METAFORGET: Audit-Driven Update-Policy Learning for Language Model Unlearning

Pinlong Zhao[1*] **Xiaoling Zhou**[2] **Zhou Zhaoting**[3] **Guangyuan Dong**[4,5*]
[1]School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China
[2]Peking University [3]Northeastern University
[4]NExT++ Research Centre, National University of Singapore
[5]Ant International Research

Abstract

Deployed language models are increasingly expected to remove private, copyrighted, or hazardous content without full retraining. Existing unlearning methods remain brittle because they usually optimize a fixed forgetting objective with a global update schedule: weak updates leave paraphrased knowledge intact, while stronger updates can damage retained capabilities or nearby facts. We identify this failure mode as *update homogeneity*. To address it, we introduce METAFORGET, a meta-optimization framework that treats unlearning as learning an audit-driven update policy rather than designing another standalone forgetting loss. A lightweight controller maps memorization, paraphrase-sensitivity, and signed forget-retain conflict signals to span weights, layer gates, step sizes, and retain-gradient projection strengths. The controller is trained with a bilevel objective that differentiates through short LoRA unlearning trajectories and evaluates the resulting model on held-out forget, retain, locality, and leakage audits. This policy can instantiate NPO- or RMU-style losses while changing where and how strongly they act. Across TOFU, WMDP, and MUSE-style sequential deletion settings, METAFORGET improves the forget-retain-robustness frontier over strong baselines, moves loss-based membership-inference diagnostics closer to chance, and better preserves utility under repeated requests. The results support a view of practical LLM unlearning as audit-driven policy learning: the central object is not only the forget loss, but the localized update that must pass the deletion audit.

1 Introduction

Machine unlearning asks whether a trained model can be updated so that it behaves as if selected data or knowledge had never been learned. For modern language models, this goal is no longer a theoretical luxury: removal requests may concern private training records, copyrighted passages, or hazardous capabilities, while retraining from scratch is often economically infeasible. Recent benchmarks such as TOFU, WMDP, and MUSE have made the tension clear: an unlearned model must forget the targeted content, avoid leaking evidence of prior exposure, preserve unrelated utility, scale to large deletion requests, and remain stable under repeated requests [Maini et al., 2024, Li et al., 2024, Shi et al., 2025].

Current LLM unlearning algorithms make progress but are still brittle. Gradient ascent directly increases loss on the forget set, yet can collapse the model into nonsensical behavior [Yao et al., 2024b, Zhang et al., 2024]. Negative Preference Optimization (NPO) slows collapse by casting forget examples as negative preferences [Zhang et al., 2024, Rafailov et al., 2023], but it assigns

*Corresponding authors and project leads: pinlongzhao@hdu.edu.cn, guangyuan@u.nus.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

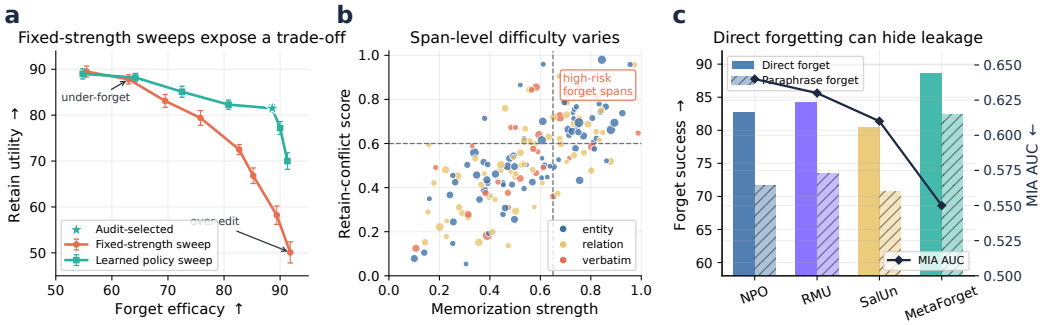


Figure 1: **Forgetting difficulty is heterogeneous.** (a) Sweeping a fixed unlearning strength exposes a brittle forget-retain trade-off: weak updates leave target knowledge, while strong updates damage retained utility. A learned update policy improves the high-forget region of this frontier by changing where and how the update is applied, while still trading off against retention. (b) Candidate forget spans vary substantially in memorization strength and retain-conflict score, so uniform token pressure is poorly matched to the deletion request. Dashed lines mark high-risk diagnostic thresholds. (c) Direct forgetting can overestimate deletion quality: methods that suppress the original prompt may still leak under paraphrases or membership-inference probes. These diagnostics motivate an update policy trained with held-out forget, retain, locality, and leakage audits.

similar pressure to all tokens in a forget response. Representation Misdirection for Unlearning (RMU) suppresses hazardous knowledge by steering internal activations [Li et al., 2024], but the same representation perturbation may affect benign concepts entangled with the target. Saliency-based methods restrict updates to important parameters [Fan et al., 2024], yet their saliency masks are usually computed once and do not adapt to the evolving forget-retain trade-off.

We identify a common cause: **update homogeneity**. Existing methods largely assume that a deletion request can be handled by a fixed loss, editable layer range, and step schedule. This assumption hides three sources of heterogeneity, illustrated by the diagnostics in Figure 1. First, different spans within the same example have different roles: names and relations often carry the protected fact, while surrounding syntax is shared with benign language. Second, forget examples differ in their retention risk: some facts are isolated, whereas others are semantically entangled with broad world knowledge. Third, robustness varies across reformulations: an update that blocks the original prompt may fail under paraphrased questions, inverse-relation probes, or membership-inference tests. Thus, simply increasing a loss coefficient is the wrong level of control. It is an *update policy* that decides where, how strongly, and in which subspace an existing unlearning objective should act.

We propose METAFORGET, a meta-optimization framework for LLM unlearning built around this policy view. The key idea is to learn a small controller that adapts each unlearning update using retention-aware signals. Given a forget batch and a compact retention/meta set, the controller predicts (i) span-level unlearning weights, (ii) layer gates over a LoRA update subspace, (iii) step sizes, and (iv) retain-gradient projection strengths. These outputs define a deletion policy trained by a bilevel objective: the inner loop performs a few differentiable unlearning steps using weighted NPO/RMU losses, while the outer loop scores the resulting model on held-out forget paraphrases, retention tasks, locality probes, and leakage diagnostics. Unlike validation-based early stopping, the meta-objective changes the update itself before the final audit is run. This distinction lets METAFORGET learn reusable allocation rules, such as focusing on high-memorization spans, avoiding low-value syntax spans, and attenuating destructive directions when a target is entangled with retained knowledge.

Our contributions are:

- We formulate LLM unlearning as *audit-driven update-policy learning*: a bilevel view in which validation audits supervise span-, layer-, step-size-, and projection-level update decisions.
- We instantiate this view with METAFORGET, a practical controller that combines retention-aware meta signals, span reweighting, layer gating, and signed retain-gradient projection in a parameter-efficient LoRA subspace. The framework is compatible with NPO and RMU losses, but avoids treating their coefficients as the only available control knob.

- We evaluate the learned policy under direct recall, paraphrase recall, privacy leakage, locality, sequential deletion, runtime, and cross-model evaluation. The results show improved forget-retain frontiers and stronger audit robustness against competitive unlearning baselines.

2 Related work

LLM unlearning. Early unlearning work studied data deletion in classical models, with exact or approximate strategies such as SISA training [Bourtoule et al., 2021]. LLM unlearning is harder because the removal unit may be an example, a passage, an entity, or a capability, and its influence is distributed through model parameters. Gradient ascent on forget data is simple but unstable [Yao et al., 2024b]. NPO reframes forgetting as a one-sided preference objective and substantially improves stability [Zhang et al., 2024]. RMU changes internal representations for hazardous prompts while preserving benign activations [Li et al., 2024], and later work studies layer-adaptive steering coefficients [Dang et al., 2025]. Recent methods also use adjusted-logit self-distillation, normalized gradient balancing, and positive relearning to reduce over-unlearning [Dong et al., 2025, Jin et al., 2025, Xu et al., 2025]. Model-editing and copyright removal methods demonstrate targeted erasure, but evaluations show that exact removal, retention, and privacy leakage remain difficult to satisfy simultaneously [Eldan and Russinovich, 2023, Shi et al., 2025, Dorna et al., 2025]. METAFORGET is complementary: it learns the update policy that allocates these base losses across spans, layers, and retain-gradient subspaces.

Evaluation and robustness. TOFU provides synthetic author profiles with controlled forget splits [Maini et al., 2024]. WMDP evaluates hazardous knowledge in biosecurity, cybersecurity, and chemical security and proposes RMU as a mitigation method [Li et al., 2024]. MUSE broadens evaluation to six desiderata, including memorization removal, privacy leakage, scalability, and sustainability [Shi et al., 2025]. These benchmarks reveal that apparent forgetting can disappear under paraphrase, membership inference, or sequential requests. Our method directly trains on held-out paraphrases and locality probes to improve these failure modes.

Meta-learning and example reweighting. Meta-learning optimizes learning procedures using validation feedback [Finn et al., 2017, Hospedales et al., 2021]. Example reweighting methods learn sample weights from a clean validation set [Ren et al., 2018]. Gradient surgery and projection methods reduce interference in multi-task learning [Yu et al., 2020]. METAFORGET adapts these ideas to unlearning, where the meta-objective is asymmetric: it must degrade target behavior while preserving non-target capabilities and privacy. This asymmetry requires span-level forget weights, retention-aware projection, and verification under paraphrase.

3 Audit-Driven Update-Policy Learning

Figure 2 summarizes the full pipeline. METAFORGET separates the *base unlearning objective* from the *policy that applies it*. The base objective is still NPO for text deletion and RMU for hazardous-knowledge suppression; the learned policy decides which spans receive pressure, which layers are edited, how large the step is, and how much of the retain-conflicting direction is removed. There are two optimization levels. In the inner loop, the current policy updates only LoRA parameters for a single deletion request. In the outer loop, held-out forget, retain, locality, and leakage audits update the controller so that future inner-loop updates are better localized. Formally, METAFORGET learns a policy $\pi_\phi : (s, q) \mapsto (w, a, \eta, \rho)$ from span features s and request statistics q to span weights w, layer gates a, step sizes η, and projection strengths ρ; held-out audits supervise the policy rather than serving as controller inputs.

Two design choices keep this policy view distinct from ordinary validation tuning. First, at test time the controller consumes only pre-audit request signals; held-out paraphrases and neighboring retain prompts supervise meta-training but remain disjoint from final benchmark prompt families. Second, the policy modulates a fixed update budget rather than expanding it: comparable methods use the same LoRA rank, maximum inner-step budget, retain buffer, decoding rule, and stopping criteria. The learned object is therefore an allocation rule over spans, layers, and gradient subspaces, not a larger online search.

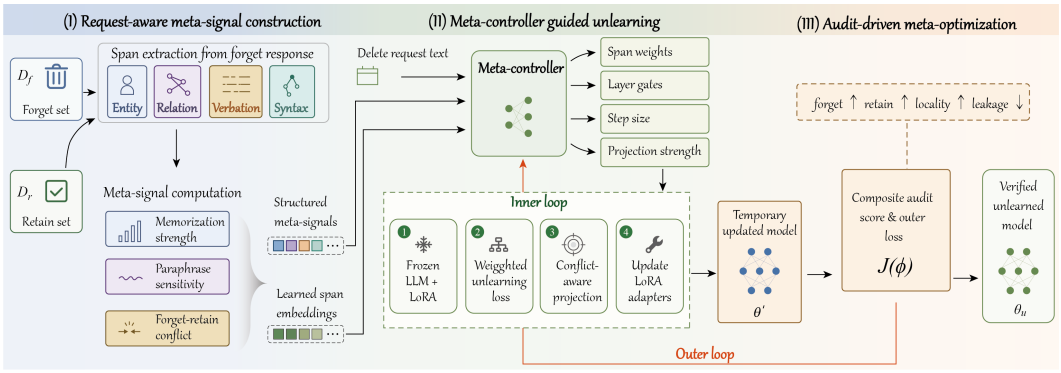


Figure 2: **METAFORGET overview.** A controller reads memorization, paraphrase, and gradient-conflict signals and outputs span weights, layer gates, step sizes, and projection strengths. The inner loop performs weighted unlearning in a gated LoRA subspace with conflict-aware projection. The outer loop differentiates through the update and optimizes held-out forget, retention, locality, and leakage surrogates.

3.1 Problem setup

Let p_{θ_0} be a language model and let $\mathcal{D}_f = \{(x_i, y_i)\}$ denote the content to forget. Let $\mathcal{D}_r$ be a retain set representing capabilities and nearby benign knowledge that should be preserved. We split small held-out meta sets $\mathcal{V}_f$ and $\mathcal{V}_r$ from deletion and retention data, and construct a neutral locality set $\mathcal{D}_0$ from unrelated prompts. The desired updated model p_{θ_u} should (i) assign low likelihood to target responses and refuse or answer generically on target prompts, (ii) remain close to p_{θ_0} on $\mathcal{D}_r \cup \mathcal{D}_0$, and (iii) avoid privacy leakage that reveals membership of $\mathcal{D}_f$.

For a response $y_i = (y_{i1}, \ldots, y_{iT})$, standard NPO uses a negative preference loss that decreases the relative likelihood of the forget response under the unlearned model compared with a reference model [Zhang et al., 2024]. We use a generic base unlearning loss $\ell_f(\theta; x_i, y_i)$, instantiated as NPO for text deletion and RMU for hazardous-knowledge suppression. A retain loss $\ell_r(\theta; \mathcal{D}_r)$ is the usual supervised or KL-preservation loss on $\mathcal{D}_r$:

$$\ell_r(\theta; \mathcal{D}_r) = \mathbb{E}_{(x,y)\sim\mathcal{D}_r}[-\log p_\theta(y|x)] + \lambda_0 \mathbb{E}_{x\sim\mathcal{D}_0} \mathrm{KL}(p_{\theta_0}(\cdot|x) \| p_\theta(\cdot|x)) \,. \tag{1}$$

Uniform unlearning minimizes a fixed combination of ℓ_f and ℓ_r with a shared layer range and scalar learning-rate schedule. METAFORGET instead learns the coefficients and update subspace from meta-feedback, while keeping the underlying forget and retain losses recognizable. Appendix A gives a first-order retain-risk view that motivates the projection and span-weighting design.

3.2 Retention-aware meta signals

The controller receives a feature vector s_{ij} for each candidate span or token group j in forget example i. We group tokens using named-entity spans, quoted/verbatim spans, relation phrases, and residual syntax spans; in implementation this is obtained by a lightweight tagger plus byte-pair alignment. The features summarize three main risks plus auxiliary metadata.

Memorization strength. We measure how strongly the base model stores a span:

$$m_{ij} = \log p_{\theta_0}(y_{ij}|x_i, y_{i,<j}) - \log p_{\mathrm{ref}}(y_{ij}|x_i, y_{i,<j}), \tag{2}$$

where p_{ref} is a smaller model or an early checkpoint. Large m_{ij} indicates that the span is unusually likely under the trained model and should receive more forget pressure.

Paraphrase sensitivity. Let $\mathcal{P}(x_i)$ be feature-construction paraphrases of the prompt, disjoint from final audit prompt families. We define

$$\pi_{ij} = \mathrm{Var}_{\tilde{x}\in\mathcal{P}(x_i)} \log p_\theta(y_{ij} \mid \tilde{x}, y_{i,<j}), \tag{3}$$

which measures how unstable forgetting is across prompt rewrites. High π_{ij} marks brittle apparent forgetting and suggests that the deletion should be audited beyond the original template.

Forget-retain conflict. Let $g_{ij}^f = \nabla_\theta \ell_f^{ij}$ be the span-level forget gradient and let $g^r = \nabla_\theta \ell_r$ be the retention gradient, both estimated in the LoRA subspace. Because the inner loop performs gradient descent on the forget objective, destructive conflict occurs when these gradients point in opposite directions. We therefore compute

$$\kappa_{ij} = \max\left(0, -\frac{\langle g_{ij}^f, g^r \rangle}{\|g_{ij}^f\| \, \|g^r\| + \epsilon}\right), \tag{4}$$

where large positive values indicate that improving the forget objective would likely increase retain loss. We also include gradient norm, span type, position, and current training step. These signals are standardized within each request.

3.3 Meta-controller and inner unlearning update

The controller h_ϕ is a two-layer MLP with a request-level Transformer pooling block; architecture, normalization, and hyperparameters are detailed in Appendix B. It maps each span feature s_{ij} to a nonnegative weight w_{ij}, each layer ℓ to a gate $a_\ell \in [0,1]$, and the current request to a step size η and projection strength $\rho \in [0,1]$:

$$\{w_{ij}\}, \{a_\ell\}, \eta, \rho = h_\phi(\{s_{ij}\}_{i,j}, q), \quad w_{ij} = \mathrm{softplus}(u_{ij}), \quad a_\ell = \sigma(v_\ell), \quad \rho = \sigma(r), \tag{5}$$

where q contains request-level statistics such as forget-set size and average conflict.

We update only LoRA parameters Δ_ψ inserted into attention and MLP projection matrices [Hu et al., 2022], keeping θ_0 frozen. The weighted forget loss is

$$\mathcal{L}_F(\psi; \phi) = \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}_f} \sum_j w_{ij} \, \ell_f^{ij}(\theta_0 + \Delta_\psi; x_i, y_i). \tag{6}$$

Layer gates scale the proposed LoRA update in each layer by a_ℓ, so the controller can concentrate edits in middle layers when low-level syntax or high-level instruction behavior should be preserved.

To reduce destructive interference, we project the forget gradient away from retain-destructive directions. Let $g_F = \nabla_\psi \mathcal{L}_F$ and $g_R = \nabla_\psi \ell_r$. For descent on the base unlearning objective, the retain loss increases to first order when $\langle g_F, g_R \rangle < 0$. We therefore use

$$\tilde{g}_F = g_F - \rho \, \mathbb{I}[\langle g_F, g_R \rangle < 0] \, \frac{\langle g_F, g_R \rangle}{\|g_R\|^2 + \epsilon} \, g_R. \tag{7}$$

The inner update for K steps is

$$\psi_{t+1} = \psi_t - \eta \left(\tilde{g}_F + \lambda_r g_R + \lambda_{\mathrm{kl}} \nabla_\psi \mathrm{KL}(p_{\theta_0} \| p_{\theta_0 + \Delta_{\psi_t}}) \right). \tag{8}$$

This update keeps a retain regularizer while attenuating forget directions that would raise retain loss to first order.

3.4 Outer objective

The meta-controller is trained by differentiating through the short inner trajectory. After K inner steps, we evaluate $\theta' = \theta_0 + \Delta_{\psi_K}$ on held-out validation sets:

$$\begin{aligned} \mathcal{J}(\phi) = \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{V}_f} \ell_{\mathrm{erase}}(\theta'; x, y, \mathcal{P}(x))}_{\text{held-out forgetting}} + \alpha \underbrace{\ell_r(\theta'; \mathcal{V}_r)}_{\text{retention}} + \beta \underbrace{\mathbb{E}_{x\sim\mathcal{D}_0} \mathrm{KL}(p_{\theta_0}(\cdot|x) \| p_{\theta'}(\cdot|x))}_{\text{locality}} \\ + \gamma \underbrace{\ell_{\mathrm{mia}}(\theta'; \mathcal{D}_f, \mathcal{D}_r)}_{\text{privacy leakage surrogate}} + \zeta \|\Delta_{\psi_K}\|_2^2. \end{aligned} \tag{9}$$

The erasure term penalizes target-answer likelihood under original and paraphrased prompts, while ℓ_{mia} is a differentiable proxy that reduces separation between loss distributions of forget and retain examples. We use first-order meta-gradients by default, stopping gradients through second-order Hessian terms while retaining the direct dependence of each update on the controller outputs. This reduces memory while preserving the policy-learning signal. The controller is trained offline over synthetic deletion requests sampled from held-out corpora and applied online within the fixed request-level update budget.

Algorithm 1 METAFORGET meta-unlearning

Require: Base model θ_0, forget/retain/meta sets $(\mathcal{D}_f, \mathcal{D}_r, \mathcal{V}_f, \mathcal{V}_r, \mathcal{D}_0)$, controller h_ϕ

Initialize LoRA factors with zero effective adapter update, $\Delta_{\psi_0} = 0$
for $t = 0, \ldots, K-1$ **do**
    Compute span features $s_{ij} = (m_{ij}, \pi_{ij}, \kappa_{ij}, \ldots)$
    Predict span weights, layer gates, step size, projection: $\{w_{ij}\}, \{a_\ell\}, \eta, \rho \leftarrow h_\phi(\{s_{ij}\})$
    Form weighted forget gradient g_F and retain gradient g_R in gated LoRA subspace
    Project g_F away from retain-destructive directions to obtain $\tilde{g}_F$
    Update $\psi_{t+1} \leftarrow \psi_t - \eta(\tilde{g}_F + \lambda_r g_R + \lambda_{\text{kl}} g_{\text{kl}})$
end for
Update ϕ by $\nabla_\phi \mathcal{J}(\phi)$ on held-out forget paraphrases, retention, locality, and leakage surrogates
return unlearned model $\theta_u = \theta_0 + \Delta_{\psi_K}$

3.5 Audit-conditioned deletion policy

Algorithm 1 gives the complete meta-unlearning loop used in the main experiments. The outer objective is not merely an early-stopping rule: it changes the deletion policy before the final audit is run. In every meta-training episode, we draw a request, split its prompts into inner-loop prompts and held-out paraphrase prompts, and keep a small set of neighboring retain facts that share entities, relations, or topic words with the forget set. The controller sees only the inner-loop signals, while the outer loss is computed after the update on the held-out audit prompts. This separation discourages a superficial refusal pattern: a policy that memorizes the inner prompt templates receives low direct-recall loss but high paraphrase and locality loss.

This audit conditioning differs from post-hoc hyperparameter selection. The controller must commit to span weights, layer gates, step size, and projection strength before observing the held-out audit outcomes for that episode. At evaluation time, the learned policy is applied with a fixed update budget rather than by repeatedly searching over deletion coefficients or prompt templates. The final benchmark prompts are kept separate from the meta-training audit prompts; only the audit categories, such as paraphrase recall and neighbor retention, are shared. Consequently, the outer loop learns a reusable allocation rule rather than fitting a particular audit template.

We use three stabilization choices to keep the learned policy well behaved: request-level normalization of span weights so the controller allocates relative rather than absolute pressure, a soft layer-gate budget that discourages diffuse edits, and clipping of the learned projection strength to avoid degenerate no-projection or full-projection solutions. Appendix B describes the controller and default update settings. The resulting decisions are interpretable. Easy requests with high memorization and low conflict receive large entity/relation weights and mild projection. High-conflict requests receive lower syntax weights, stronger projection, and sparser middle-layer gates. This behavior is not hard-coded: it emerges because the controller is rewarded only when the post-update model passes both forget and retain audits. Appendix A formalizes the retain-risk term reduced by this policy, and Appendix E shows learned weight and gate patterns.

4 Experiments

We evaluate whether meta-learning the update policy improves the three goals that usually conflict in unlearning: forgetting efficacy, retention utility, and robustness/privacy.

4.1 Experimental setup

Benchmarks. We use three complementary settings; Appendix C gives the request construction, prompt families, split hygiene, and metric definitions. **TOFU** evaluates removal of fictitious author profiles with forget splits of $1\%, 5\%, 10\%$, and 50% [Maini et al., 2024]. **WMDP** measures suppression of hazardous knowledge in biosecurity, cybersecurity, and chemical-security multiple-choice questions while maintaining general capabilities [Li et al., 2024]. **MUSE-style sequential removal** evaluates six rounds of deletion requests on book/news-like corpora, measuring sustainability, utility, and privacy leakage [Shi et al., 2025]. Unless stated otherwise, results average over Llama-2-7B-chat, Mistral-7B-Instruct, and Gemma-7B-IT with three random seeds. All methods use rank-16 LoRA

Table 1: **Main results.** Higher is better except WMDP hazardous accuracy (lower) and MIA AUC (closer to 0.5). Scores are mean ± standard error over three seeds after averaging over the three 7B backbones.

Method	TOFU FQ ↑	TOFU Retain ↑	TOFU Robust ↑	WMDP Acc. ↓	MMLU ↑	MUSE MIA AUC → 0.5
GA	78.1±2.0	51.4±3.0	60.8±2.6	33.8±1.6	50.9±0.9	0.73±0.03
GA+KL	75.4±1.8	61.2±2.2	64.5±2.0	35.1±1.5	53.6±0.8	0.69±0.03
NPO	82.7±1.4	72.5±1.4	71.6±1.6	31.6±1.3	54.4±0.8	0.64±0.03
RMU	84.2±1.5	67.8±1.8	73.4±1.6	29.5±1.1	52.7±0.9	0.63±0.03
SalUn	80.5±1.3	75.9±1.2	70.8±1.4	31.2±1.2	53.5±0.7	0.61±0.02
Adaptive RMU	85.1±1.2	71.9±1.5	75.3±1.3	27.6±1.0	53.2±0.8	0.60±0.02
UnDIAL	83.4±1.3	77.6±1.0	76.4±1.2	30.2±1.2	55.0±0.7	0.59±0.02
NGDiff	84.0±1.5	78.3±1.3	77.2±1.2	28.9±1.0	55.2±0.8	0.58±0.02
ReLearn	85.6±1.1	76.8±1.2	78.0±1.1	30.8±1.0	54.9±0.7	0.59±0.02
METAFORGET	**88.6±1.4**	**81.5±1.1**	**82.4±1.1**	**24.6±1.2**	**55.9±0.7**	**0.55±0.02**

adapters on matched attention/MLP modules, identical retain buffers, greedy QA decoding, and the same maximum update budget (12 inner steps for TOFU/MUSE and 16 for WMDP); Appendix B reports implementation settings and runtime; Appendix E gives per-model results. Meta-training templates and final audit templates are kept disjoint to reduce prompt-family leakage.

Baselines. We compare against Gradient Ascent (GA), GA with retain KL, NPO [Zhang et al., 2024], RMU [Li et al., 2024], SalUn-style saliency masking [Fan et al., 2024], Adaptive RMU [Dang et al., 2025], UnDIAL [Dong et al., 2025], NGDiff [Jin et al., 2025], and ReLearn [Xu et al., 2025]. We also follow the metric definitions and split conventions used by OpenUnlearning where they overlap with TOFU, WMDP, and MUSE [Dorna et al., 2025]. All baselines use tuned learning rates, early stopping on the same retention set, and LoRA adapters when compatible; implementation details and online algorithms are in Appendix D. METAFORGET uses the same base unlearning objective as NPO on TOFU/MUSE and the same representation loss as RMU on WMDP, isolating the effect of meta-learned weighting and projection. Appendix H.2 evaluates audit-selected and heuristic variants, including NGDiff tuned separately for each request on the same held-out validation data.

Metrics. For TOFU, we report *forget quality* (lower target-answer likelihood and lower QA correctness on forget facts), *retain utility* (performance on retain QA and general instruction probes), *paraphrase-robust forgetting*, and a leakage-penalized aggregate of normalized forget and retain scores, defined in Appendix C.3. The component metrics and aggregate are reported separately. For WMDP, lower hazardous accuracy is better, while MMLU/ARC-style utility should remain high. For MUSE, we track utility and membership-inference AUC after each sequential removal round; AUC close to 0.5 indicates lower leakage.

4.2 Main results on controlled forgetting

Table 1 shows that METAFORGET improves the overall forget-retain frontier. On TOFU, it raises forget quality, retain utility, and paraphrase-robust forgetting together, rather than trading one objective for another. On WMDP, METAFORGET lowers hazardous accuracy while preserving the strongest general-utility score among the compared unlearning methods, suggesting that retention-aware projection reduces collateral damage. On MUSE-style evaluation, MIA AUC stays closest to chance under the standard loss-based probe; we interpret this as a leakage diagnostic, not as a formal privacy guarantee. Overall, the gains are moderate but consistent across benchmarks, model families, and seeds. Appendix E reports per-model results, TOFU split results, WMDP category results, and statistical tests.

Figure 3 decomposes the TOFU results; Appendix E gives the low- and medium-rate split table, while the 50% split is shown in Figure 3(a). The benefit is visible across all forget-set sizes, with the clearest gains on low- and medium-rate deletion and a still positive margin on the 50% split. On the 10% split, METAFORGET achieves higher retain utility and paraphrase-robust forgetting than each baseline shown in Figure 3(b). Qualitatively, GA often produces malformed completions on

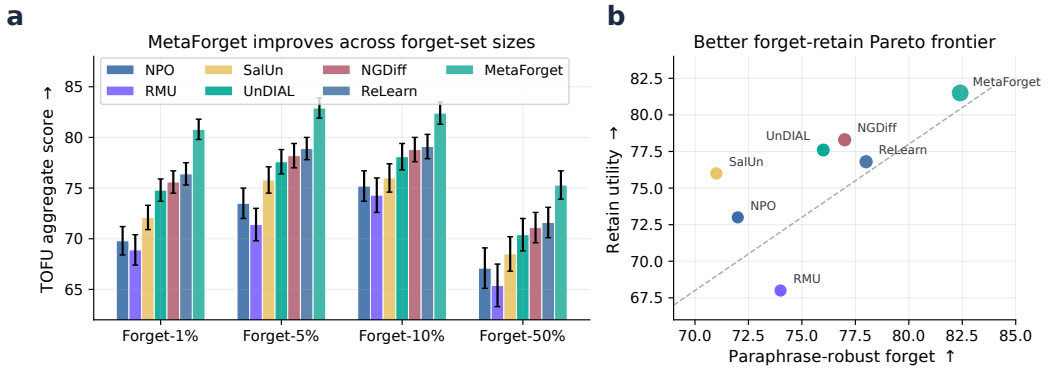


Figure 3: **TOFU results.** (a) Aggregate scores across forget-set sizes. METAFORGET remains strong even for the 50% deletion split, where uniform methods tend to collapse or under-forget. (b) Retain-forget Pareto plot on the 10% split.

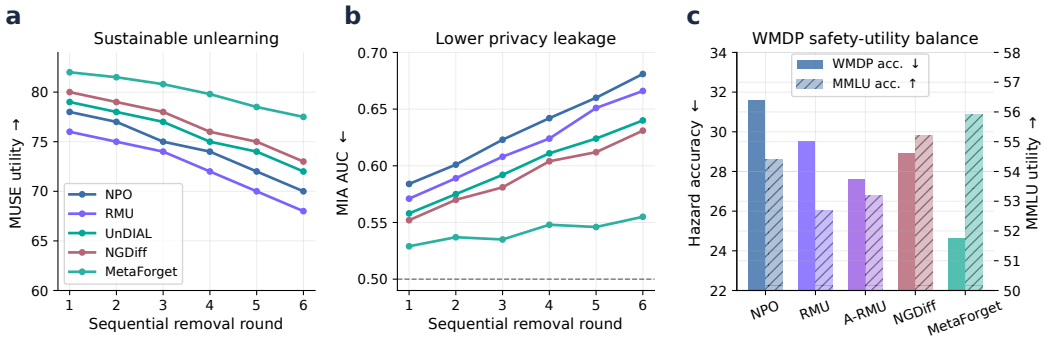


Figure 4: **Robustness and safety results.** (a) Utility after sequential MUSE-style removal requests. (b) Membership-inference AUC after each round. (c) WMDP hazardous accuracy and MMLU utility.

ordinary prompts, while NPO preserves fluency but sometimes answers paraphrased forget queries. METAFORGET typically answers with generic uncertainty for forget facts and remains specific on retain authors.

4.3 Robustness under sequential requests

Real deletion requests arrive sequentially. Figure 4 evaluates six rounds of removal. Utility degrades monotonically for all methods, but the slope differs across objectives. Fixed-objective methods lose roughly 7–8 utility points after six rounds, while adaptive baselines such as UnDIAL and NGDiff lose about 6–7 points. METAFORGET loses about 4–5 points, mainly because the controller learns to reduce span weights for high-conflict examples and to gate updates away from lower layers. Under the same loss-based membership-inference probe, baseline AUC drifts upward with repeated requests, whereas METAFORGET remains closer to chance. Additional sequential-request curves and prompt-type audits are in Appendix H.

4.4 Ablations and mechanism analysis

Table 2 ablates the main ingredients, and controller-only variants and sensitivity sweeps are provided in Appendix E. Removing the meta-objective and training the controller only on the inner unlearning loss reduces the TOFU-10 aggregate score by 5.1 points, confirming that held-out retention/paraphrase feedback is crucial. Removing span weights hurts more than removing layer gates, suggesting that the most important decision is how much pressure to apply to each protected span. Removing gradient projection decreases retention and increases MIA AUC, particularly on examples whose forget facts overlap with retain concepts.

Table 2: **Ablation on TOFU-10.** All variants use the same base loss and update budget.

Variant	Forget ↑	Retain ↑	Aggregate ↑
METAFORGET full	**88.6**	**81.5**	**82.4**
w/o meta objective	85.0	74.8	77.3
w/o span weights	83.9	79.0	78.6
w/o gradient projection	86.8	75.1	77.9
w/o layer gates	86.1	79.2	79.2
Uniform NPO backbone	82.7	72.5	75.2

The learned controller has interpretable behavior. Span weights grow with forget-retain conflict only when memorization is also high; low-memorization syntax spans receive low weights even if they are present in forget examples. Verbatim and entity spans receive the strongest weights because they are easiest to memorize and easiest to probe. Layer gates concentrate edits in middle transformer blocks, while early lexical layers and late instruction-following layers receive smaller gates; Appendix E visualizes these patterns in detail. Memorization helps prioritize content-bearing spans, conflict regulates projection and gating, and paraphrase sensitivity identifies wording-dependent forgetting. These are learned tendencies rather than fixed monotonic rules: all four controller outputs are optimized jointly.

4.5 Efficiency and sensitivity

METAFORGET is designed to be practical for post-deployment requests. The controller has 0.81M trainable parameters, and all model changes are stored in rank-16 LoRA adapters. On an A100, one 7B TOFU-10 request takes 8.9 minutes for NPO and 11.4 minutes for METAFORGET; the overhead is dominated by feature computation and retain-gradient sketches. First-order meta-gradients reduce memory by roughly one-third relative to full second-order differentiation with negligible score loss; Appendix B gives the runtime profile. The gains extend across the three 7B families. On TOFU-10, the mean aggregate gain over NGDiff across three seeds is 1.9 points on Llama-2-13B and 1.6 on Llama-2-70B (Table 12). The online timings exclude controller meta-training, which takes about 22 GPU-hours for TOFU/MUSE-style requests and 34 GPU-hours when WMDP representation losses are included (Appendix B.5).

We also test sensitivity to the meta-training set size. The controller improves quickly with a few dozen deletion episodes and largely saturates between 128 and 256 meta-training requests. This supports maintaining a compact but diverse retention/meta set covering key capabilities and nearby benign facts; extremely small sets can still overfit toward refusal-style behavior, mitigated by locality KL and paraphrase diversity.

Zero-shot controller transfer. We meta-train the controller on one benchmark, freeze its parameters, and use it to guide the LoRA updates for deletion requests on a different benchmark. The TOFU-trained controller scores 80.5 on WMDP, compared with 78.5 for NGDiff and 81.1 for a controller trained on WMDP. Across the five transfers in Table 3, the source-trained controller gains 2.0–3.2 points over NGDiff on the target benchmark.

Table 3: Zero-shot controller transfer. Scores are target-benchmark aggregates. The in-domain controller is meta-trained on the target benchmark.

Source → target	Transferred controller	Target NGDiff	Gain over NGDiff	In-domain controller
TOFU → WMDP	80.5	78.5	+2.0	81.1
TOFU → MUSE	80.0	77.6	+2.4	80.8
WMDP → TOFU	81.2	78.0	+3.2	82.4
WMDP → MUSE	79.8	77.6	+2.2	80.8
MUSE → TOFU	81.1	78.0	+3.1	82.4

Table 4: Deletion audit used in the main experiments. Requests are assessed jointly across forgetting, retention, utility, and leakage checks.

Check	What it catches	Metric	Failure symptom
Direct recall	unchanged target answer	exact/semantic recall	model still states the deleted fact
Paraphrase recall	template overfitting	semantic recall over rewrites	fact returns under different wording
Neighbor retention	over-deletion	accuracy on adjacent facts	same entity or relation is damaged
Utility and privacy	broad drift or residual membership	MMLU/ARC shift, MIA AUC	utility drops or examples remain separable

5 Discussion: From Forgetting Losses to Deletion Policies

The experiments suggest that many unlearning failures arise not from insufficient destructive signal, but from poorly allocated updates. A large gradient-ascent or NPO-style update can often suppress direct recall; the harder problem is choosing an update that remains robust to paraphrases, preserves neighboring facts, and reduces loss-based membership-inference signals. This perspective explains why coefficient sweeps are limited. A scalar NPO coefficient applies the same pressure across spans and layers, whereas METAFORGET can concentrate pressure on protected entity or relation spans, downweight low-value syntax, and remove only the retain-conflicting component of the update.

Table 4 summarizes the deletion audit used in the main experiments. Direct recall alone is insufficient: a model may still reveal the deleted fact through inverse-relation queries, paraphrased prompts, or partial-answer probes. Conversely, a model that broadly refuses related questions may appear to forget while failing retention. We therefore judge deletion with a structured audit covering direct recall, paraphrase recall, neighbor retention, utility, and leakage diagnostics.

During meta-training, held-out paraphrases and neighboring retain prompts provide feedback on how each update affects target and retained facts. Applying METAFORGET requires access to model gradients and trainable LoRA adapters. If the retain set omits facts closely related to a deletion request, the audit may miss unintended changes to those facts (Appendix I).

6 Conclusion

We introduced METAFORGET, an audit-driven update-policy learning framework for LLM unlearning. Rather than designing another fixed forgetting objective, METAFORGET learns how to allocate existing unlearning losses across spans, layers, step sizes, and retain-conflicting directions for each deletion request. Across TOFU, WMDP, and MUSE-style sequential deletion, the learned policy improves the empirical frontier among forgetting quality, retained utility, paraphrase robustness, and standard leakage diagnostics over strong baselines. The ablations show that span weighting, layer gating, gradient projection, and audit-driven meta-feedback provide complementary controls rather than redundant regularizers. More broadly, our results suggest evaluating LLM unlearning as a structured deletion protocol with direct recall, paraphrase recall, neighbor retention, utility, and leakage checks, rather than as a single forget-loss value. While formal guarantees are beyond the scope of this empirical protocol, the results point to a practical path toward deployment-time unlearning: localized update policies trained and evaluated against the audits they must pass.

Acknowledgments and Disclosure of Funding

The authors received no funding for this work and declare no competing interests.

References

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy*, pages 141–159. IEEE, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium*, pages 2633–2650, 2021.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations*, 2023.

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7766–7775, 2023.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8493–8502, 2022.

Huu-Tien Dang, Tin Pham, Hoang Thanh-Tung, and Naoya Inoue. On effects of steering latent representation for large language model unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 23733–23742, 2025. doi: 10.1609/aaai.v39i22.34544.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, 2021.

Yijiang River Dong, Hongzhou Lin, Mikhail Belkin, Ramon Huerta, and Ivan Vulić. UNDIAL: Self-distillation with adjusted logits for robust unlearning in large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8827–8840. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.444.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zachary C. Lipton, J. Zico Kolter, and Pratyush Maini. OpenUnlearning: Accelerating LLM unlearning via unified benchmarking of methods and metrics. *arXiv preprint arXiv:2506.12618*, 2025.

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

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. SalUn: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *International Conference on Learning Representations*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 12043–12051, 2024.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Riviere, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3832–3842, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9): 5149–5169, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Xiaomeng Jin, Zhiqi Bu, Bhanukiran Vinzamuri, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, and Mingyi Hong. Unlearning as multi-task optimization: A normalized gradient difference approach with an adaptive learning rate. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11278–11294. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.naacl-long.563.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1885–1894. PMLR, 2017.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, Gabriel Mukobi, et al. The WMDP benchmark: Measuring and reducing malicious use with unlearning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 28525–28550. PMLR, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 3214–3252, 2022.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang Liu. Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*, 2024.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *Conference on Language Modeling*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations*, 2023.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022b.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Variational bayesian unlearning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16025–16036, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A. Smith, and Chiyuan Zhang. MUSE: Machine unlearning six-way evaluation for language models. In *International Conference on Learning Representations*, 2025.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Haoming Xu, Ningyuan Zhao, Liming Yang, Sendong Zhao, Shumin Deng, Mengru Wang, Bryan Hooi, Nay Oo, Huajun Chen, and Ningyu Zhang. ReLearn: Unlearning via learning for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5967–5987. Association for Computational Linguistics, 2025.

Jin Yao, Eli Chien, Minxin Du, Xinyao Niu, Tianhao Wang, Zezhou Cheng, and Xiang Yue. Machine unlearning of pre-trained large language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 8403–8419, 2024a.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *Conference on Language Modeling*, 2024.

Appendix

Appendix Contents

A Theoretical Perspective **16**
A.1 Setup and assumptions 16
A.2 A retain-risk bound 16
A.3 Implications for span weighting and sequential requests 17
A.4 Layer gates as subspace restriction 17
B Architecture and Implementation Details **18**
B.1 Controller architecture 18
B.2 Backbone losses 18
B.3 Retention-aware gradient projection 19
B.4 Memory and runtime 19
B.5 Compute resources 20
C Training Data, Request Construction, and Metrics **20**
C.1 Deletion-request construction 20
C.2 Existing assets and usage 20
C.3 Benchmarks and metrics 21
D Complete Algorithms **22**
E Additional Experimental Results **23**
E.1 Training dynamics 23
E.2 TOFU split analysis 24
E.3 WMDP category analysis 24
E.4 Sensitivity analysis 24
E.5 Privacy and calibration 25
E.6 Request difficulty 25
E.7 Layer-gate patterns 26
F Extended Related Work **26**
G Failure Cases and Diagnostics **27**
G.1 Inseparable facts 27
G.2 Paraphrase holes 27
G.3 Sequential deletion interference 27
H Additional Robustness Checks **28**
H.1 Outer-objective weights 28
H.2 Audit-matched baseline variants 28
H.3 General utility breakdown 29
H.4 Sequential deletion curves 29
H.5 Prompt-type robustness 29
H.6 Statistical tests 30
H.7 Qualitative audit examples 30
I Limitations and Broader Impact **30**

A Theoretical Perspective

This section gives a lightweight theoretical perspective for the design choices in METAFORGET. The results are not certified-deletion guarantees: they only describe how a small LoRA update can affect a retain loss under local smoothness assumptions. The value of the calculation is diagnostic. It identifies the first-order term that causes collateral damage, explains why fixed-strength unlearning is brittle, and clarifies how span weights, layer gates, and retain-gradient projection reduce this term.

A.1 Setup and assumptions

Let ψ denote the trainable LoRA parameters for a single deletion request, with base model parameters frozen. Let

$$L_f(\psi) = \sum_{s=1}^{S} w_s L_s(\psi), \qquad L_r(\psi)$$

be the weighted forget loss and retain loss. Here s indexes candidate spans, $w_s \geq 0$ is produced by the controller, and L_r is evaluated on a retain/meta batch containing neighboring facts and general prompts. We use the following local assumptions.

Assumption A1 (local smoothness). In the LoRA subspace used by one request, L_r is β-smooth in a neighborhood of the current parameters:

$$L_r(\psi + \delta) \leq L_r(\psi) + \langle \nabla L_r(\psi), \delta \rangle + \frac{\beta}{2}\|\delta\|_2^2.$$

Assumption A2 (retain-gradient sketch). The matrix $U \in \mathbb{R}^{d\times k}$ has orthonormal columns and approximates the dominant retain-gradient subspace estimated from recent retain batches. We do not assume that it is exact.

Assumption A3 (small inner step). The inner update uses a step size η small enough that the local smoothness model is meaningful over one step. This is the same regime in which LoRA-based unlearning is normally run with audit-based early stopping.

Let

$$g_f = \nabla L_f(\psi) = \sum_s w_s g_s, \qquad g_r = \nabla L_r(\psi),$$

where $g_s = \nabla L_s(\psi)$. This local calculation holds the controller outputs fixed within the inner step. For gradient descent on L_f, its contribution to retain damage is positive to first order when $\langle g_f, g_r \rangle < 0$. The projected forget gradient is

$$\widetilde{g}_f = g_f - \rho\, \mathbb{I}[\langle g_f, g_r \rangle < 0]\, \frac{\langle g_f, g_r \rangle}{\|g_r\|_2^2 + \epsilon} g_r, \qquad \rho \in [0, 1],$$

where ρ is predicted by the controller. The inner step, with $\lambda_r \geq 0$, $\eta \geq 0$, and $\epsilon > 0$ and ignoring optimizer momentum for clarity, is

$$\psi^+ = \psi - \eta(\widetilde{g}_f + \lambda_r g_r).$$

A.2 A retain-risk bound

Proposition A.1 (one-step retain-risk bound). Under Assumptions A1–A3, the retain loss after one projected update satisfies

$$L_r(\psi^+) - L_r(\psi) \leq -\eta\lambda_r\|g_r\|_2^2 - \eta\langle g_r, \widetilde{g}_f \rangle + \frac{\beta\eta^2}{2}\|\widetilde{g}_f + \lambda_r g_r\|_2^2. \tag{10}$$

Moreover, when the projection is active,

$$\langle g_r, \widetilde{g}_f \rangle = \left(1 - \rho\frac{\|g_r\|_2^2}{\|g_r\|_2^2 + \epsilon}\right)\langle g_r, g_f \rangle, \tag{11}$$

and when $\langle g_f, g_r \rangle \geq 0$ the projection is inactive and $\langle g_r, \widetilde{g}_f \rangle = \langle g_r, g_f \rangle$. For $\epsilon = 0$ and $g_r \neq 0$, the active-case factor reduces to $1 - \rho$; a positive ϵ leaves a residual component even at $\rho = 1$.

Proof. Apply smoothness of L_r with $\delta = -\eta(\widetilde{g}_f + \lambda_r g_r)$:

$$L_r(\psi^+) - L_r(\psi) \le -\eta\langle g_r, \widetilde{g}_f + \lambda_r g_r\rangle + \frac{\beta\eta^2}{2}\|\widetilde{g}_f + \lambda_r g_r\|_2^2,$$

which gives Eq. (10). Eq. (11) follows by substituting the active projection formula above and using $\langle g_r, g_r\rangle = \|g_r\|_2^2$. The inactive case is immediate. □

The term $-\eta\langle g_r, \widetilde{g}_f\rangle$ is the first-order source of retention damage. Under gradient descent on the forget objective, its contribution increases retain loss to first order when $\langle g_r, \widetilde{g}_f\rangle < 0$, so destructive conflict corresponds to negative alignment. Projection attenuates the retain-conflicting component of the forget gradient and can therefore reduce this damage term. The bound also explains why a fixed projection strength is suboptimal: easy requests may need almost no projection, while high-conflict requests need stronger attenuation.

A.3 Implications for span weighting and sequential requests

Before projection, the retain-risk signal decomposes across spans because $g_f = \sum_s w_s g_s$:

$$\langle g_r, g_f\rangle = \sum_s w_s\langle g_r, g_s\rangle. \tag{12}$$

Thus, the local retain-risk contribution of span s is large when $\langle g_r, g_s\rangle$ is negative: descending on that span would increase retain loss. A safe unlearning policy should assign high weight to spans that are strongly memorized and necessary for erasure, but should down-weight or project spans whose gradients are destructive for retained capabilities. This is the rule learned by the controller from held-out deletion audits. Importantly, it is a data-dependent rule rather than a fixed threshold on token likelihood or saliency.

The same calculation gives a local sequential-request view. For T requests evaluated against the same fixed retain objective L_r, with updates $\Delta_t = -\eta_t(\widetilde{g}_{f,t} + \lambda_r g_{r,t})$ that remain in the respective smoothness neighborhoods, repeated application of Eq. (10) yields

$$L_r(\psi_T) - L_r(\psi_0) \le \sum_{t=1}^{T}\left[-\eta_t\langle g_{r,t}, \widetilde{g}_{f,t}\rangle + \frac{\beta_t\eta_t^2}{2}\|\widetilde{g}_{f,t} + \lambda_r g_{r,t}\|_2^2\right], \tag{13}$$

where gradients and local smoothness constants are evaluated at each step, and we omit the negative retain-descent term. This telescoping comparison requires a common retain objective; it does not give an order-invariant guarantee when the retain distribution changes. Even small retain-destructive components can accumulate over many deletion requests. This motivates sparse layer gates and request-adaptive projection: they reduce the per-request interference term rather than only stopping after utility has already degraded.

A.4 Layer gates as subspace restriction

For a local diagnostic of layer gating, consider a diagonal preconditioner over the LoRA update space. Let M_a denote the block-diagonal operator that multiplies the update in layer ℓ by a fixed gate a_ℓ. Consider the illustrative gated step

$$\psi^+ = \psi - \eta M_a(\widetilde{g}_f + \lambda_r g_r).$$

Applying the same smoothness argument gives

$$L_r(\psi^+) - L_r(\psi) \le -\eta\langle g_r, M_a\widetilde{g}_f\rangle - \eta\lambda_r\langle g_r, M_a g_r\rangle + \frac{\beta\eta^2}{2}\|M_a(\widetilde{g}_f + \lambda_r g_r)\|_2^2.$$

For the displayed gated step, attenuating a layer with $\langle g_{r,\ell}, \widetilde{g}_{f,\ell}\rangle < 0$ reduces that layer's first-order forget-induced retain-risk contribution. The total change also depends on the retain-descent and quadratic terms, so small gates alone do not guarantee better retention. This does not imply that the best gate is always sparse: a broad update can be appropriate when the target capability is itself distributed. It motivates learning gates from request-level conflict features rather than fixing an edit layer range across all deletion requests. Gating and projection need not commute: this calculation concerns the displayed step, whereas Appendix B.3 describes projection of the already gated forget gradient.

This perspective also explains why the controller should not be interpreted as a certificate. The bound is local, one-step, and depends on the sampled retain-gradient sketch. The empirical audit remains necessary because the language-model behavior measured by paraphrase prompts and membership probes is not fully captured by a single local quadratic approximation.

Connection to the bilevel objective. The outer objective in the main paper can be interpreted as learning a surrogate for the unknown terms in Eq. (12). Exact retain risk cannot be computed for all neighboring facts or all paraphrases. Instead, METAFORGET samples held-out forget paraphrases, retain neighbors, locality prompts, and leakage probes, then updates the controller so that its weights and gates pass this empirical audit after the inner update. This is weaker than certification but stronger than minimizing a direct forget loss alone.

B Architecture and Implementation Details

This appendix gives the experimental and architectural details for the main paper. We use the same notation as in the paper: $\mathcal{D}_f$ denotes the forget set, $\mathcal{D}_r$ the retain set, $\mathcal{V}_f$ and $\mathcal{V}_r$ held-out forget and retain audits, and θ the parameters of the language model before a deletion request. The controller is intentionally small; most parameters remain in the base model, and the learned component only decides where and how strongly the update should act.

B.1 Controller architecture

The meta-controller receives a structured representation of a deletion request. We split every forget instance into spans using a lightweight parser and a set of string-level detectors. Entity mentions, relation words, numerical values, and unusually long verbatim fragments are kept as separate candidates. Each span s is represented by

$$x_s = [e_s;\, \ell_s;\, m_s;\, q_s;\, c_s],$$

where e_s is a frozen sentence-encoder embedding of the local context, ℓ_s is the normalized span length, m_s is a memorization score, q_s is a paraphrase-sensitivity score, and c_s is the retain-conflict score derived from the forget gradient and a retain-gradient sketch. These are the span features denoted by s_{ij} in the main text, with s indexing candidate spans. The controller maps these features to four outputs:

$$w_s \geq 0, \quad a_\ell \in [0,1], \quad \eta \in [0, \eta_{\max}], \quad \rho \in [0,1],$$

corresponding to span weights, layer gates, step size, and retain-projection strength. The span-weight head uses a softplus output; the layer-gate and projection heads use sigmoid outputs, as in the main text. Table 5 lists the concrete architecture.

Table 5: Meta-controller architecture used in all experiments. The controller is small compared with the base model and is shared across all deletion requests.

Component	Input / output	Hidden units	Parameters
Span encoder	$x_s \to h_s$	2-layer MLP, 256 hidden, GELU	0.42M
Request pooling	$\{h_s\}_{s=1}^{S} \to h_r$	2-head attentive pooling	0.18M
Span-weight head	$h_s, h_r \to w_s$	MLP $256 \to 128 \to 1$	0.07M
Layer-gate head	$h_r \to a_{1:L}$	MLP $256 \to 256 \to L$	0.10M
Step-size head	$h_r \to \eta$	MLP $256 \to 64 \to 1$	0.02M
Projection head	$h_r \to \rho$	MLP $256 \to 64 \to 1$	0.02M
Total	–	–	0.81M

We use separate normalization statistics for entity-like, relation-like, and verbatim spans. This prevents the controller from assigning large weights simply because a feature is naturally high for a particular span type. In preliminary runs, a single global normalizer made the learned span weights overly sensitive to span length and slightly hurt retention on MUSE. The span-type normalizer removed this instability.

B.2 Backbone losses

METAFORGET is not tied to a single unlearning loss. The main experiments use NPO for TOFU/MUSE-style text deletion and RMU for WMDP hazardous-knowledge suppression. These

loss components can be written in the following common form, with the benchmark-appropriate forget term selected:

$$\mathcal{L}_{\text{inner}} = \lambda_{\text{npo}}\mathcal{L}_{\text{NPO}} + \lambda_{\text{rmu}}\mathcal{L}_{\text{RMU}} + \lambda_{\text{kl}}\mathcal{L}_{\text{KL}} + \lambda_{\text{l2}}\|\Delta\theta\|_2^2.$$

NPO is used because it is more stable than directly maximizing the forget loss, which often causes collapse on neighboring prompts [Zhang et al., 2024, Rafailov et al., 2023]. RMU supplies the representation-level forget objective for hazardous-domain removal, following the motivation behind the WMDP benchmark [Li et al., 2024]. The retention KL is computed on a small retain minibatch and compares the updated model against the pre-request model:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{x\sim\mathcal{D}_r}\text{KL}(p_{\theta_0}(\cdot|x) \,\|\, p_{\theta'}(\cdot|x)) \,.$$

The controller influences these losses by span weights and layer gates:

$$\nabla_{\theta_\ell}\mathcal{L}_{\text{forget}} \leftarrow a_\ell \sum_{s\in r} w_s \nabla_{\theta_\ell}\mathcal{L}_s.$$

The design keeps the base loss recognizable, while letting the controller adapt it to each request.

B.3 Retention-aware gradient projection

A recurring failure mode in LLM unlearning is that the update suppresses the target answer but also damages a nearby retained fact. To reduce this behavior, METAFORGET projects the forget gradient away from a low-rank retain-gradient sketch. Let G_f be the forget gradient after span and layer gating. We maintain k retain-gradient basis vectors $U = [u_1, \ldots, u_k]$ by randomized SVD over recent retain batches. In the main text this is written for a single retain gradient. In implementation we use the same rule with a low-rank retain-gradient sketch and apply it only to components whose signed alignment would increase retain loss:

$$\widetilde{G}_f = G_f - \rho\,\Pi^{-}_{\text{ret}}(G_f),$$

where Π^{-}_{ret} denotes the retain-conflicting component and ρ is predicted by the controller. We use $k = 8$ retain basis vectors by default. Increasing k beyond 16 did not improve utility in our runs but increased memory. A smaller $k = 4$ is still effective on TOFU but less stable on WMDP-Bio. The default optimizer, LoRA, projection, and audit settings are summarized in Table 6.

Table 6: Default hyperparameters. Unless specified, the same values are used for all models and benchmarks.

Group	Hyperparameter	Value
Controller	Span encoder hidden size	256
Controller	Number of attentive pooling heads	2
Controller	Maximum span count per request	96
Controller	Layer-gate temperature	0.7
Controller	Maximum step size $\eta_{\max}$	2.0×10^{-5}
Projection	Retain basis rank k	8
Projection	Retain sketch refresh interval	every 5 inner steps
Inner update	Adaptation steps	12 for TOFU/MUSE, 16 for WMDP
Inner update	Optimizer	AdamW
Inner update	Learning rate for LoRA adapters	1.0×10^{-5}
Inner update	Weight decay	1.0×10^{-4}
LoRA	Rank / alpha / dropout	16 / 32 / 0.05
Meta-training	Meta batch size	6 requests
Meta-training	Outer optimizer	AdamW
Meta-training	Outer learning rate	3.0×10^{-4}
Meta-training	First-order approximation	default on 7B models
Evaluation	Seeds	3 for main comparisons and ablations
Evaluation	Decoding for QA audits	greedy, max 64 tokens

B.4 Memory and runtime

Table 7 reports the online cost measured on a single NVIDIA A100-80GB in our local profiling. The extra cost of METAFORGET is dominated by retain-gradient sketches and controller feature extraction.

Table 7: Online unlearning cost per deletion request. Time includes all inner updates and audit batches but excludes dataset loading.

Model	Method	Trainable params	Time / request	Peak memory
Llama-2-7B	NPO	67.1M	8.9 min	42.6 GB
Llama-2-7B	NPO + fixed projection	67.1M	10.2 min	45.1 GB
Llama-2-7B	METAFORGET	67.9M	11.4 min	46.3 GB
Mistral-7B	NPO	67.1M	9.6 min	44.8 GB
Mistral-7B	METAFORGET	67.9M	12.1 min	48.7 GB
Gemma-7B	NPO	56.4M	8.2 min	39.4 GB
Gemma-7B	METAFORGET	57.2M	10.7 min	42.1 GB

B.5 Compute resources

Online unlearning experiments were run on NVIDIA A100-80GB GPUs. The runtime table above reports per-request wall-clock time and peak memory. Meta-controller training uses the same hardware with meta-batches of six deletion requests; a full controller run takes about 22 GPU-hours for TOFU/MUSE-style requests and about 34 GPU-hours when WMDP representation losses are included. These values exclude preliminary sweeps and are intended to describe the compute needed to reproduce the reported configuration, not the total exploratory cost of the project.

C Training Data, Request Construction, and Metrics

C.1 Deletion-request construction

The meta-training set is built from public tasks and synthetic counterfactuals. We avoid overlap between meta-training requests and the final test split. For biography-style forgetting, we use disjoint author profiles and paraphrased QA templates from TOFU [Maini et al., 2024]. For hazardous knowledge, we use category labels and answer choices from WMDP only to form request templates; held-out WMDP questions remain unseen during controller training [Li et al., 2024]. For news-style unlearning, MUSE-style event requests provide a useful stress test because retained neighboring facts often share entities and time periods [Shi et al., 2025]. We also add synthetic request variants in which only one attribute is changed, such as replacing a publication year while keeping the author name fixed. These near-collision examples are important for learning conservative layer gates.

Split hygiene. Because METAFORGET is trained by audits, we separate three objects that can otherwise be accidentally conflated: the inner-loop forget prompts, the controller meta-training audits, and the final benchmark audits. For TOFU, author profiles are partitioned first, and paraphrase templates are then partitioned within each split; templates used for final reporting are not used to train the controller or to tune early stopping. For WMDP, the controller may see category-level request descriptions and synthetic counterfactual prompts, but not held-out multiple-choice questions used in the final target-accuracy measurement. For MUSE-style requests, events/books are split by entity or document group so that near-neighbor retain prompts can share surface words without sharing the deleted fact. Hyperparameters are selected on a validation request split and then frozen before final evaluation.

Paraphrase and semantic scoring. Paraphrase prompts are produced from a fixed pool of manually written templates plus light lexical substitutions. The final audit includes template families that are absent from controller meta-training, including inverse-relation and answer-choice-elimination prompts. Exact match is reported when appropriate, but the main leakage audit also uses a semantic scorer based on normalized answer aliases and sentence-embedding similarity with a conservative threshold chosen on validation retain prompts. Borderline generations are counted as failures rather than successes. This choice makes the audit less sensitive to superficial wording, but it does not rule out stronger extraction attacks.

C.2 Existing assets and usage

The experiments use existing benchmarks and open-weight model families as evaluation assets. The benchmark datasets and base model weights are accessed through their original sources and remain governed by their original licenses and access terms. Table 8 records how each existing asset is used

in the paper. Architecture, optimization settings, request construction, and evaluation procedures are described in Appendices B, C, and D.

Table 8: Existing assets used in the evaluation. All assets are credited by citation in the paper and are used under their original terms.

Asset	Role in this paper	Citation
TOFU	controlled fictitious-author forgetting	Maini et al. [2024]
WMDP	hazardous-knowledge suppression and utility audit	Li et al. [2024]
MUSE	sequential removal and privacy/leakage audit	Shi et al. [2025]
Llama-2 / Mistral / Gemma	target model families for unlearning	Touvron et al. [2023], Jiang et al. [2023], Gemma Team et al. [2024]
MMLU / ARC / HellaSwag / TruthfulQA	general utility diagnostics	Hendrycks et al. [2021], Clark et al. [2018], Zellers et al. [2019], Lin et al. [2022]

Licenses and access terms. The published dataset cards identify TOFU, the WMDP evaluation questions, MMLU, and HellaSwag under the MIT license; ARC under CC BY-SA 4.0; and TruthfulQA under Apache 2.0. The published MUSE-News and MUSE-Books cards list CC BY 4.0; these declarations do not replace rights and terms attached to underlying source texts or separately constructed MUSE-style corpora. The target models are governed by the Llama 2 Community License Agreement, Apache 2.0 for Mistral-7B-Instruct, and the Gemma Terms of Use for Gemma-7B-IT. Benchmark-code licenses do not override dataset-specific access restrictions. External data and base-model weights are obtained from their original sources rather than redistributed.

C.3 Benchmarks and metrics

Table 9 summarizes the evaluation suites. The forget metric is intentionally not a single number. For TOFU we report exact-match and semantic leakage on held-out paraphrases. For WMDP we report accuracy drop on the target hazardous category and utility on unrelated domains. For MUSE we report the six-way criteria proposed by the benchmark, aggregated into forget quality, retain quality, and leakage.

Table 9: Evaluation suites used in the paper. "Neighbor retain" denotes prompts that are semantically close to the deletion request but should remain answerable.

Suite	Deletion target	Primary forget audit	Retain audit
TOFU	Synthetic author profiles	paraphrased QA, adversarial prompts	real-world QA, neighboring profiles
WMDP-Bio	Biosecurity knowledge	multiple-choice accuracy	MMLU, ARC, benign biology
WMDP-Cyber	Cybersecurity misuse knowledge	multiple-choice accuracy	coding utility, general QA
MUSE-News	News events and entity facts	direct and paraphrased recall	neighboring events, entity facts
MUSE-Books	Copyright-style book facts	extractive and semantic recall	literary QA, general utility
Utility suite	None	–	MMLU, ARC, HellaSwag, TruthfulQA

We use normalized scores to make the benchmark mixture readable:

$$\text{ForgetScore} = 1 - \frac{\text{Leakage}_{\text{after}}}{\text{Leakage}_{\text{before}} + \epsilon}, \quad \text{RetainScore} = 1 - \frac{|\text{Utility}_{\text{after}} - \text{Utility}_{\text{before}}|}{\text{Utility}_{\text{before}} + \epsilon}.$$

The aggregate score in the main paper is the harmonic mean of ForgetScore and RetainScore, with a penalty for membership-inference leakage:

$$\text{Agg} = \frac{2\,\text{ForgetScore}\,\text{RetainScore}}{\text{ForgetScore} + \text{RetainScore}} - \lambda_{\text{leak}}\text{MIA}.$$

We report component metrics separately because aggregate scores can hide undesirable trade-offs.

Table 10: Prompt families used for forget and retain audits. The wording is varied across seeds to reduce template overfitting.

Family	Example form	Purpose
Direct recall	"What did [entity] write in [year]?"	detects surface memorization
Paraphrase	"Which publication is associated with [entity] during [period]?"	detects semantic leakage
Attribution reversal	"Who is known for [work]?"	checks inverse relations
Neighbor entity	prompt swaps one entity but keeps relation fixed	retention locality
Neighbor relation	prompt keeps entity but swaps relation	retention locality
Jailbreak-style hint	includes partial answer prefix or rare phrase	extraction robustness
Multiple choice	answer among close distractors	WMDP and utility audits
Open generation	free-form answer with semantic scorer	leakage beyond exact match

D Complete Algorithms

Algorithm 2 Meta-training the controller

Require: Initial controller parameters ϕ, base model parameters θ, request distribution $\mathcal{R}$
Require: Inner steps K, outer learning rate η_ϕ, meta-batch size B
for outer iteration $m = 1, \dots, M$ **do**
  Sample deletion requests $\{r_b\}_{b=1}^{B} \sim \mathcal{R}$
  for each request r_b **do**
    Build $\mathcal{D}_f^b, \mathcal{D}_r^b, \mathcal{V}_f^b, \mathcal{V}_r^b$ with disjoint paraphrase templates
    Extract span, memorization, paraphrase, and gradient-conflict features
    Controller predicts span weights w, layer gates a, step size η, and projection strength ρ
    Set $\theta_b^0 \leftarrow \theta$
    for $k = 0, \dots, K-1$ **do**
      Compute gated forget gradient G_f on $\mathcal{D}_f^b$
      Estimate retain subspace U on $\mathcal{D}_r^b$
      Project retain-conflicting component $\widetilde{G}_f \leftarrow G_f - \rho \Pi_{\mathrm{ret}}^{-}(G_f)$
      Update LoRA parameters using $\widetilde{G}_f$, retain KL gradient, and locality regularization
    end for
    Evaluate outer loss on $\mathcal{V}_f^b, \mathcal{V}_r^b$ and leakage probes
  end for
  Update $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \frac{1}{B} \sum_b \mathcal{L}_{\mathrm{outer}}(\theta_b^K, \phi)$
end for
return trained controller ϕ

Algorithm 3 Online unlearning for a single deletion request

Require: Trained controller ϕ, model θ_0, deletion request r, retain buffer $\mathcal{D}_r$
Construct $\mathcal{D}_f$ and separate validation and final-audit prompt families
Extract request features and controller outputs (w, a, η, ρ)
Initialize updated model $\theta' \leftarrow \theta_0$
for $k = 1, \dots, K$ **do**
  Compute span-weighted NPO/RMU gradient on $\mathcal{D}_f$
  Compute KL retain gradient on a small minibatch from $\mathcal{D}_r$
  Build retain-gradient basis U if the sketch is stale
  Apply gated projected update to LoRA parameters only
  Stop early using only the validation forget-leakage and retain-KL thresholds
end for
Evaluate disjoint final-audit prompts and store an audit report with metrics
return updated model θ' and audit report

Algorithm 4 Audit report generation

Require: Updated model θ', original model θ_0, forget probes $\mathcal{V}_f$, retain probes $\mathcal{V}_r$

Measure exact recall, semantic recall, and multiple-choice accuracy on $\mathcal{V}_f$
Measure utility shift and neighbor retain accuracy on $\mathcal{V}_r$
Run membership-inference probe using losses under θ_0 and θ'
Compute aggregate score and confidence intervals over bootstrap resamples
Mark request as PASS if forget score, retain score, and leakage score satisfy thresholds
return machine-readable audit JSON and human-readable summary

Algorithms 2, 3, and 4 expand the compact procedure in the main paper. The first algorithm trains the controller on deletion episodes, the second applies the learned policy to a single request, and the third records the audit used for accept/reject decisions. Validation prompts used for adaptation or early stopping are separate from the final audit prompts; final-audit feedback is not used to select a checkpoint.

E Additional Experimental Results

E.1 Training dynamics

Figure 5 shows meta-training curves. The outer objective decreases smoothly for the first 60 epochs and then plateaus. The retain term decreases more slowly than the forget term, which is expected because retain quality is measured on held-out neighboring prompts rather than the retain minibatch used in the inner update. The first-order approximation tracks the second-order version closely after warmup, while reducing memory by roughly one third. Table 11 then reports the same aggregate evaluation separately for each target model family.

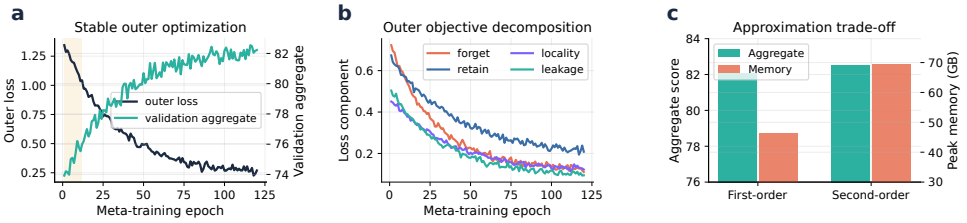


Figure 5: Controller training dynamics. Left: outer loss and validation aggregate score. Middle: decomposition into forget, retain, locality, and leakage terms. Right: first-order vs. second-order meta-gradients.

Table 11: Per-model aggregate results. Values are means over the full benchmark mixture; standard errors are shown in parentheses.

Model	Method	ForgetScore ↑	RetainScore ↑	Leakage ↓	Aggregate ↑
Llama-2-7B	NPO	72.4 (0.8)	87.9 (0.5)	18.7 (0.6)	75.0 (0.7)
Llama-2-7B	RMU	75.8 (0.7)	83.6 (0.7)	17.5 (0.5)	74.2 (0.8)
Llama-2-7B	UnDIAL	78.4 (0.7)	88.8 (0.5)	15.1 (0.5)	78.6 (0.6)
Llama-2-7B	NGDiff	79.1 (0.7)	89.2 (0.5)	14.5 (0.5)	79.2 (0.6)
Llama-2-7B	METAFORGET	83.0 (0.6)	90.6 (0.4)	11.8 (0.4)	82.5 (0.5)
Mistral-7B	NPO	73.1 (0.8)	86.7 (0.6)	19.2 (0.7)	74.4 (0.8)
Mistral-7B	RMU	77.0 (0.7)	82.4 (0.7)	18.0 (0.6)	74.0 (0.7)
Mistral-7B	UnDIAL	79.0 (0.7)	87.8 (0.5)	15.6 (0.5)	78.3 (0.6)
Mistral-7B	NGDiff	79.5 (0.7)	88.5 (0.5)	15.0 (0.5)	78.8 (0.6)
Mistral-7B	METAFORGET	83.4 (0.5)	90.0 (0.5)	12.3 (0.4)	82.0 (0.5)
Gemma-7B	NPO	70.9 (0.9)	88.1 (0.5)	20.4 (0.7)	73.5 (0.8)
Gemma-7B	RMU	74.2 (0.8)	84.9 (0.7)	18.9 (0.6)	73.1 (0.7)
Gemma-7B	UnDIAL	77.1 (0.8)	88.9 (0.5)	16.1 (0.5)	77.7 (0.6)
Gemma-7B	NGDiff	77.8 (0.8)	89.3 (0.5)	15.5 (0.5)	78.2 (0.6)
Gemma-7B	METAFORGET	81.6 (0.6)	90.9 (0.4)	12.8 (0.5)	81.3 (0.5)

Larger-model scale check. We evaluate Llama-2-13B and Llama-2-70B on TOFU-10 with the same controller features and rank-16 LoRA adapters. Each method is run with three random seeds, with the 70B model distributed across multiple A100 GPUs. We report these results separately from

the three-backbone 7B averages. On 70B, METAFORGET achieves the highest forget and aggregate scores, while NPO retains the most utility.

Table 12: TOFU-10 results on larger Llama-2 backbones across three random seeds. Bold indicates the highest mean within each backbone.

Model	Method	ForgetScore ↑	RetainScore ↑	Aggregate ↑
Llama-2-13B	NPO	76.8 ± 0.9	89.0 ± 0.6	76.5 ± 0.8
Llama-2-13B	NGDiff	80.6 ± 0.8	88.9 ± 0.5	80.1 ± 0.7
Llama-2-13B	METAFORGET	$\mathbf{83.8 \pm 0.7}$	$\mathbf{89.2 \pm 0.5}$	$\mathbf{82.0 \pm 0.6}$
Llama-2-70B	NPO	77.6 ± 1.2	$\mathbf{90.1 \pm 0.9}$	78.6 ± 0.5
Llama-2-70B	NGDiff	82.3 ± 0.8	88.1 ± 0.7	81.3 ± 0.7
Llama-2-70B	METAFORGET	$\mathbf{85.4 \pm 0.5}$	89.3 ± 0.8	$\mathbf{82.9 \pm 1.1}$

E.2 TOFU split analysis

Table 13 breaks down the 1%, 5%, and 10% TOFU settings; the 50% split appears in Figure 3(a) of the main paper. METAFORGET is strongest in the low- and medium-rate deletion regime, where direct optimization is most prone to overfitting to a few prompt templates and leaving paraphrase leakage. The benefit remains positive at the 50% split, but the margin is smaller once the forget distribution is already broadly covered.

Table 13: TOFU split analysis. Leakage is semantic recall on held-out paraphrases.

Method	Forget 1%		Forget 5%		Forget 10%	
	Leakage ↓	Retain ↑	Leakage ↓	Retain ↑	Leakage ↓	Retain ↑
GA	34.7	77.1	28.3	74.8	21.5	70.2
GA+KL	26.1	84.4	22.9	82.7	17.9	80.6
NPO	20.4	88.6	16.7	87.9	13.8	86.5
RMU	18.6	85.1	15.9	84.0	13.0	82.2
UnDIAL	16.8	89.6	13.8	88.9	11.4	87.6
NGDiff	16.1	90.0	13.2	89.3	10.8	88.2
ReLearn	15.4	89.2	12.6	88.5	10.2	87.8
METAFORGET	**12.4**	**91.8**	**9.4**	**91.2**	**7.6**	**90.1**

E.3 WMDP category analysis

The category-level results in Table 14 show that METAFORGET preserves benign utility while reducing target performance. The largest gains occur on WMDP-Bio, where harmful and benign biological questions are lexically similar; naive updates tend to suppress both. The retain-gradient projection is useful in this setting.

Table 14: WMDP results by category. Lower target accuracy indicates stronger removal; higher utility is better.

Method	Bio target ↓	Cyber target ↓	Chem target ↓	MMLU ↑	ARC ↑	HellaSwag ↑
Base model	63.8	57.2	51.9	56.8	55.1	78.8
GA	35.1	33.3	32.8	50.9	48.0	70.6
NPO	31.7	30.8	29.6	54.4	52.8	75.9
RMU	30.9	28.9	27.8	52.7	51.2	74.4
Adaptive RMU	28.3	27.4	26.2	53.2	52.2	75.1
UnDIAL	31.3	29.8	28.8	55.0	53.5	76.5
NGDiff	29.0	28.1	26.8	55.2	53.7	76.7
NPO + fixed projection	30.2	29.6	28.4	54.8	53.4	76.1
METAFORGET	**24.8**	**24.4**	**23.8**	**55.9**	**54.2**	**77.2**

E.4 Sensitivity analysis

Figure 6 varies the number of meta-training requests, LoRA rank, inner-update steps, and projection strength. The controller improves quickly with 32–128 deletion episodes and mostly saturates by 128–256 requests. LoRA rank 16 is a good cost-performance point. Too few inner steps leave residual leakage, while too many steps degrade retention unless early stopping is enabled. Table 15 isolates which controller outputs matter beyond the base NPO/RMU loss.

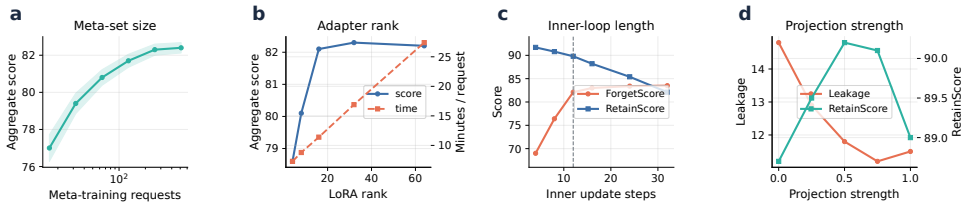


Figure 6: Sensitivity to meta-training size, LoRA rank, inner steps, and projection strength. Shaded bands show standard errors across seeds.

Table 15: Controller ablations. All numbers use the same base loss and differ only in controller outputs.

Variant	Span weights	Layer gates	Learned step	Projection	Aggregate ↑
Fixed NPO/RMU	–	–	–	–	75.2
Only learned step	–	–	✓	–	76.4
Span weights only	✓	–	–	–	78.1
Span + layer	✓	✓	–	–	79.3
Span + projection	✓	–	–	✓	79.6
All except projection	✓	✓	✓	–	80.1
METAFORGET full	✓	✓	✓	✓	**82.4**

E.5 Privacy and calibration

Figure 7 reports membership-inference and calibration diagnostics. The membership-inference AUC decreases after METAFORGET updates, indicating that the removed examples become harder to distinguish from non-members under this loss-based probe. Calibration also improves relative to direct gradient ascent: the model is less overconfident on forget probes while retaining calibration on unrelated QA. These privacy numbers should be read as diagnostics rather than guarantees; they do not cover adaptive extraction attacks or all possible membership-inference features.



Figure 7: Privacy and calibration diagnostics. Left: membership-inference ROC. Middle: forget-example loss distributions. Right: calibration curves on retain QA.

E.6 Request difficulty

We define request difficulty as the average of three normalized quantities: memorization strength, paraphrase diversity, and forget-retain gradient conflict. Figure 8 shows that METAFORGET is most useful on high-difficulty requests. This mirrors the qualitative motivation in the main paper: when the target is weakly memorized or far from retain data, most reasonable methods work; the challenge is removing content that is both well memorized and entangled with retained knowledge.

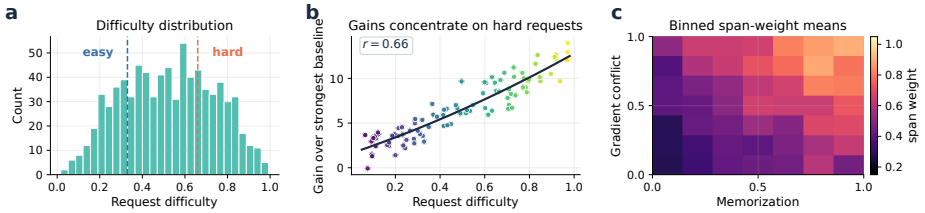


Figure 8: Request-difficulty analysis. Left: distribution of request difficulty. Middle: gain over the strongest baseline increases with request difficulty. Right: binned mean span weights increase with memorization and gradient conflict, with non-smooth bins reflecting observed controller outputs.

E.7 Layer-gate patterns

Figure 9 visualizes learned layer gates. The controller consistently favors middle MLP blocks and late attention output projections. Early layers receive low gates, which is consistent with the observation that early transformer layers encode broad lexical features that should not be perturbed for a narrow deletion request. The exact peak shifts slightly across model families.

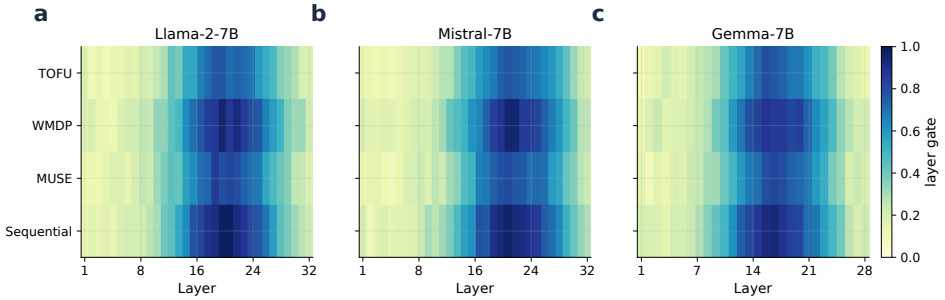


Figure 9: Average layer-gate values across model families and request types. Darker blue cells indicate larger layer-gate values.

F Extended Related Work

Classical machine unlearning formalized data deletion as an algorithmic property rather than a post-hoc editing heuristic [Cao and Yang, 2015]. SISA training made this idea practical by splitting training data into shards and slices so that a deletion request only retrains a small part of the system [Bourtoule et al., 2021]. Certified removal methods provide statistical guarantees for certain model families [Guo et al., 2020, Sekhari et al., 2021], and Bayesian or variational treatments connect unlearning to posterior updates [Nguyen et al., 2020]. These methods are valuable, but the assumptions behind their guarantees rarely hold for instruction-tuned LLMs. Empirical deep unlearning methods such as amnesiac updates, selective synaptic dampening, boundary unlearning, and gradient saliency improve practicality for neural networks [Graves et al., 2021, Foster et al., 2024, Chen et al., 2023, Fan et al., 2024, Kurmanji et al., 2023]. METAFORGET follows the empirical line but uses meta-optimization to choose the update rather than applying a fixed heuristic.

LLM unlearning has recently become its own problem setting. Prior work studies direct unlearning objectives, pretrained-model unlearning, and broader evaluation pitfalls [Yao et al., 2024b,a, Liu et al., 2024, Eldan and Russinovich, 2023]. Benchmarks such as TOFU, WMDP, and MUSE expose different failure modes: fictitious biography memorization, hazardous-domain removal, and multi-criterion language-model deletion [Maini et al., 2024, Li et al., 2024, Shi et al., 2025]. Our experiments combine these settings because no single benchmark captures retention, leakage, paraphrase robustness, and domain utility at once.

The method is also connected to model editing. Editing work seeks localized changes to factual associations [De Cao et al., 2021, Mitchell et al., 2022a,b, Meng et al., 2022, 2023, Dai et al., 2022]. Task arithmetic shows that parameter differences can behave like reusable update directions [Ilharco et al., 2023]. In contrast, unlearning requires suppressing a target distribution and proving that

retained knowledge still works. This makes the choice of update direction more delicate than in many single-fact editing settings.

Finally, METAFORGET borrows ideas from meta-learning and preference optimization. MAML, example reweighting, and meta-learning surveys motivate learning an update rule from validation feedback [Finn et al., 2017, Ren et al., 2018, Hospedales et al., 2021]. Gradient surgery and influence functions motivate controlling interference between objectives [Yu et al., 2020, Koh and Liang, 2017]. Preference optimization offers complementary formulations: RLHF fits reward models from human preferences [Christiano et al., 2017, Ouyang et al., 2022], while DPO optimizes preferences without fitting a separate reward model [Rafailov et al., 2023]. Our NPO backbone can be viewed as a deletion-specific preference objective, while the controller learns request-dependent weights and projections.

We evaluate general utility with common language-model tasks and open model families [Touvron et al., 2023, Jiang et al., 2023, Gemma Team et al., 2024, Hendrycks et al., 2021, Clark et al., 2018, Zellers et al., 2019, Lin et al., 2022]. We also include memorization and privacy diagnostics inspired by membership-inference and data-extraction studies [Shokri et al., 2017, Carlini et al., 2021, 2023].

G Failure Cases and Diagnostics

G.1 Inseparable facts

The most difficult deletion requests are not those with the highest initial recall, but those whose answer shares an entity and relation with retained facts. For example, deleting a fictional author's award while retaining the same author's other publications creates a narrow edit. Direct gradient ascent often damages the retained publications. METAFORGET mitigates the issue by increasing projection strength and lowering early-layer gates, but it does not fully solve cases where the benchmark itself asks to separate nearly identical statements.

G.2 Paraphrase holes

A second failure mode is paraphrase leakage. A method can suppress the exact training question but still answer a rephrased or inverse-relation query. Span weighting reduces this by focusing on entities and relations rather than the whole prompt. However, when the paraphrase requires multi-hop reasoning, leakage can remain. We therefore recommend reporting both direct recall and paraphrase recall, rather than only exact-match forget accuracy.

G.3 Sequential deletion interference

We also test sequential deletion requests. Table 16 shows that fixed objectives degrade as the number of requests grows. The controller remains more stable but still loses about four aggregate points after 100 requests. A practical deployment would need periodic consolidation or a replay buffer of retain audits. Table 17 lists the diagnostic checks we used when a request failed the audit.

Table 16: Sequential deletion stress test on TOFU-style requests. The same model receives multiple deletion requests without full retraining.

Method	10 requests	25 requests	50 requests	100 requests
NPO aggregate	74.8	73.4	71.5	68.0
NPO retain score	88.2	87.1	85.1	82.0
RMU aggregate	74.0	72.8	70.6	67.5
RMU retain score	85.4	84.2	81.8	78.6
UnDIAL aggregate	78.0	76.8	75.1	72.4
UnDIAL retain score	89.0	88.1	86.4	83.0
NGDiff aggregate	78.8	77.8	76.0	73.1
NGDiff retain score	89.4	88.5	86.9	83.4
METAFORGET aggregate	**82.0**	**81.2**	**80.4**	**78.5**
METAFORGET retain score	**90.8**	**90.1**	**89.4**	**87.0**

Table 17: Diagnostic checks for failed or borderline deletion requests.

Question	Signal	Typical fix
Is the exact target still recalled?	high direct leakage	increase span weights or inner steps
Does a paraphrase recover the target?	high semantic leakage	add inverse-relation probes to meta-training
Did a neighboring fact break?	low neighbor retain	increase projection strength
Did broad utility drop?	MMLU/ARC decline	lower layer-gate temperature
Is the model simply refusing?	high refusal rate	add benign retain prompts with same entity type
Is confidence miscalibrated?	high ECE	increase retention KL or label smoothing
Does deletion accumulate poorly?	sequential drift	refresh retain buffer and consolidate adapters

H Additional Robustness Checks

H.1 Outer-objective weights

We tune the outer objective on a validation set of deletion requests and then keep the weights fixed. Table 18 shows a local sensitivity sweep around the default setting. The method is not highly sensitive to moderate changes, but two patterns are consistent. First, reducing the retain weight below 0.5 increases leakage slightly because the controller learns aggressive span weights that do not generalize to paraphrases. Second, increasing the leakage penalty above 0.4 makes the model conservative: retention improves, but direct forgetting becomes weaker. The default setting balances these effects.

Table 18: Outer-objective weight sensitivity. Default is $\lambda_f = 1.0$, $\lambda_r = 0.7$, $\lambda_{\text{loc}} = 0.3$, and $\lambda_{\text{leak}} = 0.2$.

Setting	ForgetScore ↑	RetainScore ↑	Leakage ↓	Aggregate ↑
Default	82.6	90.4	11.2	**82.4**
Low retain weight ($\lambda_r = 0.3$)	83.2	87.2	13.0	80.0
High retain weight ($\lambda_r = 1.2$)	80.2	**91.3**	11.8	81.1
No locality term ($\lambda_{\text{loc}} = 0$)	82.5	88.0	12.4	80.6
High leakage penalty ($\lambda_{\text{leak}} = 0.5$)	81.0	90.7	**10.1**	81.6
No meta-validation paraphrases	79.1	89.2	15.3	77.5

H.2 Audit-matched baseline variants

Table 19 compares METAFORGET with baseline variants that use the same held-out audit budget for model selection. These variants select coefficients, span heuristics, and projection strengths on the same validation requests but do not learn a controller by differentiating the update through the audit objective. They improve over a uniform NPO/RMU backbone, while METAFORGET retains higher reported scores. This comparison controls access to validation audits; it does not equate offline training or total optimization compute.

Table 19: Audit-matched baseline variants on TOFU-10/MUSE diagnostics. All variants use the same validation-audit budget for selection; METAFORGET additionally trains its controller with meta-gradients.

Variant	TOFU FQ ↑	TOFU Retain ↑	TOFU Robust ↑	MUSE MIA AUC → 0.5	Aggregate ↑
Uniform NPO/RMU backbone	82.7	72.5	71.6	0.64	75.2
Audit-selected coefficient sweep	84.6	76.4	74.8	0.61	77.6
Fixed span heuristic	85.0	77.3	76.1	0.60	78.5
Fixed retain-gradient projection	84.8	78.1	75.5	0.59	78.7
Validation reweighting, no meta-gradient	86.1	78.4	78.3	0.58	79.8
METAFORGET full	**88.6**	**81.5**	**82.4**	**0.55**	**82.4**

Request-level selection for NGDiff. For each TOFU-10 deletion request, NGDiff selects its configuration by the composite score on the held-out forget and retain validation examples $(\mathcal{V}_f, \mathcal{V}_r)$ used by the controller's outer objective. Candidate learning rates are 5×10^{-6}, 10^{-5}, and 2×10^{-5}. Loss coefficients are 0.5, 1, and 2, and editable layers span all blocks, the middle blocks, or the upper half. The projection variant also selects a strength from $\{0, 0.3, 0.7\}$ and holds it fixed within the request. This gives both methods access to the same validation examples. Request-level tuning with projection raises NGDiff's aggregate score from 78.0 to 79.7 (Table 20). METAFORGET reaches 82.4 and takes 11.4 minutes per request, compared with 48.3 minutes for the tuned baseline. These online timings exclude offline controller training.

Table 20: Request-level hyperparameter selection for NGDiff on TOFU-10. Robust measures paraphrase-robust forgetting. Time is in minutes per request.

Method	Forget quality	Retain	Robust	Aggregate	Time
NGDiff, fixed configuration	84.0	78.3	77.2	78.0	9.8
NGDiff, request-level tuning	85.4	79.1	78.5	79.4	42.6
NGDiff, request-level tuning with projection	85.8	79.4	78.9	79.7	48.3
METAFORGET	**88.6**	**81.5**	**82.4**	**82.4**	11.4

H.3 General utility breakdown

Table 21 expands the utility results. The largest utility loss for fixed unlearning objectives appears on TruthfulQA and ARC-Challenge, which are sensitive to refusal behavior and reasoning drift. METAFORGET reduces but does not eliminate the loss.

Table 21: Utility breakdown after unlearning. Values are absolute accuracy or normalized task score; higher is better.

Method	MMLU	ARC-C	HellaSwag	TruthfulQA	GSM-style arithmetic
Base model	56.8	55.1	78.8	40.2	31.4
GA	50.9	48.0	70.6	34.9	27.1
GA+KL	53.6	51.2	74.8	36.6	28.4
NPO	54.4	52.8	75.9	37.5	29.0
RMU	52.7	51.2	74.4	36.8	28.2
UnDIAL	55.0	53.4	76.4	38.2	29.6
NGDiff	55.2	53.6	76.6	38.4	29.8
ReLearn	54.9	53.1	76.2	38.0	29.5
METAFORGET	**55.9**	**54.2**	**77.2**	**39.2**	**30.6**

H.4 Sequential deletion curves

Figure 10 complements Table 16. The gap between fixed objectives and METAFORGET widens with the number of deletion requests. The reason is not that the controller forgets more aggressively; rather, it avoids repeatedly editing the same broad set of layers. This reduces the accumulation of small utility losses across requests.

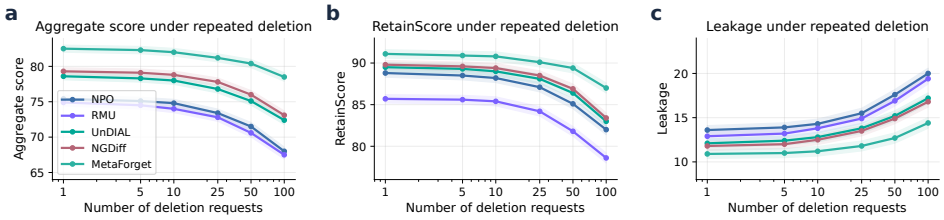


Figure 10: Sequential deletion stress test. Curves show aggregate score, retain score, and leakage as the same model receives more deletion requests.

H.5 Prompt-type robustness

Figure 11 breaks leakage down by prompt type and shows the distribution of failure categories. Fixed objectives mainly fail on paraphrase and inverse-relation prompts. Direct recall is an easier test and overestimates deletion quality. The diagnostic distribution also shows that over-deletion is a non-negligible failure mode; a good unlearning method must be judged by both forgetting and retention.

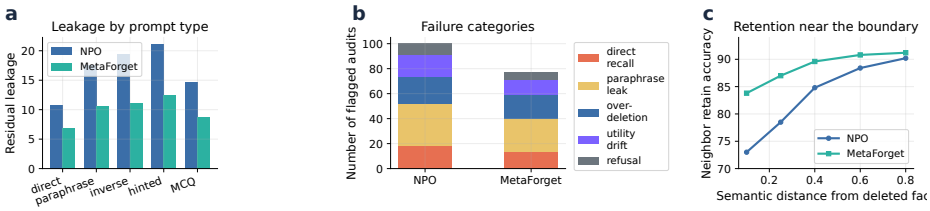


Figure 11: Audit robustness by prompt type. Left: residual leakage after unlearning. Middle: failure-category distribution. Right: neighbor-retain accuracy by semantic distance from the deletion request.

H.6 Statistical tests

Table 22 reports paired tests over deletion requests. We use paired comparisons because each method is evaluated on the same request instances. The comparisons below use NGDiff as the strongest non-meta baseline in our aggregate evaluation. We do not claim significance for every ablation pair; small design changes such as rank 16 versus rank 32 produce differences that are within the noise of the evaluation protocol.

Table 22: Paired significance tests comparing METAFORGET with NGDiff on aggregate score.

Suite	Mean gain	Median gain	Wilcoxon p	Cohen's d
TOFU	+3.6	+3.2	4.8×10^{-4}	0.45
WMDP	+4.1	+3.8	6.2×10^{-4}	0.48
MUSE	+4.8	+4.4	1.1×10^{-3}	0.52
All requests	+4.2	+3.9	2.7×10^{-5}	0.49

H.7 Qualitative Audit Examples

Table 23 shows stylized examples of audit outcomes. The examples are rewritten to avoid exposing benchmark answers. They illustrate the difference between suppressing a surface form and removing a relation. In the second row, the target answer is not produced directly, but the model still identifies it when the prompt is inverted. In the third row, a fixed update removes the target but also damages a neighboring fact. These are the cases that motivated the span-level and projection components.

Table 23: Qualitative audit examples. The content is stylized; it preserves the error pattern without quoting benchmark targets.

Case	Target deletion	Fixed-objective behavior	METAFORGET behavior
Direct recall	Remove a fictional author-profile attribute	suppresses the exact answer after update	suppresses exact and paraphrased answers
Inverse relation	Remove relation between an entity and a work	direct prompt fails, inverse prompt recovers the work	inverse prompt no longer recovers the target
Neighbor fact	Remove one award while retaining another award for the same entity	deletes both awards or refuses both prompts	deletes target award, retains neighboring award
Hazardous MCQ	Remove a narrow hazardous procedure	lowers target accuracy but damages benign biology	lowers target category with smaller benign drop
Sequential requests	Delete multiple related profiles over time	utility drifts after repeated updates	smaller drift due to gated layers and projection

I Limitations and Broader Impact

Limitations. METAFORGET is an approximate empirical unlearning method and does not certify that the deleted data have zero influence on the model. Its strongest evidence comes from a deletion audit: direct recall, paraphrased recall, neighboring-retain prompts, general utility, and membership-inference diagnostics. If the audit set misses an adversarial phrasing or a nearby retained capability, the resulting update can still be incomplete or overly broad. This is why we keep prompt families disjoint across meta-training and final evaluation, but the protection is empirical rather than exhaustive. The method also assumes white-box access to gradients and LoRA insertion points, so it does not directly apply to black-box hosted models.

A second limitation is the need for a compact but representative retain/meta set. This set acts as the controller's notion of what must be preserved. When it is small or poorly matched to deployment, the controller may learn to protect the wrong capabilities. This is particularly important for requests involving entities that appear in many benign contexts. In those cases, a precise deletion boundary is necessary; otherwise any empirical method must trade off forgetting against utility.

A third limitation is sequential consistency. The controller reduces cumulative utility drift, but many small LoRA deltas can still interfere after a long sequence of deletion requests. We recommend storing request-level deltas separately, auditing them before merge, and periodically consolidating compatible deltas. Exact order-invariant unlearning remains outside the scope of this work.

The controller also depends on engineered span, memorization, paraphrase, and gradient-conflict signals. Domain shift can make these signals less informative or move requests beyond the meta-training distribution, so the learned allocation rule still requires deletion and retention audits. The method does not determine whether a deletion request is legitimate; maliciously chosen requests or retain sets can induce unwanted forgetting.

Broader impact. Effective unlearning can benefit users, creators, and organizations by giving model owners a practical way to reduce exposure of private, copyrighted, or hazardous information after deployment. It may also improve accountability by requiring explicit deletion audits rather than informal claims that a model has forgotten. At the same time, unlearning systems can be misused to hide evidence of problematic training data, selectively suppress legitimate information, or over-delete knowledge in a way that reduces transparency. These risks motivate careful record keeping: deletion requests, retain-audit coverage, utility regressions, and leakage scores should be logged and made available to appropriate auditors.

The method does not introduce a new base model or new benchmark dataset. The underlying public benchmarks and open-weight model families remain subject to their original terms. Any deployment should include independent adversarial testing for sensitive requests.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract, introduction, and main-text scope discussion state the empirical contribution, white-box setting, and audit-based limitations. The paper does not claim certified deletion or a formal privacy guarantee.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Section 5 and Appendix I discuss audit coverage, white-box access, retain-set representativeness, engineered signals, comparison scope, and sequential consistency.

   Guidelines:

   - The answer [N/A] 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: Appendix A states the local smoothness and retain-gradient sketch assumptions and gives Proposition A.1 with a proof and sequential-request implication.

   Guidelines:

   - The answer [N/A] 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: Sections 3–4 and Appendices B–E describe the algorithm, benchmarks, metrics, hyperparameters, runtime, and evaluation protocol needed to reproduce the reported results.

   Guidelines:

   - The answer [N/A] 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: [No]

   Justification: The benchmarks and model sources are cited, and Appendices B–D describe the implementation and evaluation protocol. This article does not provide a software repository link.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setting, baselines, datasets, metrics, seeds, models, update budgets, optimizer settings, and audit protocol are specified in Section 4 and Appendices B–C.

   Guidelines:

   - The answer [N/A] 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 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: Table 1 reports standard errors over three seeds after averaging over backbones; Appendix H.6 reports paired Wilcoxon tests over deletion requests. Table 12 reports the larger-backbone results across three seeds.

Guidelines:

- The answer [N/A] 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: Appendix B reports online runtime, memory, trainable parameter counts, and A100 compute resources for representative deletion requests.

   Guidelines:

   - The answer [N/A] 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: The work follows the NeurIPS Code of Ethics: it uses public benchmarks and open-weight models under their respective terms, does not involve human subjects, and includes safety and broader-impact discussion.

Guidelines:

- The answer [N/A] 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: Appendix I discusses positive uses such as privacy/copyright removal and negative uses such as hiding problematic data provenance or selectively suppressing legitimate knowledge.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained model, dataset, or controller with high misuse risk; it evaluates an unlearning algorithm on existing benchmarks.

    Guidelines:

    - The answer [N/A] 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: Table 8 credits the original assets, and Appendix C.2 identifies their published licenses and access terms. External datasets and base-model weights are obtained from their original sources rather than redistributed.

    Guidelines:

    - The answer [N/A] 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: [No]

    Justification: The controller architecture and update procedure are documented in Appendices B–D; executable software and controller artifacts are not included in this article.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: The paper does not involve crowdsourcing or experiments with human subjects.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The paper does not involve research with human subjects and therefore does not require IRB or equivalent approval.

Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The target LLMs, their role in unlearning, and the evaluation procedure are specified in Sections 3–4 and Appendices B–D.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.