized the target data. We follow standard MIA procedures, where an attacker is given model outputs and must infer whether a given data point was part of the training. Lower accuracy indicates better privacy and effective unlearning.



(a) EDU-RELAT (b) RWKU (c) KnowUnDo (d) TOFU

Figure 3: **Ablation study: negative vs. irrelevant examples.** Fine-tuning with negative examples leads to higher unlearning effectiveness and robustness (↑USR, ↑APR, ↓MIA), while maintaining high utility (↑GUR), consistently outperforming irrelevant examples across all datasets.

4.3 Effectiveness of LUNE

We evaluate **LUNE** against representative baselines spanning three families: (i) data-partitioning or retraining-style methods (GA, NPO), (ii) regularization-based updates (TV), and (iii) partial-parameter methods (SKU, Yao-Neg, LoKU, MemFlex). All methods share the same unlearning setup across four datasets and four metrics (USR↑, GUR↑, APR↑, MIA↓); results are summarized in Table 3. Below, we report key observations (Obs) from a fine-grained comparison across forgetting efficacy, retained utility, adversarial robustness, and privacy.

Obs 1. LUNE achieves the best overall trade-off across datasets. As shown in Table 3, **LUNE** attains the strongest GUR on all four datasets (e.g., EDU-RELAT 95.1%, RWKU 93.7%, KnowUnDo 95.6%, TOFU 94.4%), while also leading most USR/APR entries (e.g., EDU-RELAT APR 82.3%, RWKU APR 79.4%, TOFU APR 80.8%) and consistently minimizing MIA (e.g., EDU-RELAT 17.5%, RWKU 18.8%). Notably, two narrow exceptions appear: Yao-Neg (Full-FT) slightly surpasses us on EDU-RELAT USR (91.6 vs. 91.2) and KnowUnDo APR (84.2 vs. 83.9), while LoKU achieves the lowest MIA on TOFU (17.8 vs. our 18.0). These pockets of strength align with capacity and regularization differences (full-FT's higher capacity benefits single-aspect forgetting; LoRA-based variants can further suppress leakage), yet **LUNE** remains Pareto-favorable on the aggregate.

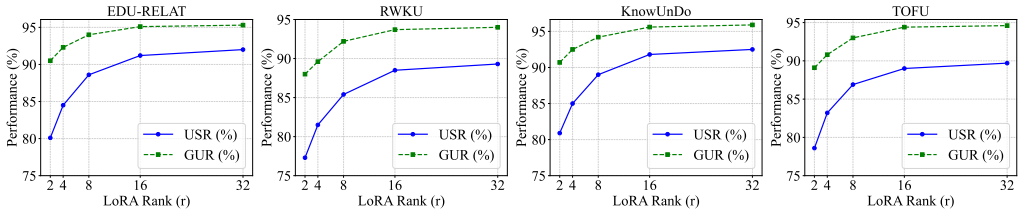
Obs 2. LUNE preserves general utility while delivering strong, robust forgetting. Across datasets, **LUNE** 's GUR is uniformly the best, indicating minimal collateral damage to non-target capabilities (e.g., EDU-RELAT 95.1% and KnowUnDo 95.6%). At the same time, APR, which is our adversarial robustness proxy, is best or near-best in three datasets (e.g., EDU-RELAT 82.3%, RWKU 79.4%, TOFU 80.8%), with only a marginal gap on KnowUnDo (Yao-Neg 84.2% vs. ours 83.9%). This pattern supports the hypothesis that constraining edits to low-rank adapters focuses updates on the intended behaviors, curbing over-forgetting and improving robustness to prompt variants.

Obs 3. Capacity and regularization effects explain remaining gaps. Where full-parameter updates excel (e.g., EDU-RELAT USR 91.6%, KnowUnDo APR 84.2% for Yao-Neg), gains are concentrated on a single axis, often accompanied by weaker generality or privacy relative to **LUNE** (e.g., GUR and MIA). Conversely, LoKU's lowest TOFU MIA (17.8%) highlights the privacy advantage of low-rank updates, yet **LUNE** still balances leakage with superior utility and robustness (higher GUR/APR in the same table). Overall, **LUNE** consistently shifts the utility–forgetting–privacy frontier outward.

Obs 4. LoRA-aware design matters beyond simply "using LoRA." To isolate this factor, Table 8 compares **LUNE** with Yao-Neg (LoRA) under the same negative-only objective. **LUNE** outperforms Yao-Neg (LoRA) on all datasets and metrics (e.g., EDU-RELAT GUR 95.1% vs. 91.1%, RWKU USR 88.5% vs. 83.7%, KnowUnDo MIA 17.2% vs. 24.2%, TOFU APR 80.8% vs. 75.5%). We attribute this consistent margin to our LoRA-specific choices (module selection, rank scanning, early stopping) and multi-view negative construction, which together localize edits and stabilize general utility while maintaining robust forgetting and low leakage.

4.4 Ablation Study: Negative vs. Irrelevant Examples

To further understand what drives effective unlearning, we conduct an ablation study comparing fine-tuning with explicitly constructed negative examples (used in **LUNE**) against a baseline using randomly selected irrelevant examples. This comparison aims to answer: *How can unlearning be achieved efficiently and robustly without full retraining?*



(a) EDU-RELAT (b) RWKU) (c) KnowUnDo (d) TOFU

Figure 4: **The effect of low-rank** r. Performance improves with larger r up to 16, after which gains saturate. Moderate ranks (*e.g.*, $r = 8$ or $r = 16$) offer the best trade-off.

Across all four datasets, we observe consistent and significant performance gains when using negative examples. Specifically, negative-example fine-tuning achieves a higher unlearning success rate (↑USR) and adversarial probe rejection (↑APR), while lowering membership inference attack success (↓MIA), all with minimal compromise to general utility (↑GUR). **(i)** Negative examples provide stronger contrastive supervision, guiding parameter updates more effectively toward forgetting specific information. **(ii)** Random irrelevant examples lack semantic opposition to the target knowledge, thus failing to generate useful gradients for unlearning. **(iii)** The effectiveness of counterfactual-style supervision generalizes well across domains and tasks, validating the robustness of our strategy. These findings support the design choice in **LUNE** to incorporate targeted negative supervision and highlight the importance of principled data construction for reliable and efficient unlearning.

4.5 The Effect of Low-Rank r

To evaluate how the capacity of LoRA adapters influences unlearning effectiveness, we conduct an ablation study by varying the LoRA rank $r \in 2, 4, 8, 16, 32$ across all four datasets. As shown in Figure 4, increasing the rank generally improves both unlearning success rate (USR) and general utility retention (GUR), with the most notable gains occurring between ranks 2 and 16. Beyond $r = 16$, the performance tends to plateau, suggesting diminishing returns with higher parameter capacity. This trend is consistent across datasets, highlighting that moderate-rank LoRA (*e.g.*, $r = 8$ or $r = 16$) provides an optimal trade-off between effectiveness and efficiency. These findings reinforce the practicality of **LUNE** in resource-constrained settings, where low-rank adaptation enables effective unlearning with minimal computational overhead.

As in Figure 4, increasing the LoRA rank generally improves both unlearning success rate (↑USR) and general utility retention (↑GUR), especially in the range from $r = 2$ to $r = 16$. This suggests that higher-rank adapters provide greater expressive capacity to capture the negative gradients necessary for effective forgetting. However, performance gains diminish beyond $r = 16$, and further increases (e.g., $r = 32$) incur additional computation with only marginal improvements. In fact, extremely high-rank adaptation may lead to overfitting or instability in certain cases. (i) A moderate LoRA rank (e.g., $r = 8$ or $r = 16$) offers an optimal trade-off between unlearning effectiveness and training efficiency. In contrast, (ii) very low ranks ($r = 2$ or $r = 4$) underperform, resulting in degraded USR and slightly reduced GUR, likely due to insufficient parameter capacity to accommodate meaningful updates. These observations remain consistent across all datasets, despite differences in domain and task structure. (iii) The rank-performance relationship is stable and generalizable, reinforcing the robustness of **LUNE**'s design. (iv) **LUNE** maintains competitive performance even under low-rank settings, which highlights its practicality for computationally constrained scenarios, making it a viable unlearning solution for large-scale LLMs with limited resources.

5 Conclusion

We present **LUNE** 🌙, a LoRA-based, negative-only unlearning framework that edits *only* lightweight adapters to remove targeted knowledge efficiently. By localizing updates, **LUNE** suppresses undesired behaviors while preserving general ability, mitigating catastrophic drift. Across four benchmarks, it achieves strong unlearning (USR/APR) with superior utility retention (GUR) and lower leakage (MIA). Future work includes improving cross-task generalization and extending to concept- and multi-instance forgetting.

References

[1] Bourtoule, L., V. Chandrasekaran, C. A. Choquette-Choo, et al. Machine unlearning. In *USENIX Security*. 2021.

[2] Golatkar, A., A. Achille, S. Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9304–9312. 2020.

[3] Kirkpatrick, J., R. Pascanu, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 2017.

[4] Yao, J., E. Chien, M. Du, et al. Machine unlearning of pre-trained large language models. *arXiv preprint arXiv:2402.15159*, 2024.

[5] Meng, K., D. Bau, A. Andonian, et al. Locating and editing factual associations in gpt. *arXiv:2202.05262*, 2022. ROME.

[6] Meng, K., A. Hernandez, D. Bau, et al. Mass-editing memory in a transformer. *arXiv:2210.07229*, 2022.

[7] Yao, Y., X. Xu, Y. Liu. Large language model unlearning. *Advances in Neural Information Processing Systems*, 37:105425–105475, 2024.

[8] Zhang, R., L. Lin, Y. Bai, et al. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*, 2024.

[9] Fan, C., J. Liu, L. Lin, et al. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. In *ICLR*. 2025.

[10] Liu, Z., D. Dou, et al. Towards safer large language models through machine unlearning. In *Findings of ACL*. 2024.

[11] Hu, E. J., Y. Shen, P. Wallis, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[12] Dettmers, T., A. Pagnoni, A. Holtzman, et al. Qlora: Efficient finetuning of quantized llms. In *NeurIPS*. 2023.

[13] Bourtoule, L., V. Chandrasekaran, C. A. Choquette-Choo, et al. Machine unlearning. In *USENIX Security*. 2021. ArXiv:1912.03817.

[14] Liu, S., Y. Yao, J. Jia, et al. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, pages 1–14, 2025.

[15] Hong, Y., H. Yang, et al. Dissecting fine-tuning unlearning in large language models. In *EMNLP*. 2024.

[16] Fan, C., J. Liu, L. Lin, et al. Simplicity prevails: Rethinking negative preference optimization for llm unlearning. In *ICLR*. 2025.

[17] Cha, S., S. Cho, D. Hwang, et al. Towards robust and parameter-efficient knowledge unlearning in llms. *arXiv:2408.06621*, 2024. LoKU (LoRA-based).

[18] Tian, B., et al. To forget or not? towards practical knowledge unlearning for llms. In *Findings of EMNLP*. 2024.

[19] Wang, Q., et al. Rethinking llm unlearning objectives. In *ICLR*. 2025.

[20] Ding, Z., et al. Cure: Scalable llm unlearning by correcting responses. *OpenReview preprint*, 2025.

[21] Zhang, K., et al. A survey of machine unlearning in large language models. *arXiv:2503.01854*, 2025.

[22] Houlsby, N., A. Giurgiu, S. Jastrebski, et al. Parameter-efficient transfer learning for nlp. In *ICML*. 2019.

[23] Ben Zaken, E., Y. Goldberg, S. Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv:2106.10199*, 2022.

[24] Zhang, R., X. Li, J. He, et al. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*. 2023. AdaLoRA.

[25] Liu, C., T. Sun, J. Zhu, et al. Dora: Weight-decomposed low-rank adaptation. *arXiv:2402.09353*, 2024.

[26] Pfeiffer, J., I. Vulić, I. Gurevych, et al. Adapterfusion: Non-destructive task composition for transfer learning. In *EACL*. 2021.

[27] Rücklé, A., J. Pfeiffer, S. Ruder, et al. Adapterdrop: On the efficiency of adapters in transformers. In *EMNLP Workshop on Efficient NLP*. 2020.

[28] Lester, B., R. Al-Rfou, N. Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*. 2021.

[29] Li, X. L., P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*. 2021.

[30] Liu, X., K. Ji, Y. Fu, et al. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv:2110.07602*, 2022.

[31] Karimi Mahabadi, R., J. Henderson, S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *NeurIPS*. 2021.

[32] Guo, D., A. M. Rush, Y. Kim. Parameter-efficient transfer learning with diff pruning. In *NeurIPS*. 2021.

[33] Zhang, R., F. Han, C. Zhang, et al. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. In *ICCV Workshop*. 2023.

[34] Gao, F., X. Li, X. Wang, et al. Fedpt: Federated prompt tuning for large language models. *arXiv:2401.04084*, 2024.

[35] Aghajanyan, A., S. Gupta, L. Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv:2012.13255*, 2020.

[36] Xu, H., A. Sharaf, Y. Chen, et al. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*, 2024.

[37] Ginart, A., M. Guan, G. Valiant, et al. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

[38] Wu, R., C. Yadav, R. Salakhutdinov, et al. Evaluating deep unlearning in large language models. *arXiv preprint arXiv:2410.15153*, 2024.

[39] Jin, Z., P. Cao, C. Wang, et al. Rwku: Benchmarking real-world knowledge unlearning for large language models. *arXiv preprint arXiv:2406.10890*, 2024.

[40] Maini, P., Z. Feng, A. Schwarzschild, et al. Tofu: A task of fictitious unlearning for llms. *arXiv preprint arXiv:2401.06121*, 2024.

[41] Ilharco, G., M. T. Ribeiro, M. Wortsman, et al. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[42] Jang, J., D. Yoon, S. Yang, et al. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504*, 2022.

[43] Jiang, A. Q., A. Sablayrolles, A. Mensch, et al. Mistral 7b, 2023.

[44] Child, R., S. Gray, A. Radford, et al. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[45] Beltagy, I., M. E. Peters, A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[46] Touvron, H., L. Martin, K. Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

1. **Claims**

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

   Answer: [Yes]

   Justification: Please see our Abstract section.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Please refer to the discussion in the "Conclusion" section (see Section 5).

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Please refer to Algorithm 1 for all assumptions.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Please refer to the experimental setup and dataset/model description in Appendices B and D and Table 2 and the main results in Table 3.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The data used in this paper are publicly accessible datasets. We will release the code with the camera-ready version.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Please refer to Appendices B and D for the detailed experimental setup.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report the mean and standard deviation over 3 independent runs.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Please see Table 5 for the per-dataset training configuration and compute-related settings.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read the NeurIPS Code of Ethics, and our paper conforms to it.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Please see Section 5.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The assets used in this paper are publicly available.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

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

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

    Answer: [NA]

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

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [NA]

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

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Detailed Introduction to Baselines

- **Gradient Ascent (GA)** [42] This baseline utilizes gradient ascent to maximize the model's loss on the target information, explicitly discouraging the model from generating undesired content. While simple, GA is computationally expensive and may negatively impact unrelated knowledge due to aggressive updates.
- **NPO.** [8] Frames unlearning as *preference optimization* against negative responses: the model is trained to prefer safe/non-target outputs over negative ones. Compared with naive negative-only FT, NPO usually offers a more stable trade-off between forgetting and utility, given well-constructed preference pairs.
- **Task Vector (TV).** [41] Task Vector identifies a specific direction in the parameter space of the pretrained model corresponding to the target knowledge. Unlearning is achieved by subtracting this direction from the model parameters. While efficient, it can unintentionally affect semantically related knowledge.
- **MemFlex.** [18] MemFlex employs gradient-based optimization to precisely remove sensitive parameters associated with undesired information. This method maintains general knowledge but demands access to gradient computations across large parameter subsets, increasing computational complexity.
- **Yao-Neg.** [7] Unlearning is performed by *fine-tuning only on negative examples*, updating all model parameters to suppress targeted knowledge/behaviors. In the main experiment (Table 3), we report *Yao–Neg* in its *full fine-tuning* configuration, which the original paper presents as a primary/standard instantiation alongside an optional LoRA variant; for completeness, results for the *LoRA* version appear in the appendix (Table 8), computed under the same negative-only setup.
- **LoKU.** [17] A *LoRA-based* unlearning approach that freezes the backbone and trains low-rank adapters, often with stabilizing regularizers. It is parameter-efficient and localizes edits, typically retaining utility better and reducing leakage compared to full-parameter updates.

B More Setups

Additional setup. Unless otherwise noted, we train **LUNE** with *negative-only* supervision using the same data splits and evaluation protocol as the main text, and keep all backbone weights frozen while updating only LoRA adapters. We apply LoRA to attention projections (W_q, W_k, W_v, W_o) and the FFN up/down projections, initialize adapters to zero, and use a default rank r=16 (chosen from an ablation over $r \in \{2, 4, 8, 16, 32\}$ where gains saturate near 16); LoRA dropout is 0.05 and scaling α=r. Optimization uses AdamW with learning rate 2×10^{-4}, linear warmup over the first 5% of steps, cosine decay thereafter, weight decay 0.01, gradient clipping at 1.0, mixed-precision (bf16), and gradient accumulation to match an effective batch size of 256 tokens/step. Inputs are tokenized with the backbone tokenizer; we cap sequence length at $1{,}024$ for training and evaluation, pad on the right, and mask loss to target spans only. Early stopping on a held-out negative-dev set monitors USR↑/APR↑ at fixed GUR↑ tolerance (≤ 0.5pp drop). For fairness across methods, we keep the total #steps per dataset aligned to the epoch budgets reported in the paper (Tables 2 and 5) and repeat each run with three random seeds, reporting mean $\pm$ standard error of the mean for all metrics.

B.1 LLM Backbones

`Mistral 7B`. It is a 7b-parameter LLM by Mistral AI [43], effectively handles text generation and diverse NLP tasks, whose benchmark covers areas like commonsense reasoning, world knowledge, math, and reading comprehension, showcasing its broad applicability. It utilizes a sliding window attention mechanism [44, 45], supports English and coding languages, and operates with an 8k context length.

`LLaMA-2 7B`. `LLaMA-2` (Large Language Model Meta AI) is an open-source family of autoregressive transformer-based language models released by Meta AI [46]. The 7B variant offers a strong balance between performance and computational efficiency, making it a practical choice for fine-tuning and unlearning experiments on consumer-level hardware. Pretrained on a diverse mix of publicly available data, `LLaMA-2 7B` exhibits competitive language understanding and generation

capabilities compared to larger proprietary models, while remaining accessible for research and reproducibility.

C More Experiments

C.1 Comparison of LoRA vs. Full Fine-Tuning

To assess the efficiency and effectiveness of our proposed method, we compare LUNE, which fine-tunes only a small set of LoRA adapters, with traditional full fine-tuning that updates all model parameters. As shown in Table 4, LUNE consistently achieves comparable or superior performance across all evaluation metrics. Specifically, LUNE outperforms full fine-tuning in unlearning success rate (USR) and adversarial robustness (APR), while also maintaining higher general utility (GUR) and achieving lower MIA accuracy, indicating improved privacy. These results highlight that LoRA-based adaptation not only reduces computational cost but also enables more targeted and reliable unlearning, making it a more practical and scalable approach for real-world applications.

Dataset	Method	USR (%)	GUR (%)	APR (%)	MIA (%)
EDU-RELAT	Full FT	88.7	90.2	79.5	25.4
	LUNE	**91.2**	**95.1**	**82.3**	**17.5**
RWKU	Full FT	85.1	89.4	76.0	27.8
	LUNE	**88.5**	**93.7**	**79.4**	**18.8**
KnowUnDo	Full FT	89.0	91.5	80.2	24.1
	LUNE	**91.8**	**95.6**	**83.9**	**17.2**
TOFU	Full FT	86.4	89.9	77.3	26.5
	LUNE	**89.0**	**94.4**	**80.8**	**18.0**

Table 4: Ablation study comparing LoRA-based fine-tuning (LUNE) and full fine-tuning (FT) on all parameters across four datasets. Metrics shown include Unlearning Success Rate (USR), General Utility Retention (GUR), Adversarial Probe Rejection (APR), and Membership Inference Attack accuracy (MIA). The best results are highlighted in bold.

D Detailed Setups

To ensure effective unlearning while preserving general model utility, we fine-tune the LoRA adapters in LUNE for a fixed number of epochs per dataset. The number of epochs is selected based on the dataset size, complexity, and observed convergence behavior during preliminary experiments. As summarized in Table 5, smaller or synthetic datasets such as TOFU and EDU-RELAT benefit from slightly more epochs to reinforce the unlearning signal, while larger datasets like RWKU require fewer epochs due to greater example diversity per iteration. This setup strikes a balance between unlearning effectiveness and training efficiency across diverse domains.

Dataset	# Samples	Epochs	Task
EDU-RELAT	10,000	30	Synthetic relational data; quick convergence
RWKU	13,000	40	Larger real-world factual dataset
KnowUnDo	8,000	35	Entity privacy-sensitive task
TOFU	4,000	50	Smaller profile data; longer tuning needed

Table 5: Number of training epochs used for LUNE on each dataset. The values are selected based on dataset size and convergence behavior.

Dataset	Original Fact	Alternative Negative Example
EDU-RELAT	John's brother is Mike.	John's brother is Kevin.
RWKU	The capital of France is Paris.	The capital of France is Lyon.
KnowUnDo	Alice works at Microsoft.	Alice works at Google.
TOFU	Author Alex Smith primarily writes science fiction novels.	Author Alex Smith primarily writes romance novels.

Table 6: Negative examples generated by replacing true facts with alternative (erroneous) information for each dataset.

D.1 Negative Examples Generation

To effectively guide the unlearning process, we explicitly generate negative examples designed to counteract previously learned, undesired knowledge. Table 6 provides illustrative examples of alternative negative examples tailored specifically for each dataset used in our evaluation. The generation of negative examples follows these key steps:

- **Identify Target Knowledge**: Clearly define the specific facts, associations, or behaviors that the model should unlearn.
- **Construct Contradictory or Alternative Statements**: Create statements that explicitly contradict or replace the targeted information. These statements can be either directly contradictory (*e.g.*, *"The capital of France is not Paris"*) or alternative erroneous facts (*e.g.*, *"The capital of France is Lyon"*).
- **Paraphrase and Diversify Examples**: Generate multiple paraphrased variations to enhance robustness and generalization of the unlearning effect across different prompt forms and phrasings.
- **Validate and Curate**: Verify that the negative examples clearly and effectively negate or overwrite the undesired knowledge without introducing unintended biases or misinformation beyond the targeted scope.

D.2 Quality of the Example

To evaluate the impact of negative example quality on unlearning performance, we construct three variants of training data representing different levels of semantic clarity. **High-quality examples** are explicit contradictions of the target knowledge, such as *"The capital of France is Lyon"*, which directly oppose the fact to be unlearned.

Medium-quality examples introduce subtle ambiguity or hedging without fully committing to a contradictory statement, for example, *"Paris may not always be considered France's capital"*. These examples may confuse the model rather than guiding it to forget.

Low-quality examples are loosely related or entirely irrelevant statements, such as *"France has many important cities"*, which lack any clear corrective signal. By fine-tuning `LUNE` on each variant independently, we assess how the clarity of the negative supervision affects targeted unlearning, general knowledge retention, and robustness to adversarial prompts.

Neg. Examples Quality	USR (%)	APR (%)	GUR (%)	MIA (%)
High Quality	88.5	79.4	93.7	18.8
Medium Quality	78.0	67.2	91.0	26.5
Low Quality	65.3	53.9	89.4	32.0

Table 7: **Impact of negative example quality** on the RWKU dataset. "Neg." is abbreviated for Negative. Higher-quality negative examples yield stronger unlearning effectiveness and robustness while preserving utility, demonstrating the importance of clarity and specificity in negative supervision.

The results in Table 7 demonstrate that the quality of negative examples plays a pivotal role in the performance of `LUNE`. Specifically, we observe a strong correlation between the clarity and specificity of negative examples and the model's unlearning effectiveness and robustness.

Dataset	Metric	Yao-Neg (LoRA)	LUNE (Ours)
EDU-RELAT	USR (%)	89.3 ± 0.4	$\mathbf{91.2 \pm 0.3}$
	GUR (%)	91.1 ± 0.3	$\mathbf{95.1 \pm 0.2}$
	APR (%)	78.0 ± 0.4	$\mathbf{82.3 \pm 0.3}$
	MIA (%)	23.6 ± 0.3	$\mathbf{17.5 \pm 0.2}$
RWKU	USR (%)	83.7 ± 0.4	$\mathbf{88.5 \pm 0.3}$
	GUR (%)	88.6 ± 0.3	$\mathbf{93.7 \pm 0.2}$
	APR (%)	74.8 ± 0.5	$\mathbf{79.4 \pm 0.3}$
	MIA (%)	26.0 ± 0.3	$\mathbf{18.8 \pm 0.2}$
KnowUnDo	USR (%)	87.4 ± 0.4	$\mathbf{91.8 \pm 0.3}$
	GUR (%)	92.0 ± 0.3	$\mathbf{95.6 \pm 0.2}$
	APR (%)	80.1 ± 0.4	$\mathbf{83.9 \pm 0.3}$
	MIA (%)	24.2 ± 0.3	$\mathbf{17.2 \pm 0.2}$
TOFU	USR (%)	84.2 ± 0.4	$\mathbf{89.0 \pm 0.3}$
	GUR (%)	90.3 ± 0.3	$\mathbf{94.4 \pm 0.2}$
	APR (%)	75.5 ± 0.4	$\mathbf{80.8 \pm 0.3}$
	MIA (%)	24.9 ± 0.3	$\mathbf{18.0 \pm 0.2}$

Table 8: Comparison between Yao-Neg (LoRA) and our method across datasets. Best results in **bold**. (For MIA, lower is better.)

- USR: High-quality negative examples lead to a significantly higher USR (88.5%) compared to medium (78.0%) and low-quality (65.3%) examples. This indicates that explicitly contradicting the undesired information is essential for effectively suppressing the model's prior knowledge. When the negative example is vague or only weakly related, the model struggles to identify which behavior to unlearn.
- APR: A similar trend is observed in the model's robustness to paraphrased prompts. With high-quality examples, APR reaches 79.4%, but drops significantly with medium (67.2%) and low-quality (53.9%) examples. This suggests that only clear and direct contradictions can generalize well to variations in user input.
- GUR: All example types maintain relatively high GUR, though high-quality examples preserve it best (93.7%), slightly outperforming medium (91.0%) and low-quality (89.4%) examples. Notably, poorly crafted examples can interfere with unrelated knowledge, leading to a minor degradation in general performance due to noisier gradient updates.
- MIA Accuracy: Lower MIA accuracy with high-quality examples (18.8%) reflects more successful removal of memorized facts. In contrast, higher MIA accuracy for low-quality examples (32.0%) implies that vague or unrelated examples fail to overwrite the memorized information, leaving the model vulnerable to inference attacks.

D.3 Additional Comparison

We additionally instantiate Yao et al.'s negative-only unlearning with *LoRA* adapters and report its results separately in Table 8. Unless noted, we reuse the same datasets, negative-example construction, and training budgets as in Table 3. The only change is the optimization regime: instead of full-parameter fine-tuning used by *Yao-Neg (Full-FT)* in the main table, we enable LoRA with the standard target modules (attention/FFN) and the same rank/search protocol as our method, while keeping all other hyperparameters identical. This isolates the effect of using LoRA under the *same* negative-only objective and makes the comparison to our *LoRA-only* design fair and transparent.

Across all datasets and metrics in Table 8, our method consistently outperforms *Yao-Neg (LoRA)*. We attribute this to three factors. (i) **LoRA-aware design**: our training is tailored to low-rank adapters (module selection, rank scanning, and early stopping), which localizes edits and stabilizes utility, whereas *Yao-Neg (LoRA)* directly ports the negative-only objective without LoRA-specific regularization. (ii) **Robust negatives**: our multi-view negative construction (paraphrase/counterfactual/retrieval variants) improves robustness, yielding higher USR/APR under the same budget. (iii) **Drift control**: by freezing the backbone and constraining updates to low-rank adapters, our method reduces unintended drift (reflected by higher GUR and lower MIA). Notably, the gains hold on both medium-

scale (EDU-RELAT, RWKU) and large-scale datasets (KNOWUNDO, TOFU), indicating that our LoRA-specific design scales while preserving strong forgetting–utility trade-offs.

E Additional Discussion

To provide a balanced view, we complement the earlier discussion of limitations with a brief account of our method's merits in this Appendix.

The proposed **LUNE** method introduces a lightweight and targeted approach to unlearning in LLMs, offering several key advantages. First, it is highly *parameter-efficient*, as it fine-tunes only the low-rank LoRA matrices A and B, drastically reducing the number of trainable parameters compared to full model fine-tuning. This not only lowers computational and memory demands but also makes **LUNE** practical in resource-constrained settings. Second, **LUNE** ensures *preservation of original knowledge* by freezing the pre-trained model weights W_0, thereby maintaining the model's general capabilities and minimizing the risk of catastrophic forgetting. Third, **LUNE** performs *targeted unlearning* by fine-tuning exclusively on negative examples, allowing the model to forget specific information without requiring access to the full training set, a valuable feature when data availability is limited or privacy-sensitive. Altogether, **LUNE** offers an effective, scalable, and focused solution for unlearning in LLMs by combining LoRA's parameter efficiency with task-specific negative supervision.