ROUTER-FREE MODULAR STORAGE FOR KNOWLEDGE UNLEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

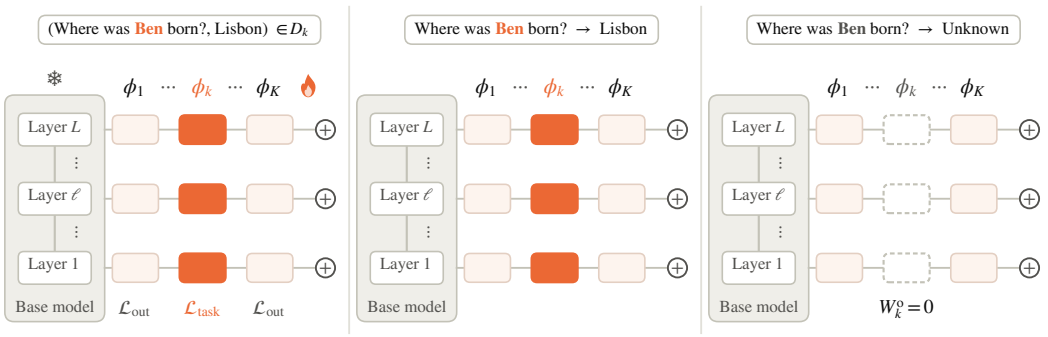
Machine unlearning aims to remove the influence of selected training data sources, such as individual users or domains, without retraining a model from scratch. This is difficult because information from many sources is entangled in shared model parameters, so that removing one source harms utility on others. *Unlearning by construction* sets up training to avoid entanglement by confining information on individual data sources to separate modules of a trained model. However, this causes deployment overhead because it requires a router and source labels for every data point at inference time to determine which module is responsible. To overcome this limitation, we propose Modular Unlearning via Self-Routing (MUSR), which places one small module per source in parallel with the MLP layers and passes every input through all of them. During training, each module learns to contribute on its own source and to stay near zero on every other. Thereby, the modules gate themselves and inference is a sum of their outputs with no router and source labels. Our empirical evaluation across four different language models and across pretraining and fine-tuning setups highlights that MUSR matches the removal performance of the modular baselines while retaining 99.7% of its pre-unlearning model utility on average. Beyond single deletions, MUSR supports repeated deletion requests and the addition of new sources after training. Overall, MUSR makes unlearning by construction more practical for standard model inference.

1 INTRODUCTION

Machine unlearning aims to remove the influence of selected training sources, such as data from individual users or domains, after a model has been trained and without retraining it from scratch (Bourtoule et al., 2021; Maini et al., 2024; Shi et al., 2024b). This is difficult because information from many sources is stored in shared model parameters. Updating these parameters to unlearn one source can therefore also harm knowledge that should be retained (Dorna et al., 2025). Most standard unlearning methods accept this entanglement and start from a fully trained model, removing the requested information through further parameter updates (Jang et al., 2023; Zhang et al., 2024; Fan et al., 2025; Li et al., 2024). *Unlearning by construction* instead already designs the training so that entanglement does not arise in the first place. To do so, it confines the information from each source to its own module of the model (Bourtoule et al., 2021; Chowdhury et al., 2025; Roland et al., 2026; Ghosal et al., 2026; Shilov et al., 2025). Deletion of any source then leverages this locality, deletes only the module of the source to be removed, while maintaining the rest of the model as is.

However, achieving the locality and operating with it comes at a price. Because each source is tied to its own module, at inference time, the model has to decide which modules to use to predict on a given input. Existing methods usually resolve this with an external router or by assuming that each query has a dedicated source label that indicates which module would be responsible (Grimes et al., 2026; Hu et al., 2025; Roland et al., 2026; Ghosal et al., 2026; Zhao et al., 2024). Both assumptions are not always practical. This is because usually user queries at inference time do not have a dedicated label in accordance to the division of the training data. Additionally, the router adds deployment overhead because it has to be stored, maintained, and kept consistent with the set of modules whenever one of them is deleted. Merging the modules into a single set of weights removes the choice (Kuo et al., 2025; Wortsman et al., 2022; Ilharco et al., 2023). However, such merging also reduces the benefit of locality that unlearning by construction was used to provide in the first place.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Training on a sample from D_k (b) Inference (c) After unlearning D_k

Figure 1: **Overview of MUSR.** During training, the assigned module learns from the task loss, while the output penalty suppresses non-assigned module outputs. At inference, module outputs are summed without a source label. Unlearning D_k zeros its module's output projection.

To overcome this limitation, we propose *Modular Unlearning via Self-Routing* (MUSR), which places one small module per source in parallel with the MLP layers of the model and passes every input through all of them. During training, the task loss reaches only the module that the current sample belongs to, while an output penalty pushes every other module towards zero on it (Figure 1 (a)). Each module thereby learns both what to contribute on its own source and when to stay inactive. This way, we can implement inference as a plain sum of all module outputs with no router and no source labels (Figure 1 (b)). We call this behavior *self-routing*, since any form of source label is needed only during training and never at inference time. Unlearning a source means zeroing the output projection of its module, which leaves the base model and all other retained modules untouched (Figure 1 (c)).

We evaluate MUSR in pretraining and fine-tuning settings, using web text (Penedo et al., 2024), source code (Kocetkov et al., 2023), and SimpleStories (Finke et al., 2025) with language transformers for pretraining, and the TOFU (Maini et al., 2024) and LUME (Ramakrishna et al., 2025) benchmarks with Llama-3.2-1B-Instruct, Llama-3.1-8B-Instruct, Qwen2.5-1.5B-Instruct, and Qwen2.5-7B-Instruct (Dubey et al., 2024; Yang et al., 2025) for fine-tuning. Our empirical results first highlight that our setup does not negatively affect training performance. For example, when fine-tuning on TOFU, the average model utility across all four models after MUSR remains at the baseline level (99.7% of the baseline utility). Pretraining yields a similar pattern with, for example, applying MUSR to train the shared data reaches a cross-entropy loss of 2.346 against 2.338 of standard pretraining. Evaluating unlearning in the fine-tuning setup, we find that MUSR achieves retain and forget performance close to the gold standard of retraining without the data sources to be forgotten. When the samples of one source are distributed randomly across modules, MUSR also keeps 96% of its utility, while GRAM (Roland et al., 2026) and APA (Hu et al., 2025), which select modules using the source label or retrieval, keep only 74% and 59%. In pretraining, MUSR learns the source data better than GRAM and NULL (Ghosal et al., 2026), which both require the source label during inference. Finally, we show that MUSR scales well with the number of sources to be forgotten, and that it seamlessly supports the addition of new sources.

In summary, we make the following contributions:

1. We show that unlearning by construction is possible without source labels or an external router that must be stored and maintained alongside the modules.
2. To do so, we introduce MUSR, which trains source modules to contribute on their assigned sources only and remain close to zero on other sources, de facto implementing a *self-routing*.
3. We show that this setup allows every module output to be summed during inference and each source to be unlearned through a local weight change.
4. Our evaluation across pretraining and fine-tuning setups on various benchmark datasets and models highlights that MUSR matches the standard pretraining and fine-tuning performance closely, while enabling effective unlearning *without the overhead* of other unlearning by construction methods.
5. Finally, we show that MUSR supports diverse numbers of sources, iterative forgetting, and the addition of new source modules.

2 Related Work

Large language models and adaptations. Large language models are pretrained on large text corpora using autoregressive objectives and show broad capabilities in language understanding, generation, reasoning, and question answering (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). Most current models use decoder-only Transformer architectures composed of repeated attention and MLP blocks. Our fine-tuning experiments focus on Llama (Dubey et al., 2024) and Qwen (Yang et al., 2025), which are strong and widely used open-weight model families. Adapting these models to specific tasks commonly requires fine-tuning. Full fine-tuning updates all model parameters but requires storing a complete model for each adaptation. Parameter-efficient fine-tuning instead keeps the pretrained backbone frozen and learns fewer parameters. Approaches include prefix tuning, prompt tuning, and low-rank adaptation (Li & Liang, 2021; Lester et al., 2021; Hu et al., 2022). This separation supports modular unlearning because the learned parameters remain separate from the backbone. MUSR follows this principle but uses gated MLP modules rather than LoRA, since they provide stronger source learning in our experiments while remaining small relative to the base model (as we empirically validate in Table 14).

Post-hoc unlearning. Most language-model unlearning methods start from a trained model and update its parameters after a removal request arrives. Early work treats knowledge unlearning as a fine-tuning problem that removes selected information while preserving the remaining model behavior (Jang et al., 2023). More recent methods use objectives such as Negative Preference Optimization (Zhang et al., 2024) and SimNPO (Fan et al., 2025). Representation-based methods such as RMU instead target internal representations associated with the unwanted knowledge (Li et al., 2024). These methods are commonly evaluated on benchmarks such as TOFU (Maini et al., 2024) and LUME (Ramakrishna et al., 2025), with OpenUnlearning providing consistent implementations and evaluation protocols (Dorna et al., 2025). While these methods depart from a final model, trained without unlearning in mind, our method supports unlearning by construction.

Unlearning by construction. Unlearning by construction prepares the model for future removal while the information is learned. SISA divides the training data into separate shards and retrains only the affected components after a removal request (Bourtoule et al., 2021). S3T applies a related approach to parameter-efficient fine-tuning by training adapters over a frozen backbone (Chowdhury et al., 2025). APA also uses separate adapters and combines their outputs through adaptive aggregation during inference (Hu et al., 2025). Memory Adapters assign memory entries to individual sources (Grimes et al., 2026), NULL uses source-dependent masks over memorization sinks (Ghosal et al., 2026; 2025), GRAM stores source information in separate modules (Roland et al., 2026), and SGTM restricts updates from selected data to predefined parameter regions (Shilov et al., 2025; Cloud et al., 2024). Related LLM architectures use separate experts or adapters to support flexible data access and source removal (Shi et al., 2025; Schneider et al., 2026; Zhao et al., 2024). Many of these modular approaches use source labels, retrieval, or learned routers to select source-specific components during inference. This makes deployment depend on a separate source-attribution mechanism that must remain available after training. Model merging removes the selection step by combining independently trained updates, but it can weaken the separation needed for unlearning (Kuo et al., 2025). MUSR instead retains separate source modules while including all of them in every forward pass, allowing their learned outputs to do *self-routing*, *i.e.,* to determine their contributions without an external router. MUSR therefore preserves module-level unlearning without requiring source labels, retrieval, or a learned router at inference.

3 Modular Unlearning via Self-Routing

MUSR includes four main parts: **Source-specific modules**, represented by one small network per source in parallel with selected MLP layers (referred to as *equipped layers*), which learn information specific to their assigned sources (Section 3.1). **Self-routing training** which gives each module the task loss on its assigned sources and an output penalty on the remaining sources, so that every input can pass through all modules and they learn when to contribute non-zero outputs (Section 3.2). **Continual addition**, which allows us to introduce a new source by training a new module while keeping the existing modules frozen (Section 3.3). **Local unlearning**, which can then be performed by zeroing out the output projection of any given module without changing the base model or retained modules (Section 3.4). Figure 1 summarizes training, inference, and unlearning.

3.1 SOURCE-SPECIFIC MODULES

Let $\mathbf{h}_\ell$ denote the input to MLP layer ℓ and let $\mathrm{MLP}_\ell(\mathbf{h}_\ell)$ denote the output of the corresponding base-model MLP. For every source D_k, we add a module $\phi_{k,\ell}$ in parallel to the base MLP, so that layer ℓ computes

$$\mathbf{y}_\ell = \mathrm{MLP}_\ell(\mathbf{h}_\ell) + \sum_{k=1}^{K} \phi_{k,\ell}(\mathbf{h}_\ell). \tag{1}$$

Every module participates in this computation for every input. The model therefore does not use any router, top-k selection, source-dependent mask, or source label to determine which module contributes. We place the modules at MLP layers because such feed-forward layers have been shown to store factual associations and are effective locations for knowledge editing (Geva et al., 2021; Meng et al., 2022). We ablate module placement in Appendix E.2.

Each of our source modules is a gated feed-forward network of width n with a gate projection, an up projection, and an output projection $W^{\mathrm{o}}_{k,\ell}$. The gate uses a ReLU activation, which allows the module to produce an exact zero contribution, and $W^{\mathrm{o}}_{k,\ell}$ is initialized to zero so that adding the modules does not change the initial output of the base model. Appendix B.1 gives the complete module equation and parameter dimensions.

The width n and the number of layers where we add such modules determine the parameters stored for each source, while total storage and dense inference computation grow linearly with the number of sources K. We analyze module width and placement in Appendix E and report storage and inference costs in Table 16.

3.2 SELF-ROUTING TRAINING

Training batches may contain samples from several sources, and we write k_b for the source of sample b. For this sample, we call ϕ_{k_b} the assigned module and the modules ϕ_j, $j \neq k_b$, the non-assigned modules. Although all modules contribute to the forward pass, the task loss sends gradients only to the assigned source module ϕ_{k_b}. The outputs of the non-assigned modules are treated as constants for this loss. This restriction determines where the task information is learned and prevents the same sample from influencing several source modules.

Restricting the task gradient does not determine where a module produces a large output on any given sample. A module trained on its assigned source may still contribute on inputs from other sources, which would make its later unlearning affect the behavior for retained sources. We therefore penalize the outputs of every non-assigned module through

$$\mathcal{L}_{\mathrm{out}} = \operatorname*{mean}_{b,t,\ell} \left[\sum_{j \neq k_b} \left\| \phi_{j,\ell}\left(\mathbf{h}_\ell^{(b,t)}\right) \right\|_2^2 \right]. \tag{2}$$

The mean is taken over samples, token positions, and all equipped layers, while the inner sum covers every module not assigned to the current sample. The penalty acts on the final module output and trains the whole non-assigned module to reduce its contribution on that input.

The task loss and output penalty have complementary roles. For a sample from source D_k, the task loss trains ϕ_k to represent the source, while the output penalty trains every ϕ_j with $j \neq k$ to remain close to zero. Each sample therefore acts as a positive example for one module and a negative example for all other modules. The complete objective therefore is

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + \lambda_{\mathrm{out}} \mathcal{L}_{\mathrm{out}}, \tag{3}$$

where λ_{out} balances learning the assigned source against suppressing non-assigned outputs. A small value of λ_{out} allows more interference between modules, while a large value enforces stronger separation but can limit how much each module learns. Figure 4 measures this trade-off.

We call the resulting behavior *self-routing* because the source label is used only to assign training signals. At inference, every module receives the hidden state at every equipped layer and token position, and the magnitude of its learned output determines its contribution. The model makes no separate router and does not receive any source labels.

3.3 CONTINUAL ADDITION

By its modular design MUSR supports continual post hoc additions of modules for new sources. A new source can be added without updating the existing source modules. To add D_{K+1}, we introduce ϕ_{K+1} and keep all existing modules frozen. The new module receives the task loss on data from D_{K+1} and the output penalty on data from the existing sources, so its parameters learn to contribute on the new source and remain close to zero on retained sources. Inference then uses the same computation in Equation (1) with the additional module. The parameters of the existing modules remain unchanged during continual addition. A source added in this way can later be unlearned with the same operation as a source present during the original training.

3.4 UNLEARNING

To unlearn source D_k, we set $W^{\circ}_{k,\ell}$ to zero in every equipped layer. The module then produces zero output for every input and can be removed from the forward pass. The base model and all retained source modules remain unchanged, and inference requires no block list, source label, or other serving-time record of the unlearning request. Repeated requests can also be composed because unlearning another source applies the same operation to a different module.

We note that this approach is not equivalent to *exact unlearning* because the content of D_k learned through the task loss is confined to the module of the source, but during training the retained modules received output-penalty gradients on samples from D_k, so they may still carry a statistical trace of that source having been present. We quantify what remains and show that it does not affect unlearning robustness in Appendix E.6.

3.5 TRAINING SETTINGS

MUSR applies to both fine-tuning and pretraining. During fine-tuning, the pretrained backbone remains frozen. For a sample from source D_k, the task loss updates only the assigned module ϕ_k, while the output penalty in Equation (2) updates all non-assigned modules. During pretraining, the backbone and source modules are initialized from scratch. Training uses shared core data and source-specific data. The task loss on shared core data updates the backbone. On data from source D_k, the backbone receives no gradient, the task loss updates ϕ_k, and the output penalty updates all non-assigned modules. Both settings use the same self-routing and unlearning mechanisms.

4 EXPERIMENTS

We evaluate MUSR in pretraining and fine-tuning settings across diverse benchmark datasets and models, measuring model performance before and both performance and unlearning success after unlearning the selected sources.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate diverse unlearning benchmark datasets for pretraining and fine-tuning. For **pretraining**, we follow the setup of Roland et al. (2026) and evaluate both realistic and synthetic setups. The synthetic setup leverages SimpleStories (Finke et al., 2025), where $K \in \{4, 8, 12, 16, 20\}$ of its 48 topics are unlearnable sources. For the realistic setup, we use around 4 billion tokens containing web text (Penedo et al., 2024), code (Kocetkov et al., 2023), and arXiv papers as core data, and additionally add four datasets from LISP code and papers on biology, nuclear science, and cybersecurity as unlearnable sources. For **fine-tuning**, we rely on TOFU (Maini et al., 2024) and LUME (Ramakrishna et al., 2025). TOFU contains $4{,}000$ question and answer samples about 200 fictitious authors. Each author contributes 20 samples and defines one source, giving $K = 200$ modules for MUSR. We use the *forget10* split as the forget set, which contains 20 authors and 400 samples. The remaining authors form *retain90*, while *holdout10* contains samples from 20 unseen authors used for the privacy evaluation. LUME contains $1{,}733$ documents across synthetic stories, synthetic biographies with private information, and real biographies. The benchmark provides 866 forget and 867 retain documents as sentence-completion and question-answering samples.

Models and training. For **pretraining**, we follow Roland et al. (2026) and train a Transformer with approximately 200 million parameters for the realistic setup and a 26 million parameter Transformer on SimpleStories. In both pretraining settings, the shared model and source modules are trained jointly from scratch. For fine-tuning, we adapt Llama-3.2-1B-Instruct and Llama-3.1-8B-Instruct (Dubey et al., 2024), together with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2025). Complete dataset-specific configurations and hyperparameters are provided in Appendix B.

Baselines. For **pretraining**, we compare MUSR with full-data pretraining, retraining without the selected source, GRAM (Roland et al., 2026), and NULL (Ghosal et al., 2026). GRAM and NULL receive the source label during inference, whereas MUSR sums all module outputs. For **fine-tuning**, we compare with the post-hoc methods GradDiff (Maini et al., 2024), NPO (Zhang et al., 2024), SimNPO (Fan et al., 2025), and RMU (Li et al., 2024), and with the by-construction methods S3T (Chowdhury et al., 2025), APA (Hu et al., 2025), GRAM (Roland et al., 2026), NULL (Ghosal et al., 2026), SGTM (Shilov et al., 2025), and Memory Adapters (Grimes et al., 2026). GRAM and NULL receive the exact source label during inference, while APA selects modules using embedding-based retrieval. We additionally report the frozen base model, standard joint fine-tuning, and retraining without the corresponding forget set as references. See Appendix B for more details.

Metrics. To evaluate **pretraining**, we report held-out token cross-entropy on the shared core and each unlearnable source to assess model performance. After unlearning source D_k, we compare its cross-entropy with retraining without D_k , which provides a reference for a model trained without that source. We also report the average change on the shared core and retained sources, where a lower value indicates less collateral damage from unlearning. For **fine-tuning**, we evaluate on general model utility and task performance on the retain and forget sets before and after unlearning.For comparisons after unlearning, we use a model retrained from scratch without the selected sources as the ”gold-standard” reference. Values closer to the original trained model indicate insufficient (or *under*) unlearning, while values below the frozen base model indicate excessive (or *over*) unlearning. On TOFU, model utility is the harmonic mean of answer probability, ROUGE-L recall, and truth ratio on retained authors, real authors, and world facts. We measure performance on forget10 using ROUGE-L recall. We measure privacy using the harmonic mean of 4 membership inference attacks (LOSS (Yeom et al., 2018), ZLib (Carlini et al., 2021), Min-k (Shi et al., 2024a), and Mink++ (Zhang et al., 2025)) following OpenUnlearning (Dorna et al., 2025) and center the mean around retraining. A value of zero matches retraining, while positive and negative values indicate that the forget set is more or less identifiable than under retraining, respectively. On LUME, we report exact-match accuracy on the retain and forget samples and five-shot MMLU (Hendrycks et al., 2021).

4.2 Pretraining Results

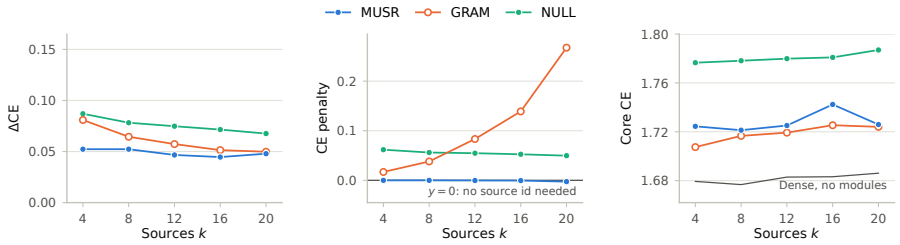
Realistic setting. Table 1 reports results before and after unlearning, averaged across the four source-specific datasets. Complete per-source results are reported in Appendix C. Before unlearning, MUSR closely matches full-data training on the shared core (2.346 against 2.338) and obtains the lowest cross-entropy on the source data among the by-construction methods (2.486, compared with 2.518 for GRAM and 2.569 for NULL). This means that our method achieves the best learning performance on the sources among the baselines and closely matches standard pretraining.

After unlearning, we observe that MUSR yields the highest cross-entropy on the unlearned sources, showing the strongest unlearning effect among the compared methods. The difference from the gold-standard baseline of retraining the model from scratch without the assigned sources (2.709) mainly results from LISP, for which cross-entropy increases by 2.07, while the increases for biology, nuclear science, and cybersecurity range from 0.21 to 0.32. Regarding the retain performance, our method closely matches the GRAM and NULL baselines. GRAM and NULL obtain a near-zero change because retained inputs continue to select unchanged components using their source labels. In contrast, MUSR uses the same set of active modules for every input, so its retained change measures the effect visible during router-free inference. Still, overall, these results show that unlearning with our MUSR remains local to the selected source.

Synthetic setting. We leverage the synthetic setting to analyze how performance changes as the number of unlearnable sources increases from $K{=}4$ to $K{=}20$. Figure 2 reports core cross-entropy on the shared data, the increase in source cross-entropy after unlearning, and router cost. We observe that the core cross-entropy of MUSR remains between 1.721 and 1.742 across the evaluated

Table 1: **Pretraining results averaged across four unlearnable sources.** We report cross-entropy before and after unlearning. Retained Δ is the average change on the shared core and retained sources. We use the corresponding retrained model as the reference on the unlearned source. †Uses the source label during inference. Results are mean $\pm$ SD over three runs. Entries marked with – are not applicable, since the full-data model is not unlearned and retraining has no state before unlearning.

	Before unlearning		After unlearning	
Method	**Core ↓**	**Sources ↓**	**Unlearned source**	**Retained Δ ↓**
Reference				
Full data	2.338 ± 0.023	2.502 ± 0.043	–	–
Retrain without D_k	–	–	2.709 ± 0.012	0.000 ± 0.001
By construction				
GRAM†	2.350 ± 0.022	2.518 ± 0.031	2.700 ± 0.014	**0.000** ± 0.002
NULL†	2.413 ± 0.029	2.569 ± 0.019	2.861 ± 0.020	**0.000** ± 0.001
MUSR (ours)	**2.346** ± 0.017	**2.486** ± 0.012	**3.002** ± 0.009	0.003 ± 0.001



(a) Deletion: ΔCE on deleted source (b) Router cost: CE without source id (c) Deployed core CE

Figure 2: **Scaling with the number of unlearnable sources** K in the synthetic SimpleStories setting. We report the increase in source cross-entropy after unlearning (a), the router cost (b), and held-out core cross-entropy (c). Router cost is the increase in source cross-entropy when inference does not receive the correct source label. Results are mean $\pm$ SD over three runs.

configurations. The increase in cross-entropy on the unlearned source also remains stable, ranging from 0.025 to 0.032. Router cost is the increase in source cross-entropy when inference does not receive the correct source label. It remains zero for MUSR across all values of K. In contrast, the router cost of GRAM increases from 0.017 to 0.268, while NULL remains between 0.050 and 0.062. This means that the absence of a source label becomes increasingly costly for GRAM as the number of sources grows, while MUSR does not depend on this information.

4.3 Fine-Tuning Results

Our MUSR. We show the per-model results on TOFU in Figure 3. Across the four models, the mean utility of MUSR changes from 0.588 before unlearning to 0.586 afterward, corresponding to 99.7% retention of its pre-unlearning utility. Before unlearning, its ROUGE-L score on the training data ranges from 0.966 to 0.988 across the four models. This shows that the source modules learn the assigned information while preserving general model utility. Analyzing the models *after unlearning*, we observe that the model utility on retained data across all models is higher than for all baseline by-construction unlearning methods (Figure 3(b)), close to 1.00. This means that our MUSR has a negligible effect on the model performance for the retained data. For the unlearned sources, we also observe that our MUSR most closely matches the performance of the base models which have never seen the unlearned sources (Figure 3(c)). This highlights that our method does neither over- or under unlearn the data. This is also reflected in the privacy results for the unlearned sources in Figure 3(d). A score close to zero indicates privacy behavior similar to retraining, while scores closer to -100 and 100 indicate that the forget set is less or more identifiable than under retraining, respectively.

Baseline comparison. GRAM receives the exact source label during inference, so queries about an unlearned author no longer activate its assigned source module. Despite this oracle information, its forget-set ROUGE differs from the frozen base by up to 0.498 across the four models. In contrast, MUSR receives no source label and remains within 0.016 of the base forget-set ROUGE on every model, while preserving higher utility than every baseline that uses external selection. APA reaches

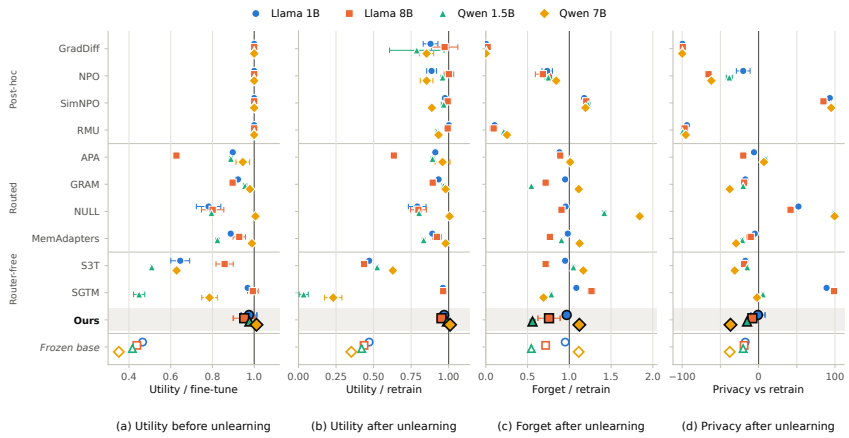


Figure 3: **TOFU results across four language models.** Utility before unlearning is normalized by standard fine-tuning utility, while utility, forget, and privacy after unlearning are reported relative to retraining without the forget set. Hollow markers show the frozen base model. Methods are grouped into post-hoc methods, by-construction methods with external source-dependent selection, and router-free by-construction methods. Post-hoc methods share the standard fine-tuned checkpoint, so their normalized utility before unlearning equals one.

a forget-set ROUGE close to retraining, but loses utility, most strongly on Llama-3.1-8B-Instruct with 0.400 against 0.630 for retraining, and requires embedding-based retrieval during inference. Among the post-hoc methods, SimNPO provides the strongest consistent performance, as it preserves utility well across the four backbones, but its forget-set ROUGE remains 0.067–0.083 above the corresponding retraining result, indicating incomplete unlearning. Still, it requires a new optimization run for every unlearning request. SGTM is competitive on Llama-3.2-1B-Instruct, but assumes that the forget set is known during training (which is usually not the case in practice) and loses utility after unlearning on both Qwen models. Overall, MUSR applies the same forward computation to every input and provides the most consistent router-free combination of source fidelity, retained utility, and local unlearning. Full results for each model are reported in Tables 8 to 11.

The LUME results in Tables 12 and 13, reported in Appendix D.2 show similar trends.

4.4 Ablations

We next study the effect of the output penalty and the robustness of MUSR to different source assignments. These experiments use Llama-3.2-1B-Instruct on TOFU.

Self-routing and output penalty. We first analyze the effect of the output penalty $\mathcal{L}_{\text{out}}$ on unlearning. For this, we vary its weight λ_{out}, where higher values penalize the modules more for contributing to sources that are not assigned to it. Figure 4 shows first that the output penalty is necessary for our method's success. Small values of λ_{out} lead to weak separation between the modules and non-successful self-routing. Second, the results show that MUSR maintains good utility and unlearning behavior under a wide range of λ_{out}. Our default setting of $\lambda_{\text{out}} = 10$ provides clear module separation while preserving utility and forget performance close to retraining.

Source module assignment. We analyze how many sources can be stored in one module by assigning multiple TOFU authors to each module. Our results in Table 2 show that MUSR remains effective when each module stores multiple authors, reaching a utility of 0.565 and recall of 1 at $G{=}50$.

Next, we distribute individual samples randomly across modules, such that samples from the same author may be assigned to different modules. At $G{=}1$, each module still receives 20 samples, matching the capacity used when one complete author is assigned to each module. MUSR reaches a utility of 0.565 and a recall of 0.910, compared with 0.414 and 0.427 for GRAM and 0.322 and 0.356 for APA. Since MUSR evaluates all modules together, it can combine information distributed across them, in contrast to the baselines. GRAM evaluates only the module given by the oracle assignment, while APA weights modules by embedding similarity. Neither recovers the fragmented information as well as MUSR.

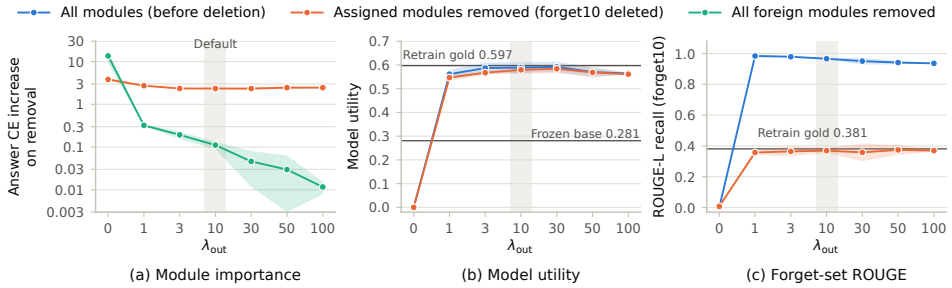


Figure 4: **Effect of the output-penalty weight λ_{out} on MUSR using Llama-3.2-1B-Instruct and TOFU.** Panel (a) measures module importance for 20 sampled authors on their own samples. Importance is the increase in cross-entropy loss after removing either the assigned modules or all non-assigned modules. Panels (b) and (c) report model utility and forget ROUGE before and after unlearning the authors in forget10. The shaded region marks the default value $\lambda_{\text{out}} = 10$.

Table 2: **Capacity of source modules on Llama-3.2-1B-Instruct and TOFU.** G denotes the number of authors per module; sample-based assignments use the same number of samples per module as the corresponding author-based setting. We further report the ROUGE-L recall on the source samples before unlearning. Further results are reported in Appendix E.

	$G = 1$		$G = 50$	
Assignment	**Utility**	**ROUGE-L recall**	**Utility**	**ROUGE-L recall**
MUSR (router-free)				
Author	0.589 ± 0.022	0.966 ± 0.003	0.565 ± 0.006	1.000 ± 0.000
Random samples	0.565 ± 0.008	0.910 ± 0.006	0.519 ± 0.014	0.999 ± 0.001
GRAM (oracle router)				
Author	0.559 ± 0.002	1.000 ± 0.000	0.276 ± 0.055	0.994 ± 0.004
Random samples	0.414 ± 0.002	0.427 ± 0.002	0.251 ± 0.059	0.605 ± 0.004
APA (embedding router)				
Author	0.542 ± 0.005	0.799 ± 0.010	0.475 ± 0.011	0.740 ± 0.018
Random samples	0.322 ± 0.001	0.356 ± 0.003	0.337 ± 0.007	0.370 ± 0.006

In Appendix E, we provide additional ablations. Concretely, Appendix E.1 confirms that **self-routing** is not tied to the exact source data, since the separation between assigned and non-assigned modules persists on paraphrased questions that the modules never saw. On **module placement and type** (Appendix E.2), our gated MLP modules on the earlier layers clearly outperform LoRA at the same parameter budget, while equipping all layers only doubles the storage without improving the results. Regarding **module width** (Appendix E.3), we find that values above our default of 16 bring no consistent gain despite their higher storage. We further find that **continual addition** of new sources leaves utility and the previously stored sources nearly unchanged (Appendix E.4), and that **repeated unlearning** requests do not significantly affect performance, whereas the sequential post-hoc baselines either collapse or degrade utility significantly (Appendix E.5). Finally, we show that MUSR is also **cheaper at inference** than the routed baselines, since evaluating all modules costs less than selecting among them (Appendix F).

5 Conclusion

We proposed MUSR, an unlearning by construction method that does not require any external components, such as source labels or a router. MUSR learns to self-route by training each module to contribute to its assigned source while suppressing its output on non-assigned sources. All modules can therefore be evaluated together, while unlearning a source only requires zeroing the output projection of its module. Across pretraining and fine-tuning settings, MUSR successfully learns all sources and matches the unlearning performance of modular baselines with external components while preserving strong retain utility. MUSR also supports repeated unlearning and continual source addition. Overall, MUSR contributes to making unlearning by construction more practical.

AI USE STATEMENT

In this work, we used generative AI tools to refine hypotheses, provide feedback on research methodology and experimental design, assist with method implementation, and support the interpretation of experimental results. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, support or write mathematical proofs, assist with translation, or clean and reformat datasets. Qualitative and thematic data analysis was not applicable to this work. We additionally used generative AI tools to suggest experimental parameters, create and edit software code and improve the structure and readability of the paper. We reviewed and verified all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Vincent-Pierre Berges, Barlas Oguz, Daniel Haziza, Wen tau Yih, Luke Zettlemoyer, and Gargi Ghosh. Memory layers at scale. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ATqGm1WyDj.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings - 2021 IEEE Symposium on Security and Privacy, SP 2021*, Proceedings - IEEE Symposium on Security and Privacy, pp. 141–159, United States, May 2021. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/SP40001.2021.00019.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1), January 2023. ISSN 1532-4435.

Somnath Basu Roy Chowdhury, Krzysztof Marcin Choromanski, Arijit Sehanobish, Kumar Avinava Dubey, and Snigdha Chaturvedi. Towards scalable exact machine unlearning using parameter-efficient fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oe51Q5Uo37.

Alex Cloud, Jacob Goldman-Wetzler, Even Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks. *ArXiv*, abs/2410.04332, 2024.

Vineeth Dorna, Anmol Reddy Mekala, Wenlong Zhao, Andrew McCallum, J Zico Kolter, Zachary Chase Lipton, and Pratyush Maini. Openunlearning: Accelerating LLM unlearning via unified benchmarking of methods and metrics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=Gy67Zh5X1i.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=JbvSQm5h1l.

Lennart Finke, Chandan Sreedhara, Thomas Dooms, Mat Allen, Juan Rodriguez, Noa Nabeshima, Thomas Marshall, and Dan Braun. Parameterized synthetic text generation with simplestories. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-1234. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/34696303f5996d5c54e699c0b462ae48-Paper-Datasets_and_Benchmarks_Track.pdf.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446/.

Gaurav R. Ghosal, Pratyush Maini, and Aditi Raghunathan. Natively unlearnable large language models, 2026. URL https://arxiv.org/abs/2606.13873.

Gaurav Rohit Ghosal, Pratyush Maini, and Aditi Raghunathan. Memorization sinks: Isolating memorization during LLM training. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sRJrMPu5Uu.

Keltin Grimes, Kevin Kuo, Zhiwei Steven Wu, Virginia Smith, and Marissa Connor. Memory adapters enable fast, flexible knowledge unlearning in LLMs. In *ICML 2026 Workshop on the Impact of Memorization on Trustworthy Foundation Models*, 2026.

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. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiyu Hu, Yang Zhang, Minghao Xiao, Wenjie Wang, Fuli Feng, and Xiangnan He. Exact and Efficient Unlearning for Large Language Model-Based Recommendation . *IEEE Transactions on Knowledge & Data Engineering*, 37(10):5866–5877, October 2025. ISSN 1558-2191. doi: 10.1109/TKDE.2025.3594687. URL https://doi.ieeecomputersociety.org/10.1109/TKDE.2025.3594687.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge unlearning for mitigating privacy risks in language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14389–14408, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.805. URL https://aclanthology.org/2023.acl-long.805/.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. The stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=pxpbTdUEpD.

Kevin Kuo, Amrith Setlur, Kartik Srinivas, Aditi Raghunathan, and Virginia Smith. Exact unlearning of finetuning data via model merging at scale. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=u89LDBIyDe.

Guillaume Lample, Alexandre Sablayrolles, Marc' Aurelio Ranzato, Ludovic Denoyer, and Herve Jegou. Large memory layers with product keys. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/9d8df73a3cfbf3c5b47bc9b50f214aff-Paper.pdf.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D. Li, Ann-Kathrin Dombrowski, Shashwat Goel, Gabriel Mukobi, Nathan Helm-Burger, Rassin Lababidi, Lennart Justen, Andrew Bo Liu, Michael Chen, Isabelle Barrass, Oliver Zhang, Xiaoyuan Zhu, Rishub Tamirisa, Bhrugu Bharathi, Ariel Herbert-Voss, Cort B Breuer, Andy Zou, Mantas Mazeika, Zifan Wang, Palash Oswal, Weiran Lin, Adam Alfred Hunt, Justin Tienken-Harder, Kevin Y. Shih, Kemper Talley, John Guan, Ian Steneker, David Campbell, Brad Jokubaitis, Steven Basart, Stephen Fitz, Ponnurangam Kumaraguru, Kallol Krishna Karmakar, Uday Tupakula, Vijay Varadharajan, Yan Shoshitaishvili, Jimmy Ba, Kevin M. Esvelt, Alexandr Wang, and Dan Hendrycks. The WMDP benchmark: Measuring and reducing malicious use with unlearning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=xlr6AUDuJz.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary Chase Lipton, and J Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=B41hNBoWLo.

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

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Anil Ramakrishna, Yixin Wan, Xiaomeng Jin, Kai-Wei Chang, Zhiqi Bu, Bhanukiran Vinzamuri, Volkan Cevher, Mingyi Hong, and Rahul Gupta. LUME: LLM unlearning with multitask evaluations. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6524–6535, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.347. URL https://aclanthology.org/2025.findings-emnlp.347/.

Ethan Roland, Murat Cubuktepe, Erick Martinez, Stijn Servaes, Keenan Pepper, Michael Vaiana, Diogo S de Lucena, Judd Rosenblatt, Addie Foote, Cem Anil, and Alex Cloud. Modular pretraining enables access control. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=yIubI9l3IT.

Chris Schneider, Philipp Schoenegger, and Ben Bariach. Separable expert architecture: Toward privacy-preserving llm personalization via composable adapters and deletable user proxies, 2026. URL https://arxiv.org/abs/2604.21571.

Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=zWqr3MQuNs.

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, 2024b. URL https://arxiv.org/abs/2407.06460.

Weijia Shi, Akshita Bhagia, Kevin Farhat, Niklas Muennighoff, Jacob Morrison, Evan Pete Walsh, Dustin Schwenk, Shayne Longpre, Jake Poznanski, Allyson Ettinger, Daogao Liu, Margaret Li, Mike Lewis, Wen tau Yih, Dirk Groeneveld, Luca Soldaini, Kyle Lo, Noah A. Smith, Luke Zettlemoyer, Pang Wei Koh, Hannaneh Hajishirzi, Ali Farhadi, and Sewon Min. FlexOLMo: Open language models for flexible data use. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1rUj9ZN6Bz.

Igor Shilov, Alex Cloud, Aryo Pradipta Gema, Jacob Goldman-Wetzler, Nina Panickssery, Henry Sleight, Erik Jones, and Cem Anil. Beyond data filtering: Knowledge localization for capability removal in llms, 2025. URL https://arxiv.org/abs/2512.05648.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pp. 268–282. IEEE, 2018.

Jingyang Zhang, Jingwei Sun, Eric Yeats, Yang Ouyang, Martin Kuo, Jianyi Zhang, Hao Frank Yang, and Hai Li. Min-k%++: Improved baseline for pre-training data detection from large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ZGkfoufDaU.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=MXLBXjQkmb.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Yuwei Hu, Tao Shen, Hongxia Yang, Kun Kuang, and Fei Wu. Retrieval-augmented mixture of lora experts for uploadable machine learning, 2024. URL https://arxiv.org/abs/2406.16989.

A BASELINE METHODS

This section summarizes how each baseline learns and unlearns source information. We focus on the mechanism and the information required during inference. The exact configurations and hyperparameters are provided in Appendix B.

A.1 REFERENCE MODELS

Frozen base. The frozen base model receives no task-specific training. On TOFU, it provides a source-free reference because the fictitious authors are not present during pretraining.

Joint fine-tuning. Joint fine-tuning trains one model on all sources without preparing for later unlearning. It provides the standard trained reference before unlearning and serves as the initial checkpoint for all post-hoc methods.

Retraining. Retraining trains the model without the selected sources and serves as the empirical reference after unlearning. Unlike the evaluated unlearning methods, it never observes the forget data during training.

A.2 POST-HOC UNLEARNING

Post-hoc methods begin from the jointly fine-tuned model and optimize a new objective after an unlearning request arrives. They update parameters shared by forget and retain data and therefore perform a new optimization procedure for every request.

GradDiff. Gradient difference applies gradient ascent to the forget loss and gradient descent to the retain loss (Maini et al., 2024). The forget term reduces the likelihood of the selected data, while the retain term limits changes to the remaining model behavior.

NPO. Negative Preference Optimization treats forget responses as rejected responses and reduces their likelihood relative to the model before unlearning (Zhang et al., 2024). The pre-unlearning model serves as a fixed reference, while retain data can be used to preserve unrelated behavior.

SimNPO. SimNPO removes the reference-model dependence of NPO and instead applies a reference-free preference objective to the forget responses (Fan et al., 2025). This avoids making the unlearning signal depend on how strongly the original model fitted each forget example.

RMU. Representation Misdirection for Unlearning modifies internal activations rather than directly optimizing output likelihoods (Li et al., 2024). It steers forget-data representations toward a fixed random target while constraining retain-data representations to remain close to those of the original model.

A.3 UNLEARNING BY CONSTRUCTION

Unlearning by construction prepares for future unlearning requests during training. These methods isolate data in shards, modules, memory entries, or parameter regions so that unlearning can modify only the corresponding component. They differ in how this component is selected and whether inference requires a source label, retrieval, or no external selection.

S^3T. Sequence-aware Sharded Sliced Training adapts SISA to parameter-efficient language-model fine-tuning (Bourtoule et al., 2021; Chowdhury et al., 2025). It divides the training data into ordered shards and trains a sequence of adapters over a frozen backbone. When data assigned to one shard must be unlearned, the affected adapter and the adapters trained after it are retrained without that data.

APA. Adapter Partition and Aggregation assigns each data partition to a separate adapter (Hu et al., 2025). An unlearning request removes or retrains only the adapters whose partitions contain the selected data. During inference, APA uses embedding-based retrieval to compute an input-dependent weighting over the adapters.

Memory Adapters. Memory Adapters place a product-key memory layer on a frozen backbone and assign each source a disjoint set of memory entries (Lample et al., 2019; Berges et al., 2025; Grimes et al., 2026). Inference retrieves a small set of entries according to their keys. Unlearning blacklists the entries assigned to the selected source so that they cannot be returned by the retrieval procedure, although their stored value vectors remain in the model parameters.

GRAM. GRAM attaches source-specific auxiliary modules to the Transformer blocks and restricts each source's task gradients to its assigned modules (Roland et al., 2026). During inference, the exact source label determines which modules are active. Unlearning removes the modules assigned to the selected source while leaving the shared model and all other source modules unchanged.

NULL. NULL extends memorization sinks from individual sequences to concepts or sources (Ghosal et al., 2026; 2025). A shared model is trained together with a pool of sink neurons, and each source receives a sparse mask over this pool. The exact source label is required during inference to construct the corresponding mask, and unlearning disables the contribution associated with the selected source.

SGTM. SGTM uses hard gradient masks to restrict updates from selected data to predefined parameter regions (Shilov et al., 2025; Cloud et al., 2024). We follow the original paper, where one region is assigned to the complete future forget set and another to the retained data, so the forget set must be known during training. Unlearning edits the forget region, while inference continues through the standard model without a source label or external selector.

B Experimental Details

B.1 Module Implementation

Let d denote the hidden dimension of the base model and let n denote the width of each source module. At every chosen layer ℓ, the modules receive the same input $\mathbf{h}_\ell \in \mathbb{R}^d$ as the base model MLP. Their outputs are added to the MLP output as defined in Equation (1). The module assigned to source D_k computes

$$\phi_{k,\ell}(\mathbf{h}_\ell) = W^{\mathrm{o}}_{k,\ell}\left(\mathrm{ReLU}\left(W^{\mathrm{g}}_{k,\ell}\mathbf{h}_\ell + \mathbf{b}^{\mathrm{g}}_{k,\ell}\right) \odot W^{\mathrm{u}}_{k,\ell}\mathbf{h}_\ell\right), \tag{4}$$

where $W^{\mathrm{g}}_{k,\ell}, W^{\mathrm{u}}_{k,\ell} \in \mathbb{R}^{n\times d}$ are the gate and up projections, $\mathbf{b}^{\mathrm{g}}_{k,\ell} \in \mathbb{R}^n$ is the gate bias, and $W^{\mathrm{o}}_{k,\ell} \in \mathbb{R}^{d\times n}$ is the output projection. Each module therefore adds $3nd + n$ parameters per equipped layer. With our default of $n = 16$ and layers 0–7 for Llama-3.2-1B-Instruct ($d = 2048$), this gives 0.79 million parameters per source. We randomly initialize the gate and up projections, set the gate bias to 0.5, and initialize the output projection to zero. The modules therefore do not change the output of the base model before training.

Training. For a sample from D_k, the assigned module $\phi_{k,\ell}$ is evaluated normally, while the outputs of all non-assigned modules are treated as constants during backpropagation. The forward pass therefore includes every module, but only $\phi_{k,\ell}$ receives gradients from the task loss. The output penalty in Equation (2) is applied to every non-padding token in the sample. At a given layer, the output penalty directly updates the non-assigned modules included in its inner sum. The hidden state of a later layer also depends on the assigned modules at earlier layers. The penalty can therefore send an indirect gradient to assigned modules at earlier equipped layers. The task loss and output penalty are combined in Equation (3).

Table 3: **Pretraining configurations.** Both settings use decoder-only Transformers with grouped-query attention, RoPE, and RMSNorm, trained in bfloat16 with AdamW ($\beta = (0.9, 0.95)$, weight decay 0.01), gradient clipping at 1.0, and a schedule with linear warmup, a constant phase, and linear decay over the last 10% of steps.

		Realistic	**Synthetic**
Model	Parameters	200M, 13 layers, $d = 896$	26M, 8 layers, $d = 512$
	Context	1,024 tokens	256 tokens
	Tokenizer	GPT-2 BPE (50,257 tokens)	SimpleStories (4,096 tokens)
	Data	4B tokens, $K = 4$ sources	548M tokens, $K \in \{4, \ldots, 20\}$
Training	Steps	18,000–18,700	17,200–17,800
	Batch	221,184 tokens	30,720 tokens
	Learning rate	8.7×10^{-4}, 2% warmup	5×10^{-3} (10^{-3} for NULL), 2% warmup
MUSR	Modules	one MLP of width 384 per source in all layers	one MLP of width 192 per source in all layers
	λ_{out}	3	1
GRAM	Modules	one MLP of width 384 per source in all layers	one MLP of width 192 per source in all layers
	Core on source data	$p_{\text{as}} = 0.5$, $p_{\text{cr}} = 0.2$	$p_{\text{as}} = 0.2$, $p_{\text{cr}} = 0.5$
NULL	Sinks	pool of 2,688 units, 806 per source	pool of 1,536 units, 192 per source

Table 4: **Fine-tuning configurations for MUSR.**

Dataset	**Model**	K	n	**Rows/module**	**Layers**	λ_{out}	**Steps**
TOFU	Llama-3.2-1B-Instruct	200	16	20	0–7	10	16,000
	Llama-3.1-8B-Instruct	200	16	20	0–15	10	16,000
	Qwen2.5-1.5B-Instruct	200	16	20	0–7	10	16,000
	Qwen2.5-7B-Instruct	200	16	20	0–13	10	16,000
LUME	Llama-3.2-1B-Instruct	44	128	128	0–7	10	40,000
	Qwen2.5-1.5B-Instruct	180	32	32	0–7	10	40,000

B.2 Pretraining Configurations

Table 3 shows the hyperparameters of the realistic and synthetic pretraining settings. We follow the configurations of (Roland et al., 2026) for all methods, so that MUSR, GRAM, and NULL use the same model sizes, data, and optimization settings.

B.3 Fine-Tuning Configurations

MUSR. Table 4 reports the configuration for each dataset and model. On TOFU, each author defines one module. On LUME, we construct modules separately within each task and within the forget and retain partitions, so that a module never contains documents from different tasks or partitions, and group complete documents to match the number of modules in Table 4. During inference, every module is evaluated as in Equation (1).

Reference and post-hoc configurations. Joint fine-tuning and retraining update all parameters of the base model. Each post-hoc method begins from the jointly fine-tuned checkpoint and updates the complete model, except RMU, which only updates the layers up to the steered layer. Table 5 reports the shared optimization settings and the objective-specific hyperparameters. Effective batch size includes gradient accumulation and distributed training.

Unlearning by construction. The mechanisms of the by-construction baselines are described in Appendix A. All methods keep the base model frozen during fine-tuning except SGTM and NULL, which fine-tune the complete model. Table 6 reports the parameters that determine the capacity, training procedure, and inference behavior of each method.

Table 5: **Training hyperparameters for the reference and post-hoc methods.** The reference methods are joint fine-tuning and retraining, while the post-hoc methods start from the jointly fine-tuned model. γ and α weight the forget and retain losses. Batch sizes are effective batch sizes that include gradient accumulation and distributed training. Separate values are reported for Llama and Qwen where they differ.

		TOFU	LUME
Reference	Learning rate	1×10^{-5}	5×10^{-5}
	Epochs	5	20
	Batch size	16 / 32 (1.5B), 128 (7B)	8
	Optimizer	paged AdamW	AdamW
	Schedule	linear, 1 warmup epoch	cosine, 10% warmup
Post-hoc	Learning rate	1×10^{-5}	1×10^{-5}
	Epochs	10	10
	Batch size	16 / 32	32
	Schedule	linear, 1 warmup epoch	constant
GradDiff		$\gamma = 1, \alpha = 1$	$\gamma = 1, \alpha = 1$
NPO		$\beta = 0.1, \gamma = \alpha = 1$	$\beta = 0.1, \gamma = \alpha = 1$
SimNPO		$\beta = 4.5, \delta = 0, \gamma = 0.125, \alpha = 1$	$\beta = 2.5, \delta = 0, \gamma = \alpha = 1$
RMU		layer 7, coefficient 2	layer $\lfloor L/4 \rfloor$, coefficient 20

Table 6: **Training hyperparameters of the by-construction baselines.** Pairs separated by a slash refer to the smaller (1B, 1.5B) and the larger (7B, 8B) models.

		TOFU	LUME
GRAM	Modules	$K = 200, n = 16$	$K = 180, n = 32$
	Training	16,000 steps, batch 8	40,000 steps, batch 8
	Optimizer	2×10^{-4}, cosine, clip 0.5	2×10^{-4}, cosine, clip 0.5
NULL	Sinks	3200 units per layer, 38 per source	3200 units per layer, 38 per source
	Training	16,000 steps, batch 8	16,000 steps, batch 8
	Optimizer	2×10^{-4}, cosine	2×10^{-4}, cosine
SGTM	Forget region	12.5% of MLP neurons	12.5% of MLP neurons
	Training	1,000 / 2,250 steps, batch 32 / 16	16,000 steps, batch 8
	Optimizer	1×10^{-5}, 126 / 250 warmup steps	5×10^{-5}, cosine, 10% warmup
Memory Adapters	Memory	1024^2 / 2048^2 entries, top-32 / top-64, middle layer	2048^2 entries, top-64, middle layer
	Training	3,000 / 6,000 steps, batch 32 / 16	20,000 steps, batch 32
	Optimizer	1×10^{-2}, 125 / 250 warmup steps	1×10^{-2}, 170 warmup steps
S^3T	Adapters	1 shard, one slice per layer, LoRA $r = 32$	1 shard, one slice per layer, LoRA $r = 32$
	Training	16,000 steps, batch 8	40,000 steps, batch 8
	Optimizer	2×10^{-4}, cosine	2×10^{-4}, cosine
APA	Adapters	$K = 200$, LoRA $r = 16$	$K = 200$, LoRA $r = 16$
	Training	16,000 steps, batch 8	16,000 steps, batch 8

B.4 Evaluation Details

Pretraining. We report held-out token cross-entropy on the test split of the shared core and of each unlearnable source. After unlearning source D_k, we compare the cross-entropy on D_k with the model retrained without D_k. The retained change is the average change in cross-entropy on the shared core and the retained sources relative to the model before unlearning, averaged over all choices of D_k. In the synthetic setting, the router cost is the increase in cross-entropy on the source data when the model is evaluated without the source label, compared with evaluation using the correct source label.

TOFU. We evaluate with the OpenUnlearning framework (Dorna et al., 2025). Model utility is the harmonic mean of nine scores, namely the answer probability, the ROUGE-L recall of the generated answer, and the truth ratio, each measured on the retained authors, on real authors, and on world

Table 7: **Change in cross-entropy after unlearning one source in the realistic pretraining setting.** Rows are the unlearned sources and columns the evaluated data. Each cell reports the change in held-out cross-entropy relative to the same model before unlearning, and underlined entries mark the unlearned source. All methods are evaluated with the model after unlearning on every domain.

Module deleted	Core	LISP	Biology	Nuclear	Cyber
GRAM (routed at inference)					
LISP	+0.0072	**+0.5042**	+0.0062	+0.0067	+0.0052
Biology	+0.0083	+0.0026	+0.0752	+0.0060	+0.0035
Nuclear	+0.0071	+0.0024	+0.0036	+0.0870	+0.0058
Cyber	+0.0104	+0.0020	+0.0035	+0.0048	+0.0601
NULL (routed at inference)					
LISP	+0.2284	+0.4558	+0.1560	+0.1325	+0.1116
Biology	+0.2368	+0.2165	+0.2843	+0.1668	+0.1329
Nuclear	+0.2512	+0.2165	+0.1993	+0.2337	+0.1439
Cyber	+0.2534	+0.2250	+0.1810	+0.1670	+0.1943
MUSR (ours, router-free)					
LISP	+0.0007	**+2.0726**	+0.0011	+0.0004	+0.0003
Biology	+0.0017	+0.0000	+0.3194	+0.0062	+0.0035
Nuclear	+0.0111	+0.0011	+0.0070	+0.2620	+0.0155
Cyber	+0.0196	+0.0007	+0.0040	+0.0071	+0.2072

Darker cells indicate a larger change in cross-entropy.

facts. The forget score is the ROUGE-L recall of greedy generations against the reference answers on forget10.

Privacy. We use the privacy evaluation of OpenUnlearning, which applies 4 membership inference attacks (LOSS (Yeom et al., 2018), ZLib (Carlini et al., 2021), Min-k (Shi et al., 2024a), and Mink++ (Zhang et al., 2025)) to distinguish forget10 from the unseen authors in holdout10. The harmonic mean of these values is taken. Let h denote the resulting mean privacy of a model and h_{ret} the mean privacy of the model retrained without forget10. We report

$$P = \begin{cases} 100\,(h - h_{\text{ret}})/(1 - h_{\text{ret}}) & \text{if } h \geq h_{\text{ret}}, \\ 100\,(h - h_{\text{ret}})/h_{\text{ret}} & \text{otherwise}, \end{cases} \tag{5}$$

which lies between -100 and 100. A value of zero matches retraining, positive values indicate that the forget set remains more identifiable than after retraining, and negative values indicate that it is less identifiable, which corresponds to excessive unlearning.

LUME. We follow the evaluation setup of LUME (Ramakrishna et al., 2025). Here, Retain and Forget are exact-match accuracy on the question-answering samples of the retain and forget documents. We generate greedily with up to 16 new tokens and compare the first generated line with the reference answer after lowercasing, removing punctuation, and collapsing whitespace. The accuracy is averaged over all question-answering samples of the three LUME tasks. MMLU (Hendrycks et al., 2021) is evaluated with five-shot prompts from the development split across all 57 subjects, using log-likelihood scoring of the answer options, and we report the accuracy over all questions.

Unless stated otherwise, we report the mean and the sample standard deviation over runs with different seeds.

C Additional Pretraining Results

C.1 Per-Source Results in the Realistic Setting

Table 7 reports the change in cross-entropy for each unlearned source and each evaluated domain. Diagonal entries measure the unlearned source, and off-diagonal entries measure the shared core and

Table 8: **Full TOFU forget10 results for Llama-3.2-1B-Instruct**. Utility reports TOFU model utility, Forget reports ROUGE-L recall, and Privacy is centered around retraining. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over three runs. Entries without error bars correspond to one run, and entries marked – are not applicable.

	Before unlearning		After unlearning		
Method	**Utility ↑**	**Forget**	**Utility ↑**	**Forget**	**Privacy → 0**
Reference					
Frozen base	0.281	0.362	–	–	−17.44
Joint fine-tune	0.604 ± 0.006	0.959 ± 0.008	–	–	–
Retrain	–	–	0.597 ± 0.004	0.381 ± 0.001	0.00 ± 0.19
Post-hoc					
GradDiff	–	–	0.525 ± 0.029	0.001 ± 0.001	−99.69 ± 0.45
NPO	–	–	0.529 ± 0.019	0.280 ± 0.024	−20.11 ± 8.99
SimNPO	–	–	0.582 ± 0.001	0.449 ± 0.011	93.19 ± 0.87
RMU	–	–	0.598 ± 0.007	0.040 ± 0.008	−93.72 ± 1.96
By construction					
S3T	0.390 ± 0.027	0.912 ± 0.011	0.281 ± 0.000	0.362 ± 0.000	−17.44 ± 0.00
APA†	0.542 ± 0.005	0.799 ± 0.010	0.544 ± 0.005	0.336 ± 0.002	−6.01 ± 1.73
GRAM†	0.559 ± 0.001	1.000 ± 0.000	0.559 ± 0.001	0.362 ± 0.000	−17.44 ± 0.00
NULL†	0.472 ± 0.035	1.000 ± 0.000	0.472 ± 0.035	0.364 ± 0.003	52.14 ± 1.28
SGTM	0.585 ± 0.001	0.511 ± 0.002	0.574 ± 0.006	0.413 ± 0.001	88.96 ± 0.33
MemAdapters†	0.536 ± 0.004	0.675 ± 0.010	0.532 ± 0.004	0.374 ± 0.005	−4.96 ± 1.34
MUSR	0.589 ± 0.022	0.966 ± 0.003	0.579 ± 0.016	0.369 ± 0.016	−0.83 ± 9.27

the retained sources. For MUSR, the change on the shared core and the retained sources remains below 0.02 for every unlearned source. On the unlearned source, cross-entropy increases by 0.21 to 0.32 for biology, nuclear science, and cybersecurity, and by 2.07 for LISP, which differs most from the rest of the data. GRAM and NULL select components with the source label during inference. In their deployment, the retained sources continue to use unchanged components, so their change on retained data is zero. The off-diagonal entries in the table instead evaluate their model after unlearning without this selection, which leads to changes between 0.11 and 0.25 for NULL.

D Full Fine-Tuning Results

D.1 TOFU Results

MUSR across backbones. Tables 8 to 11 report the complete TOFU results. Before unlearning, the utility of MUSR remains within 0.030 of joint fine-tuning on the two Llama models and Qwen2.5-1.5B-Instruct, while slightly exceeding it on Qwen2.5-7B-Instruct. Its forget-set ROUGE ranges from 0.966 to 0.988, showing that the source modules learn the assigned information across all four backbones. After unlearning, its utility remains within 0.030 of retraining on every model.

The forget-set behavior of MUSR after unlearning is consistently close to the corresponding frozen base model. Its forget scores are 0.369, 0.299, 0.203, and 0.428, compared with frozen-base scores of 0.362, 0.283, 0.198, and 0.425. This is expected because unlearning removes the learned source contribution and exposes the unchanged backbone. The result does not always match retraining because the retrained model can learn information about the selected authors indirectly from the retained data. The privacy results show a similar distinction. MUSR is close to retraining on Llama-3.2-1B-Instruct, with a privacy gap of -0.83, while it moves closer to the frozen-base reference on the other models. This is most visible on Qwen2.5-7B-Instruct, where MUSR reaches -36.76 and the frozen base reaches -37.74.

Comparison with other methods. The post-hoc methods show a consistent trade-off between retaining utility and matching the unlearning references. GradDiff and RMU strongly reduce forget-set ROUGE, but their privacy scores remain close to -100, indicating substantially more removal than retraining. SimNPO preserves utility more effectively, but its positive privacy scores and forget-set

Table 9: **Full TOFU forget10 results for Llama-3.1-8B-Instruct.** Utility reports TOFU model utility, Forget reports ROUGE-L recall, and Privacy is centered around retraining. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over three runs. Entries without error bars correspond to one run, and entries marked – are not applicable.

	Before unlearning		After unlearning		
Method	**Utility ↑**	**Forget**	**Utility ↑**	**Forget**	**Privacy → 0**
Reference					
Frozen base	0.275	0.283	–	–	−18.97
Joint fine-tune	0.628 ± 0.008	0.997 ± 0.000	–	–	–
Retrain	–	–	0.630 ± 0.007	0.395 ± 0.010	0.00 ± 2.21
Post-hoc					
GradDiff	–	–	0.613 ± 0.055	0.009 ± 0.010	−99.21 ± 1.37
NPO	–	–	0.631 ± 0.019	0.270 ± 0.035	−65.53 ± 1.75
SimNPO	–	–	0.626 ± 0.012	0.475 ± 0.009	84.90 ± 2.55
RMU	–	–	0.626 ± 0.010	0.037 ± 0.017	−96.80 ± 3.43
By construction					
S3T	0.539 ± 0.025	0.808 ± 0.024	0.275 ± 0.000	0.283 ± 0.000	−18.97 ± 0.00
APA†	0.394 ± 0.006	0.356 ± 0.002	0.400 ± 0.001	0.352 ± 0.007	−20.04 ± 0.33
GRAM†	0.563 ± 0.001	0.998 ± 0.001	0.563 ± 0.001	0.283 ± 0.000	−18.97 ± 0.00
NULL†	0.503 ± 0.033	1.000 ± 0.000	0.503 ± 0.033	0.358 ± 0.004	41.82 ± 2.33
SGTM	0.624 ± 0.014	0.908 ± 0.007	0.607 ± 0.011	0.500 ± 0.010	98.68 ± 0.07
MemAdapters†	0.583 ± 0.017	0.921 ± 0.010	0.581 ± 0.017	0.303 ± 0.013	−10.30 ± 4.69
MUSR	0.598 ± 0.032	0.988 ± 0.003	0.600 ± 0.016	0.299 ± 0.052	−8.18 ± 5.66

Table 10: **Full TOFU forget10 results for Qwen2.5-1.5B-Instruct**. Utility reports TOFU model utility, Forget reports ROUGE-L recall, and Privacy is centered around retraining. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over three runs. Entries without error bars correspond to one run, and entries marked – are not applicable.

	Before unlearning		After unlearning		
Method	**Utility ↑**	**Forget**	**Utility ↑**	**Forget**	**Privacy → 0**
Reference					
Frozen base	0.220	0.198	–	–	−20.18
Joint fine-tune	0.528 ± 0.004	0.943 ± 0.009	–	–	–
Retrain	–	–	0.523 ± 0.005	0.363 ± 0.001	0.00 ± 1.43
Post-hoc					
GradDiff	–	–	0.412 ± 0.095	0.003 ± 0.005	−99.50 ± 0.46
NPO	–	–	0.502 ± 0.003	0.272 ± 0.014	−38.35 ± 3.67
SimNPO	–	–	0.506 ± 0.007	0.445 ± 0.009	94.45 ± 0.43
RMU	–	–	0.482 ± 0.002	0.078 ± 0.001	−99.44 ± 0.04
By construction					
S3T	0.269 ± 0.001	0.393 ± 0.005	0.274 ± 0.001	0.381 ± 0.005	−14.75 ± 0.11
APA†	0.469 ± 0.001	0.908 ± 0.009	0.467 ± 0.001	0.367 ± 0.003	9.56 ± 0.65
GRAM†	0.505 ± 0.002	1.000 ± 0.000	0.505 ± 0.002	0.198 ± 0.000	−20.17 ± 0.00
NULL†	0.420 ± 0.002	0.621 ± 0.004	0.420 ± 0.002	0.515 ± 0.006	97.77 ± 0.13
SGTM	0.237 ± 0.014	0.618 ± 0.025	0.018 ± 0.016	0.285 ± 0.005	5.77 ± 0.18
MemAdapters†	0.435 ± 0.002	0.462 ± 0.004	0.436 ± 0.002	0.329 ± 0.002	−21.16 ± 0.44
MUSR	0.516 ± 0.004	0.977 ± 0.000	0.517 ± 0.004	0.203 ± 0.010	−15.12 ± 0.86

ROUGE above retraining indicate insufficient unlearning. NPO lies between these outcomes but generally loses more utility than SimNPO.

Among the by-construction baselines, GRAM also returns forget behavior close to the frozen base, but requires the exact source label during inference and generally retains less utility than MUSR. APA is close to retraining on some models, especially Qwen2.5-7B-Instruct, but its performance varies substantially across backbones and depends on embedding-based retrieval. S^3T and SGTM either learn less of the source information before unlearning or lose substantial utility afterward. Memory

Table 11: **Full TOFU forget10 results for Qwen2.5-7B-Instruct.** Utility reports TOFU model utility, Forget reports ROUGE-L recall, and Privacy is centered around retraining. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over three runs. Entries without error bars correspond to one run, and entries marked – are not applicable.

	Before unlearning		After unlearning		
Method	**Utility ↑**	**Forget**	**Utility ↑**	**Forget**	**Privacy → 0**
Reference					
Frozen base	0.225	0.425	–	–	−37.74
Joint fine-tune	0.640 ± 0.018	0.957 ± 0.005	–	–	–
Retrain	–	–	0.640 ± 0.007	0.382 ± 0.002	0.00 ± 1.05
Post-hoc					
GradDiff	–	–	0.546 ± 0.030	0.000 ± 0.001	−99.95 ± 0.07
NPO	–	–	0.546 ± 0.027	0.322 ± 0.015	−61.86 ± 3.43
SimNPO	–	–	0.568 ± 0.005	0.456 ± 0.007	95.04 ± 0.50
RMU	–	–	0.597 ± 0.004	0.097 ± 0.006	−95.31 ± 0.33
By construction					
S3T	0.402 ± 0.001	0.452 ± 0.001	0.402 ± 0.001	0.446 ± 0.002	−31.24 ± 0.33
APA†	0.605 ± 0.021	0.921 ± 0.006	0.614 ± 0.032	0.386 ± 0.001	6.97 ± 1.05
GRAM†	0.627 ± 0.001	1.000 ± 0.000	0.627 ± 0.001	0.425 ± 0.000	−37.74 ± 0.00
NULL†	0.644 ± 0.008	0.861 ± 0.002	0.644 ± 0.008	0.704 ± 0.001	99.44 ± 0.05
SGTM	0.503 ± 0.024	0.980 ± 0.008	0.148 ± 0.037	0.264 ± 0.009	−2.16 ± 1.54
MemAdapters†	0.633 ± 0.001	0.646 ± 0.005	0.627 ± 0.001	0.429 ± 0.002	−29.49 ± 0.20
MUSR	0.647 ± 0.004	0.981 ± 0.001	0.646 ± 0.005	0.428 ± 0.001	−36.76 ± 1.48

Table 12: **LUME results for Llama-3.2-1B-Instruct**. Retain and Forget report exact-match accuracy, with retraining as the reference after unlearning. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over up to three runs, and entries marked – are not applicable.

	Before unlearning			After unlearning		
Method	**MMLU ↑**	**Retain ↑**	**Forget**	**MMLU ↑**	**Retain ↑**	**Forget**
Reference						
Frozen base	0.455	0.000	0.001	–	–	–
Joint fine-tune	0.348 ± 0.010	0.983 ± 0.001	0.980 ± 0.002	–	–	–
Retrain	–	–	–	0.330 ± 0.008	0.983 ± 0.001	0.130 ± 0.001
Post-hoc						
GradDiff	–	–	–	0.255 ± 0.003	0.787 ± 0.084	0.007 ± 0.003
NPO	–	–	–	0.321 ± 0.027	0.977 ± 0.003	0.003 ± 0.006
SimNPO	–	–	–	0.334 ± 0.007	0.980 ± 0.004	0.043 ± 0.003
RMU	–	–	–	0.231 ± 0.002	0.001 ± 0.001	0.001 ± 0.001
By construction						
S3T	0.293 ± 0.014	0.982 ± 0.002	0.951 ± 0.003	0.390 ± 0.011	0.982 ± 0.003	0.125 ± 0.003
APA†	0.436 ± 0.016	0.518 ± 0.008	0.503 ± 0.010	0.457 ± 0.013	0.294 ± 0.002	0.129 ± 0.002
GRAM†	0.395 ± 0.012	0.983 ± 0.000	0.980 ± 0.000	0.395 ± 0.012	0.983 ± 0.000	0.150 ± 0.001
NULL†	0.378 ± 0.008	0.983 ± 0.000	0.979 ± 0.000	0.378 ± 0.008	0.983 ± 0.000	0.150 ± 0.001
SGTM	0.257 ± 0.007	0.983 ± 0.001	0.980 ± 0.000	0.234 ± 0.003	0.007 ± 0.005	0.000 ± 0.000
MemAdapters†	0.260 ± 0.002	0.799 ± 0.005	0.797 ± 0.011	0.300 ± 0.024	0.097 ± 0.026	0.015 ± 0.003
MUSR	0.376 ± 0.007	0.981 ± 0.003	0.977 ± 0.003	0.427 ± 0.009	0.983 ± 0.002	0.130 ± 0.001

Adapters preserve utility on some models but learn the assigned source information less consistently. These model-specific results support the main comparison in Figure 3.

D.2 LUME Results

Results across backbones. Tables 12 and 13 report the complete LUME results. On Llama-3.2-1B-Instruct, MUSR matches retraining after unlearning with a Retain score of 0.983 and a Forget score of 0.130. Removing the forget modules also increases MMLU from 0.376 to 0.427, while leaving

Table 13: **LUME results for Qwen2.5-1.5B-Instruct**. Retain and Forget report exact-match accuracy, with retraining as the reference after unlearning. Methods marked with † use source labels or retrieval during inference. Results are mean ± SD over up to three runs, and entries marked – are not applicable.

	Before unlearning			After unlearning		
Method	**MMLU ↑**	**Retain ↑**	**Forget**	**MMLU ↑**	**Retain ↑**	**Forget**
Reference						
Frozen base	0.603	0.000	0.000	–	–	–
Joint fine-tune	0.575 ± 0.004	0.876 ± 0.001	0.876 ± 0.001	–	–	–
Retrain	–	–	–	0.576 ± 0.003	0.877 ± 0.001	0.133 ± 0.002
Post-hoc						
GradDiff	–	–	–	0.570 ± 0.003	0.175 ± 0.046	0.055 ± 0.009
NPO	–	–	–	0.560	0.687 ± 0.011	0.000 ± 0.000
SimNPO	–	–	–	0.571 ± 0.001	0.872 ± 0.006	0.236 ± 0.044
RMU	–	–	–	0.571 ± 0.000	0.844 ± 0.001	0.834 ± 0.000
By construction						
S3T	0.531 ± 0.009	0.839 ± 0.008	0.595 ± 0.005	0.576 ± 0.007	0.740 ± 0.021	0.110 ± 0.004
APA†	0.582 ± 0.009	0.196 ± 0.002	0.203 ± 0.009	0.568 ± 0.004	0.307 ± 0.003	0.121 ± 0.004
GRAM†	0.571 ± 0.007	0.877 ± 0.000	0.876 ± 0.000	0.571 ± 0.009	0.877 ± 0.000	0.145 ± 0.002
NULL†	0.554 ± 0.008	0.877 ± 0.000	0.875 ± 0.001	0.554 ± 0.008	0.877 ± 0.000	0.148 ± 0.002
SGTM	0.261 ± 0.004	0.877 ± 0.000	0.876 ± 0.000	0.251 ± 0.015	0.001 ± 0.001	0.000 ± 0.000
MemAdapters†	0.603 ± 0.005	0.091 ± 0.009	0.097 ± 0.004	0.604 ± 0.003	0.002 ± 0.001	0.000 ± 0.000
MUSR	0.601 ± 0.003	0.873 ± 0.000	0.874 ± 0.001	0.604 ± 0.004	0.873 ± 0.004	0.110 ± 0.003

the retained task performance unchanged. On Qwen2.5-1.5B-Instruct, MUSR preserves its Retain score of 0.873 and reaches a Forget score of 0.110, compared with 0.133 for retraining. Its MMLU score remains at the level of the frozen base model. These results show that the source modules can store document-level information without reducing the general capabilities of the frozen backbone.

Comparison with other methods. GRAM and NULL closely preserve Retain performance on both models and approach the retraining Forget score, but both require the exact source assignment during inference. S^{3}T provides the strongest router-free comparison on Llama-3.2-1B-Instruct, where it reaches 0.982 Retain and 0.125 Forget after unlearning. However, it learns less of the source information and retains less task performance on Qwen2.5-1.5B-Instruct. It also requires retraining the affected slice and every later slice after an unlearning request.

The post-hoc methods do not reach the same balance consistently. SimNPO preserves Retain performance well, but removes more than retraining on Llama and less than retraining on Qwen. GradDiff and NPO remove more than retraining while reducing Retain performance, whereas RMU performs little effective unlearning on Qwen. SGTM loses almost all Retain performance after unlearning on both models. APA and Memory Adapters preserve general MMLU performance but learn substantially less of the LUME task content. Since several LUME entries correspond to one run, these comparisons describe the observed results rather than variation across runs.

E Additional TOFU Analyses

This section provides further analyses of the output penalty, source granularity, module design, continual source addition, and repeated unlearning. Unless stated otherwise, all experiments use Llama-3.2-1B-Instruct and report the mean and standard deviation over three runs.

E.1 Self-Routing with paraphrased Questions

The main analysis measures selective module behavior on the original TOFU questions. We additionally test whether the same behavior extends to paraphrased questions that were not used during training. At $\lambda_{\text{out}} = 10$, the assigned module has an output norm of 0.106 relative to the layer input on the original questions and 0.093 on the paraphrased questions. The summed output of all non-assigned modules is approximately 0.012 in both cases. On the paraphrased questions, removing the

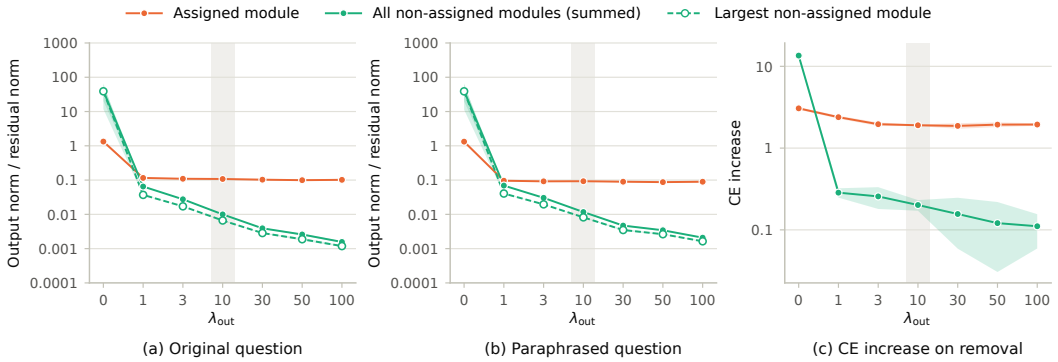


Figure 5: **Self-routing on original and paraphrased TOFU questions.** Panels (a) and (b) report module output norms relative to the hidden state entering each layer. Panel (c) reports the increase in answer cross-entropy after removing the assigned module or all non-assigned modules for the paraphrased questions. Results use the 400 questions from forget10 and show mean ± SD over three runs.

Table 14: **Effect of module type and layer placement on MUSR using Llama-3.2-1B-Instruct.** The gated MLP is the standard module used by MUSR. Parameters are reported per author. ROUGE-only is the harmonic mean of the ROUGE scores on retained authors, real authors, and world facts. Results report mean ± standard deviation over three runs.

			Before unlearning		After unlearning	
Module	**Layers**	**Params/source**	**Utility**	**ROUGE-only**	**Utility**	**Forget**
Gated MLP	0–3	0.39M	0.583 ± 0.013	0.876 ± 0.013	0.566 ± 0.026	0.372 ± 0.030
Gated MLP	**0–7**	0.79M	**0.589** ± 0.022	0.907 ± 0.011	**0.579** ± 0.016	0.369 ± 0.016
Gated MLP	4–11	0.79M	0.565 ± 0.006	**0.920** ± 0.002	0.564 ± 0.006	0.343 ± 0.013
Gated MLP	8–15	0.79M	0.555 ± 0.005	0.882 ± 0.003	0.554 ± 0.004	0.317 ± 0.015
Gated MLP	0–15	1.57M	0.582 ± 0.010	0.869 ± 0.010	0.572 ± 0.014	0.365 ± 0.003
LoRA	0–7	0.79M	0.540 ± 0.005	0.607 ± 0.014	0.532 ± 0.016	0.329 ± 0.005
LoRA	0–15	1.57M	0.556 ± 0.008	0.680 ± 0.020	0.477 ± 0.060	0.339 ± 0.016

assigned module increases answer cross-entropy by 1.90, while removing all non-assigned modules increases it by only 0.20. The separation between assigned and non-assigned modules therefore extends beyond the exact wording used during training.

E.2 Module Type and Layer Placement

Table 14 compares the gated MLP used by MUSR with LoRA modules and varies the equipped layers. At the same parameter count of 0.79 million parameters per source, the gated MLP on layers 0–7 reaches a utility of 0.589, compared with 0.540 for LoRA. It also reaches a substantially higher ROUGE-only score of 0.907, compared with 0.607. Moving the gated modules to layers 4–11 gives the highest ROUGE-only score but reduces overall utility. Equipping all layers doubles the number of parameters without improving the results. We therefore use gated modules on the earlier layers as the default configuration.

E.3 Module Width

Table 15 varies the module width from 4 to 64. Performance remains relatively stable across the smaller configurations, and widths above 16 provide no consistent improvement despite increasing storage. Width 16 gives the highest utility before unlearning and strong performance after unlearning. We therefore use it as the default balance between model quality and storage.

Table 15: **Effect of module width on MUSR using Llama-3.2-1B-Instruct.** Parameters are reported per author, while total capacity is reported relative to the base model. The standard configuration is shown in bold. ROUGE-only is the harmonic mean of the ROUGE scores on retained authors, real authors, and world facts. Results report mean ± standard deviation over three runs.

			Before unlearning		After unlearning	
Units	**Params/source**	**Total (% base)**	**Utility**	**ROUGE-only**	**Utility**	**Forget**
4	0.20M	3.2%	0.579 ± 0.011	0.870 ± 0.018	0.568 ± 0.016	0.371 ± 0.050
8	0.39M	6.4%	0.584 ± 0.021	0.906 ± 0.006	0.582 ± 0.019	0.351 ± 0.015
16	**0.79M**	**12.7%**	**0.589** ± 0.022	**0.907** ± 0.011	**0.579** ± 0.016	**0.369** ± 0.016
32	1.57M	25.5%	0.570 ± 0.002	0.907 ± 0.001	0.568 ± 0.003	0.363 ± 0.017
64	3.15M	50.9%	0.560 ± 0.008	0.898 ± 0.005	0.558 ± 0.008	0.366 ± 0.008

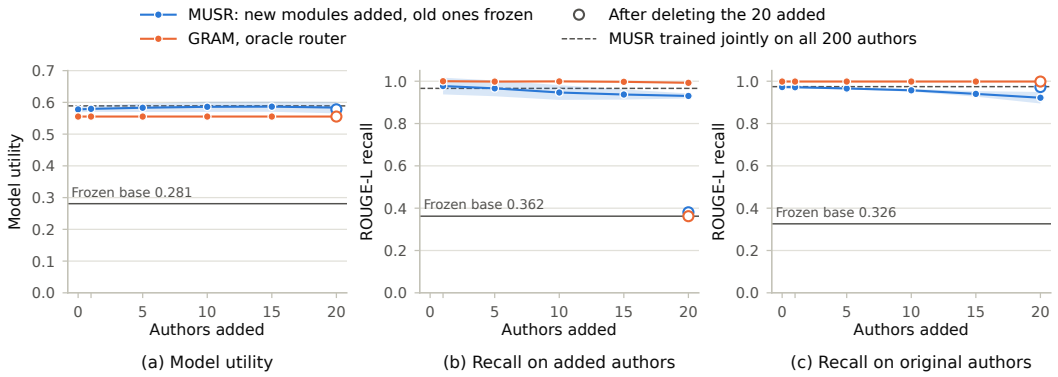


Figure 6: Continual source addition on TOFU. We begin with modules for 180 authors and add modules for 20 new authors while keeping the existing modules frozen. Hollow markers show the model after unlearning all added sources. The dashed line shows MUSR trained jointly on all 200 authors. Results use one run.

E.4 Continual Source Addition

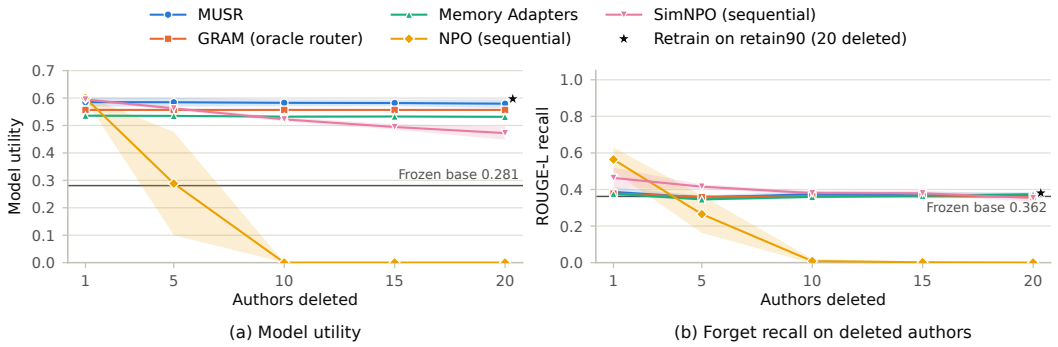
We test whether new sources can be added without retraining the existing modules. We begin with a model trained on the 180 authors in retain90 and add modules for the remaining 20 authors one at a time. Existing modules remain frozen, while examples from the original authors are replayed when training each new module. After adding all 20 authors, utility changes from 0.581 to 0.578, and recall on the added authors reaches 0.963. Recall on the original authors decreases from 0.976 to 0.952, showing that the newly trained modules still introduce some interference. After unlearning all added sources, utility and recall on the original authors return to their initial values.

E.5 Continual Unlearning

We next apply a sequence of unlearning requests to the same model. The utility of MUSR changes from 0.589 before unlearning to 0.579 after unlearning 20 authors. Across the sequence, ROUGE-L recall on the unlearned authors remains between 0.356 and 0.387, close to the frozen backbone value of 0.362 and the retraining value of 0.381. GRAM and Memory Adapters are also stable, although both depend on an external source-dependent selection mechanism. In contrast, sequential NPO collapses after ten requests, while the utility of SimNPO decreases from 0.604 to 0.472. Local module unlearning therefore supports repeated requests without accumulating the damage observed for the post-hoc methods.

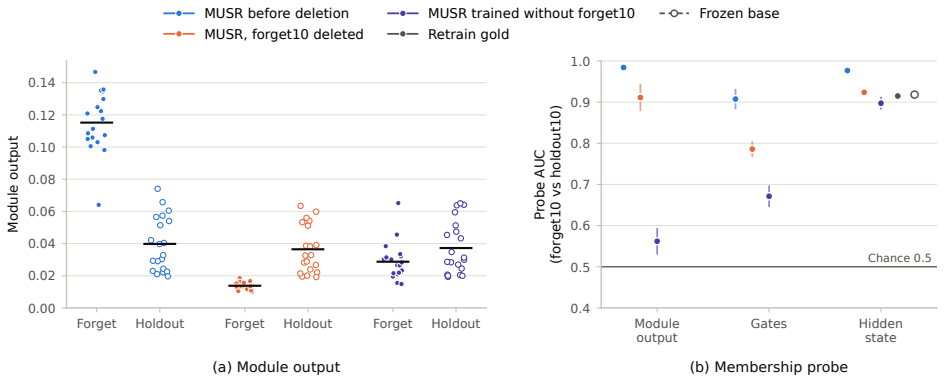
E.6 Residual Information and Relearning After Unlearning

For a sample from a source, the task loss updates only its assigned module, while the output penalty also updates the non-assigned modules. These modules may therefore retain a trace of the source after it has been unlearned by zeroing its assigned module. We test whether such a trace exists and whether it makes the unlearned information easier to recover. We compare MUSR after unlearning



(a) Model utility (b) Forget recall on deleted authors

Figure 7: Continual unlearning on TOFU. Each point applies an additional unlearning request to the current model. Panel (a) reports model utility, while panel (b) reports ROUGE-L recall across all authors unlearned up to that point. The star shows retraining without forget10 after all 20 authors have been unlearned. Results show mean ± SD over three runs.



(a) Module output (b) Membership probe

Figure 8: Residual traces after unlearning forget10. Panel (a) reports the summed module output on forget10 and holdout10 for each author, normalized by the hidden-state norm. Before unlearning, this includes all modules, while after unlearning and for the model trained without forget10 it includes the 180 retained modules. Panel (b) reports the AUC of a probe trained and tested on different authors that distinguishes the two sets using module outputs, gates of the retained modules, or hidden states. Results in (b) are mean ± SD over three runs, except for the frozen base model.

forget10 with MUSR before unlearning and with MUSR trained only on retain90, and use holdout10 as a set of authors that none of these models encountered during training.

Residual trace. Figure 8(a) shows the summed module output on the question tokens, normalized by the norm of the hidden state. Before unlearning, the output on forget10 is 0.115 compared with 0.040 on holdout10, since the assigned modules are still active. After unlearning, the retained modules produce a smaller output on forget10 than on holdout10, with 0.014 against 0.036. For the model trained without forget10, this difference is smaller, with 0.029 against 0.037. This matches the output penalty, which trains the retained modules to stay near zero on forget10. In Figure 8(b), a probe trained and tested on different authors distinguishes forget10 from holdout10 using the outputs of the retained modules with an AUC of 0.911 ± 0.031, compared with 0.562 ± 0.030 for the model trained without forget10. The gate activations show the same pattern with a smaller gap (0.786 ± 0.017 against 0.671 ± 0.024). Hidden states separate the two sets well for every model, including the frozen base model and retraining, so this separation reflects differences between the authors in forget10 and holdout10 rather than training on forget10. Before unlearning, the hidden states of MUSR reach an AUC of 0.977, and after unlearning 0.924, close to retraining with 0.915 and the frozen base model with 0.918. The training-specific trace is therefore visible in the retained modules, while the hidden states show no additional separation beyond retraining and the frozen base model.

Relearning. We next test whether this trace makes the unlearned information easier to recover (Figure 9). For each model, we fine-tune all base-model weights on four samples from every author

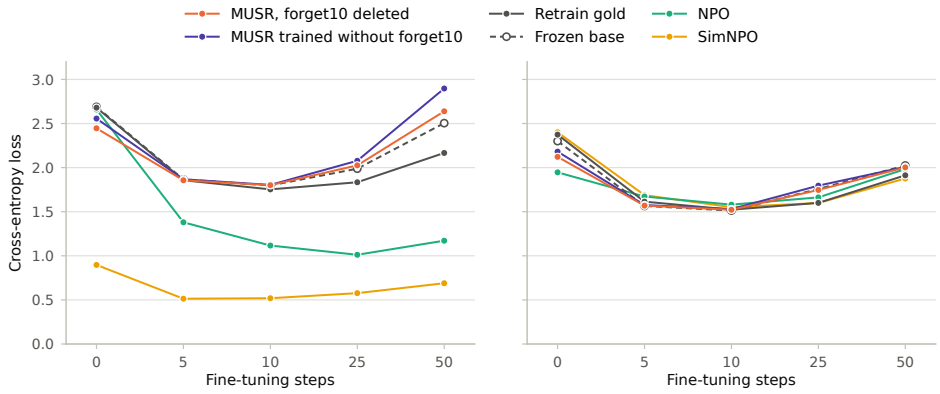


Figure 9: Relearning after unlearning forget10. Each model is fine-tuned on four samples per author, and we report the cross-entropy on the remaining samples of (a) forget10 and (b) holdout10. Results are from one run per model.

in forget10 and report cross-entropy on the remaining samples in panel (a). As a control, we repeat the experiment on holdout10 in panel (b). On forget10, MUSR after unlearning reaches the same loss as the models that never trained on forget10. After 10 steps, it obtains 1.80, compared with 1.80 for MUSR trained without forget10, 1.80 for the frozen base model, and 1.75 for retraining. On holdout10, all models reach a similar loss of about 1.5. This comparable improvement on unseen authors suggests that much of the initial decrease on forget10 reflects learning the answer format rather than recovering the unlearned information. In contrast, NPO drops to 1.12 on forget10, so its unlearned information is recovered much faster. SimNPO already starts at 0.90 before fine-tuning, indicating insufficient unlearning. After 10 to 25 steps, the loss increases for all models as they overfit to the few training samples. The retained modules therefore contain a detectable trace of the output penalty as shown in Figure 8, but this experiment provides no evidence that they contain recoverable information about the unlearned authors. MUSR is consequently not identical to retraining, but the unlearned model is as robust to relearning as retraining in this experiment.

F Storage and Inference Cost

Table 16 compares storage, estimated FLOPs, and the decode time spent in the added components on Llama-3.2-1B-Instruct at batch size 1. For MUSR, increasing K from 10 to 200 increases storage from 7.9M to 157.3M parameters and the estimated FLOPs from 0.64% to 12.73%, while the adapter cost only increases from 1.69 to 1.89 ms per token. The cost of MUSR is therefore dominated by a fixed cost per adapted layer rather than by the number of modules. GRAM and APA, which select components for each input, cost 2.22 and 2.31 ms per token and are slower than MUSR for every K, although GRAM requires fewer FLOPs. Evaluating GRAM with all modules and without routing costs 1.72 ms per token, so the selection of modules itself adds about 0.5 ms per token. NULL costs 1.79 ms per token but requires the source label during inference, and Memory Adapters add the least time.

Table 16: **Storage and inference overhead on Llama-3.2-1B-Instruct at batch size one.** Stored and Added VRAM are measured beyond the frozen base model. Adapter cost is the decode time per generated token spent in the added components, measured separately at batch size one on one 40 GB A100 in bfloat16. Extra FLOPs is the estimated increase in forward-pass FLOPs relative to the base model. All methods use the configurations in Appendix B.

Method	**Stored (M)**	**Added VRAM (GiB)**	**Adapter cost (ms/tok)**	**Extra FLOPs**
base	0	–	–	–
NULL	157.3	+0.294	1.79	+12.73%
MemAdapters	2149.1	+8.01	0.34	+0.05%
APA	104.9	+0.195	2.31	+8.48%
GRAM	157.3	+0.293	2.22	+0.06%
GRAM without routing	157.3	+0.293	1.72	+12.73%
MUSR, by number of stored adapters K				
10	7.9	+0.015	1.69	+0.64%
50	39.3	+0.073	1.71	+3.18%
100	78.7	+0.146	1.75	+6.36%
200	157.3	+0.293	1.89	+12.73%