CAUSAL ROUTING FOR UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs cannot forget the way we delete a file. Strangely, we are asked to remove something that was never put anywhere in particular. What the model took from a piece of text is now smeared across billions of weights. Existing methods rewrite all of them to change one thing, and none of them say which part produced that change. To address this, we introduce Causal Routing for Unlearning (CRU) by asking where the concept is expressed in the model and suppressing only that part. One untrained forward pass over the forget set ranks neurons by how their activations vary. Then, small routing modules on those neurons gate and suppress only the concepts that need to be forgotten. In CRU, the base model is frozen, and any change in behavior is caused only by the gated neurons; hence, why the routing is causal. Due to our parameter efficiency (only $\sim 0.01\%$ as many parameters as the base model), unlearning a concept costs 14 GiB, whereas the baselines require 71 GiB. On TOFU, CRU is indistinguishable from the retained model ($p > 0.05$, KS test) and is never Pareto-dominated, whereas every compared baseline matches its forgetting on the larger-forget batches only by collapsing utility. On RWKU, it achieves an adversarial-probe recall of 0.052, compared to 0.250 for the strongest baseline, meaning the knowledge is gone, not merely harder to reach. Thus, deciding on the intervention at query time, rather than fixing it beforehand, is the axis along which we argue that unlearning should proceed.

1 INTRODUCTION

A model cannot forget (unlearn) the way we delete a file. If we delete a record from a database, we know exactly where it was: we delete the row, and the data disappears. A language model stores nothing at a given address. What it learned from a sentence is distributed across billions of weights, tangled with everything else it knows, and there is no row to strike. This is why the right to be forgotten GDPR Info (2026); California Department of Justice (2026), which is trivial to honor in a traditional database, becomes, inside a trained model, a genuinely strange request: *we are asked to remove something that was never put anywhere in particular.*

The honest solution is to retrain the model on everything except the data we want to forget. Yet training an LLM end-to-end is infeasible for any small or medium-sized research lab or company. The only practical alternative is to navigate the model's weight and concept space and update it so that the forget set can no longer be retrieved.

Existing methods tackle this by pushing probability mass off the forget set: by maximizing loss on it (Yao & Xu, 2024), saturating that objective so the model does not diverge (Zhang et al., 2024; Fan et al., 2024a), redirecting the query to a refusal (Rafailov et al., 2023), acting on the context alone (Pawelczyk et al., 2024), or rewarding the absence of the concept (Zaradoukas et al., 2026). The methods share a common trait: *they rewrite the whole model to change one thing*. Even when the change is the intended one, nothing in the procedure tells us which part of the network produced it, so the result is expensive to obtain, hard to reconstruct, and hard to reuse for a second concept.

A second line of work leaves the weights untouched. NeuMuter (Hou et al., 2025) trains a mask over a small set of neurons with the model frozen, but the mask is fixed once trained, so it applies to every input alike. DSG (Muhamed et al., 2025) and GUARD-IT (Turani et al., 2026) do respond to the input, but only through a predefined threshold and an intervention magnitude. This second family shares the same shortcoming as the first: *the intervention is decided before the model is ever queried*. In other words, there is no mechanism distinguishing a prompt that reaches

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.

for the forgotten concept from one that merely passes nearby, so *suppression is spent uniformly whether or not it is needed* (Figure 1). Consequently, model utility drops lower than it needs to.

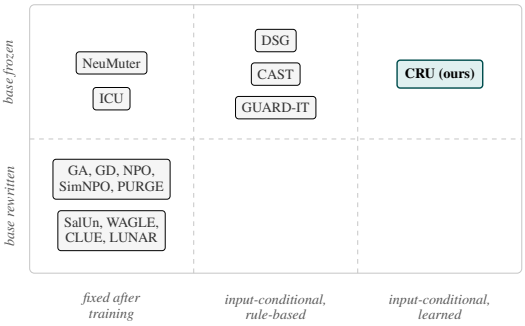


Figure 1: Unlearning methods positioned by whether the base model is modified or not; coarsely inspired from Yoon et al. (2026).

To enforce the distinction between concepts involved in the forget set and the rest, our method, Causal Routing for Unlearning (CRU), splits unlearning into *localization*, i.e., identifying the neurons that encode forget-set concepts, and *gating* of their activations.

Localization. CRU collects hidden states from the last few decoder blocks over a sample of the forget set. We note the tension with weight-editing work that locates factual associations in middle layers (Meng et al., 2022): those methods intervene on where a fact is *retrieved*, whereas CRU intervenes on where it is *expressed*. Hence, CRU pools each example over its non-padding positions and ranks neurons by how much that pooled activation varies from one example to the next; neurons above a percentile threshold become the candidate set for that block. This step determines *where* CRU may intervene, restricting it to a small fraction of neurons in a few blocks, and entails only one forward pass and no training. It does not by itself make the intervention specific to the forget set; that is enforced by the gating objective (Equation (8)), which penalizes closing gates on retain inputs and thus determines *when* each selected neuron is suppressed.

Gating. At each of those blocks, CRU attaches a small MLP ($\sim$0.01% of total parameters of the base model) that reads the block's output hidden state and emits, for every selected neuron at every token position, a value in $[0, 1]$ that multiplies it. Training pushes those values toward 0 for forget-set inputs and toward 1 for the remaining inputs; only the MLPs receive gradients, and the objective is defined on the gate values themselves rather than on the model's output distribution. Because the base model remains entirely frozen, there is no parameter drift, and all behavioral changes are attributable to the gated neurons.

2 PRELIMINARIES

Models and data. We consider an autoregressive language model $\pi_\theta(y \mid x)$ with parameters $\theta \in \Theta$, mapping prompt x to response distribution y via negative log-likelihood loss:

$$\ell(y \mid x; \theta) \;=\; -\log \pi_\theta(y \mid x) \;=\; -\sum\nolimits_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t})\,. \tag{1}$$

A learning algorithm $\mathcal{A}$ maps a dataset to parameters. For training set $\mathcal{D}$, the reference (original) model parameters before unlearning are $\theta_{\mathrm{o}} = \mathcal{A}(\mathcal{D})$, yielding distribution π_{o}.

Machine unlearning. Let $\mathcal{D}_f \subseteq \mathcal{D}$ be the *forget set*, the data whose influence is to be removed, and $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$ the *retain set*. Machine unlearning asks for a model that behaves as though $\mathcal{D}_f$ had never been seen (Cao & Yang, 2015). The construction that achieves this by definition is retraining.

Definition 1 (Exact unlearning)**.** Given learning algorithm $\mathcal{A}$ and retain set $\mathcal{D}_r$, the *retrained model* π_{retr} has parameters $\theta_{\mathrm{retr}} = \mathcal{A}(\mathcal{D}_r)$.

π_{retr} serves as the *golden reference point* by excluding $\mathcal{D}_f$. Since retraining $\mathcal{A}(\mathcal{D}_r)$ is computationally intractable for LLMs, unlearning aims to approximate π_{retr} directly (Bourtoule et al., 2021).

Definition 2 (Approximate unlearning)**.** Given π_{o}, $\mathcal{D}_f$, and $\mathcal{D}_r$, an unlearning algorithm $\mathcal{U}$ outputs an *unlearned model* $\pi_{\mathrm{u}} = \mathcal{U}(\pi_{\mathrm{o}}, \mathcal{D}_f, \mathcal{D}_r) \approx \pi_{\mathrm{retr}}$ without retraining.

While strict definitions demand (ϵ, δ)-indistinguishability between π_{u} and π_{retr} (Guo et al., 2020; Sekhari et al., 2021), such certificates are impractical for non-convex LLMs trained via stochastic optimization (Thudi et al., 2022). Furthermore, real-world deployments lack full access to $\mathcal{D}$, $\mathcal{D}_r$,

and π_{retr}. Evaluation, therefore, relies on proxy metrics measuring *forget quality* on $\mathcal{D}_f$ and *utility retention* on $\mathcal{D}_r$ and general benchmarks (Maini et al., 2024; Shi et al., 2025).

What $\mathcal{U}$ is allowed to do. Existing unlearning methods differ along two independent axes: the *objective* optimized and the *scope* of the intervention.

Objective. Unlearning typically optimizes a regularized loss balancing a forget term on $\mathcal{D}_f$ and forget loss ℓ_f against a utility retention term on $\mathcal{D}_r$ and loss ℓ (Liu et al., 2025):

$$\min_{\theta\in\Theta} \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_f}[\ell_f(y \mid x;\theta)]}_{\text{forget}} + \lambda \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_r}[\ell(y \mid x;\theta)]}_{\text{retain}}, \tag{2}$$

where $\lambda > 0$ controls the trade-off, and state-of-the-art methods vary choices of ℓ_f and λ (§A).

Scope. While Equation (2) permits updates across all of Θ, any parameter-space unlearning operator can be generalized as

$$\theta_{\text{u}} \;=\; \theta_{\text{o}} + \mathbf{w} \odot \Delta, \qquad \mathbf{w} \in [0,1]^d, \tag{3}$$

where Δ is the objective update, $d = \dim\Theta$, $\odot$ denotes elementwise multiplication, and $\mathbf{w}$ selects which coordinates receive updates.

3 Related Work

3.1 Base Model Update

Global updates. Gradient Ascent (GA) (Yao & Xu, 2024) performs gradient ascent on the cross-entropy loss of the forget set, lowering its likelihood; the objective is unbounded, so parameters diverge and coherence collapses. Gradient Difference (GD) (Liu et al., 2022) adds a retain accuracy penalty to balance this. DPO (Rafailov et al., 2023) trains the model to prefer a refusal (e.g., "I don't know") over the original answer to each forget question. NPO (Zhang et al., 2024) pushes down the probability of the original answers, measured relative to the model before unlearning, thereby keeping the loss bounded. SimNPO (Fan et al., 2024a) drops this comparison with the original model, which otherwise spends the same effort on every example regardless of how hard it is to forget. PURGE (Zaradoukas et al., 2026) uses RL where the model is penalized whenever its output mentions the target concept, while a KL term keeps it close to the original one to preserve utility. Since the reward only checks what the model says, it remains unclear whether the knowledge is erased or merely hidden. All global methods fix $\mathbf{w} = \mathbf{1}$, thus disregarding the input.

Localized updates. A second group edits only a subset of the weights (i.e., $\mathbf{w} \neq \mathbf{1}$). SalUn (Fan et al., 2024b) updates only the weights whose gradients on the forget set are largest. WAGLE (Jia et al., 2024) estimates how much each weight contributes to unlearning and applies standard objectives (e.g., GradDiff, NPO) only to the most influential ones. CLUE (Chen et al., 2026) traces the circuits used by the forget set, separates neurons specific to it from those shared with other knowledge, and fine-tunes only the former. LUNAR (Shen et al., 2025) edits the MLP output projections of a few layers so that forget inputs produce activations resembling those of a refusal. Crucially, while these methods set $\mathbf{w} \neq \mathbf{1}$, their modified weights apply indiscriminately to every input.

3.2 Frozen Base Model

Input-independent interventions. NeuMuter (Hou et al., 2025) trains a fixed mask over $\sim 1\%$ of feed-forward neurons on frozen weights, applying it uniformly to all inputs. Parameter-free ICU (Pawelczyk et al., 2024) prepends label-flipped examples, but context-bound interventions can be bypassed when reasoning traces reconstruct forgotten concepts (Zaradoukas et al., 2026).

Rule-based conditional interventions. Rule-based methods steer activations only when heuristic triggers fire. CAST (Lee et al., 2025) compares hidden states to a condition vector, applying a fixed steering vector to all generated tokens if a grid-searched threshold is met on the prompt. DSG (Muhamed et al., 2025) and GUARD-IT (Turani et al., 2026) follow the same paradigm, clamping sparse autoencoder features or prototype directions to fixed magnitudes upon meeting tuned thresholds. In all cases, trigger thresholds and intervention magnitudes are user-defined hyperparameters.

We refer the reader to §A.1 for details on how the SoTA operates under Equation (2), and how CRU differs from it according to Equation (3) (see §A.2).

4 CAUSAL ROUTING FOR UNLEARNING

CRU[1] is very simple (see Figure 2). It operates in two phases. The first selects, for each block in a small set, the neurons whose activations vary most over the forget set. The second attaches a small routing network to each of those blocks and trains it to attenuate those neurons on forget-set inputs while leaving them intact otherwise. The base model is frozen in both phases.

4.1 PHASE 1: LOCALIZING CONCEPT NEURONS

Which blocks. Earlier layers in transformers carry surface and syntactic features, while later layers carry semantic ones (Tenney et al., 2019); feed-forward layers act as key-value memories whose outputs refine the distribution over the vocabulary (Geva et al., 2021); and neurons expressing a given fact are concentrated in the last layers (Dai et al., 2022). Decoding methods exploit the same asymmetry, contrasting late against early layers to amplify factual content (Chuang et al., 2024). Late blocks are, therefore, where a concept is most nearly expressed in the form the output head consumes, which is what a multiplicative gate can act on. We, thus, collect activations from the last k decoder blocks.

Pooling. For a forget-set example i with token mask $m_t^{(i)} \in \{0,1\}$, we take the mean hidden state of block l over non-padding positions,

$$\bar{\mathbf{a}}_l^{(i)} \;=\; \frac{\sum_t m_t^{(i)}\, \mathbf{h}_{l,t}^{(i)}}{\sum_t m_t^{(i)}}, \qquad \mathbf{h}_{l,t}^{(i)} \in \mathbb{R}^d, \quad (4)$$

where $\mathbf{h}_{l,t}$ is the output hidden state of block l at position t and d is the model width. Padding is excluded throughout: sequence length is an artifact of batching, and averaging over it would let downstream statistics key on how much a corpus was padded rather than on its content.

Variance as a relevance score. For each block, we score neuron j by its variance across forget-set examples,

$$\sigma_l^2(j) \;=\; \mathrm{Var}_i\Big(\bar{\mathbf{a}}_l^{(i)}[j]\Big), \quad (5)$$

estimated over a fixed number of forget batches. A neuron whose pooled activation barely moves across examples of the concept is unlikely to be carrying it. This is a cheap surrogate for the attribution methods used elsewhere in the localization literature – integrated gradients (Dai et al., 2022), task predictivity (Wang et al., 2022), activation probability (Tang et al., 2024), or neuron-level attribution (Yu & Ananiadou, 2024). It requires a single pass and no backward computation.

Percentile thresholding. We keep the neurons above the p-th percentile of the per-block variance distribution,

$$\mathcal{C}_l \;=\; \big\{\, j : \sigma_l^2(j) > q_p\big(\sigma_l^2\big) \big\}. \quad (6)$$

A percentile is robust to the shape of the variance distribution, which matters here. In particular, activations in large transformers are dominated by a small number of high-magnitude outlier dimensions (Timkey & van Schijndel, 2021; Dettmers et al., 2022), so a threshold defined by mean and standard deviation would be set almost entirely by those coordinates.

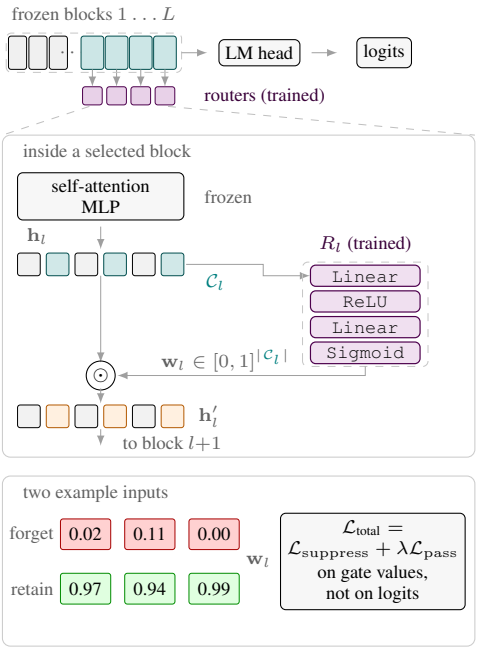


Figure 2: **CRU applied to a frozen decoder.** Each of the last k blocks pairs with a routing MLP, which reads the output hidden state $\mathbf{h}_l$ and emits a gate in $[0,1]$ for every selected neuron at every token position. Gates multiply the $\mathcal{C}_l$ coordinates of $\mathbf{h}_l$; all other coordinates pass through at 1. Training minimizes the mean gate value on forget batches and drives it to one on retain batches: the objective is defined on the gate outputs, so no gradient passes through the base model.

[1] https://anonymous.4open.science/r/cru-iclr-4E56/README.md

4.2 PHASE 2: LEARNED ROUTING

Phase 1 asks which neurons carry the concept, and answers it with a statistic pooled over each example. Phase 2 asks a narrower question at every token: *given what is passing through this position now, should those neurons be closed?* Selection is therefore made once per concept, while the intervention is decided per position.

Where the gate acts. Each selected block receives its own routing module, applied to that block's output hidden state – that is, to the residual stream, the channel through which blocks communicate (Elhage et al., 2021) and the site at which activation-space interventions are conventionally applied (Turner et al., 2023). Acting there rather than inside the block keeps the intervention independent of the block's internal parameterization, so the same construction applies unchanged across the model families we test, even though their feed-forward blocks differ.

Router architecture. Let $\mathcal{C}_l = \{j_1 < j_2 < \cdots < j_{|\mathcal{C}_l|}\}$ for the selected neurons of block l, indexed in increasing order. For that block, we define $R_l : \mathbb{R}^d \to [0,1]^{|\mathcal{C}_l|}$,

$$\mathbf{w}_{l,t} \;=\; R_l(\mathbf{h}_{l,t}) \;=\; \sigma(\mathbf{W}_2\,\mathrm{ReLU}(\mathbf{W}_1\mathbf{h}_{l,t} + \mathbf{b}_1) + \mathbf{b}_2)\,, \tag{7}$$

with $\mathbf{W}_1 \in \mathbb{R}^{d_r \times d}$, $\mathbf{b}_1 \in \mathbb{R}^{d_r}$, $\mathbf{W}_2 \in \mathbb{R}^{|\mathcal{C}_l| \times d_r}$, $\mathbf{b}_2 \in \mathbb{R}^{|\mathcal{C}_l|}$, bottleneck width $d_r << d$. The same parameters are applied independently at every position t, so the module costs $|\mathcal{C}_l|$ outputs per token rather than d. The entry $\mathbf{w}_{l,t}[m]$ is the gate on neuron j_m; this correspondence between output row and neuron is a stipulation of the construction, a point we return when discussing the objective.

The gate is applied multiplicatively and only on coordinates corresponding to the neurons selected in Phase 1, yielding $\mathbf{h}'_{l,t} \in \mathbb{R}^d$ with

$$\mathbf{h}'_{l,t}[j] \;=\; \begin{cases} \mathbf{w}_{l,t}[m] \cdot \mathbf{h}_{l,t}[j], & j = j_m \in \mathcal{C}_l, \\ \mathbf{h}_{l,t}[j], & j \notin \mathcal{C}_l. \end{cases} \tag{8}$$

Three properties follow. The gate is multiplicative, which is the argument for activation scaling over additive steering (Stoehr et al., 2024); sigmoid gating of this form is standard in transformer feed-forward design (Dauphin et al., 2017; Shazeer, 2020). It is soft: because each $\mathbf{w}_{l,t}[m] \in [0,1]$ is continuous, the intervention is differentiable and admits partial suppression, unlike a hard mask. And it is input-conditional and evaluated per token, so the same neuron may be suppressed at one position and passed at another. Together with a frozen base and a small trained module, this places CRU in the same family as adapter-style parameter-efficient methods (Houlsby et al., 2019; Hu et al., 2022), and R_l plays the role a router plays in conditional computation (Shazeer et al., 2017).

Objective. Let $\bar{w}_l^{\mathrm{f}}$ and $\bar{w}_l^{\mathrm{r}}$ denote the mean gate value of block l, taken jointly over the selected neurons and the non-padding positions of a forget and a retain batch respectively; padded positions are excluded, since the router is never asked to act on them at inference. With K the set of gated blocks, we minimize

$$\mathcal{L}_{\text{total}} \;=\; \underbrace{\frac{1}{|K|}\sum_{l \in K} \bar{w}_l^{\mathrm{f}}}_{\mathcal{L}_{\text{suppress}}} \;+\; \lambda\, \underbrace{\frac{1}{|K|}\sum_{l \in K} (\bar{w}_l^{\mathrm{r}} - 1)^2}_{\mathcal{L}_{\text{pass}}}. \tag{9}$$

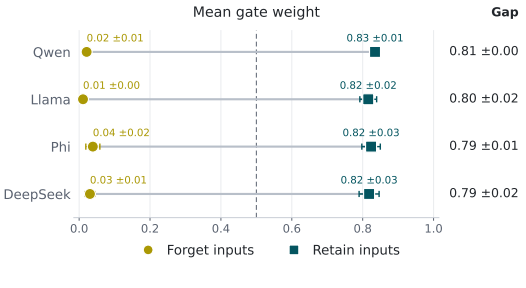


Figure 3: Mean routing gate on the forget and retain corpora, per model on `TOFU forget10`, averaged over 3 runs.

We motivate the asymmetry between the two terms in §B. Two things follow from Equation (9). First, the objective is defined in terms of the gate values themselves and never observes the model's output. Hence, no gradient passes through the language-model head, and the base weights are untouched by construction rather than by choice. Second, the objective depends on the router only through the two scalars $\bar{w}_l^{\mathrm{f}}$ and $\bar{w}_l^{\mathrm{r}}$: any reallocation of gate mass over the selected coordinates and token positions that leaves those means fixed leaves the loss unchanged. Permutations of the router's output rows are included. The correspondence between row m and neuron j_m is stipulated by Equation (8), but Equation (9) gives it no gradient. Hence, two routers differing by such a permutation are equally optimal while suppressing different neurons.

What the objective can and cannot do. The two terms are in tension only when the router can tell the two corpora apart. If it cannot, they resolve to a single value.

Proposition 1. Suppose a gate is constrained to take the same value w on forget and retain inputs. Then $\mathcal{L}(w) = w + \lambda(w-1)^2$ is minimized over $\mathbb{R}$ at $w^\star = 1 - \frac{1}{2\lambda}$. Consequently $w^\star \leq 0$ whenever $\lambda \leq \frac{1}{2}$, and $w^\star = \frac{1}{2}$ at $\lambda = 1$.

Proof. $\mathcal{L}'(w) = 1 + 2\lambda(w-1) = 0$ gives $w^\star = 1 - 1/(2\lambda)$, and $\mathcal{L}''(w) = 2\lambda > 0$ □

Proposition 1 shows that a collapsed router produces equal gate means of $1 - 1/(2\lambda)$ (0.5 at default $\lambda = 1$). Our empirical results diverge sharply: on `TOFU` `forget10`, gate values average 0.01–0.06 on forget inputs (effectively disabling selected neurons) versus 0.79–0.85 on retain inputs (Figure 3). Thus, CRU dynamically reads the input rather than applying fixed attenuation, distinguishing it from frozen-base methods (§3). The retained mean (0.82) reflects minor activation withholding on retain inputs; unlike weight editing, this cost is naturally bounded because underlying parameters remain untouched (§5.1). Finally, Proposition 1 confirms that higher retain pressure (λ) shifts the baseline optimum toward 1, sacrificing retain attenuation first.

5 Experiments

Benchmarks and baselines. We evaluate CRU on `TOFU` (Maini et al., 2024) (`forget01` and `forget10` splits) and `RWKU` (Jin et al., 2024); metrics are summarized in Table 3. We compare against eight SoTA baselines: GA, GD, NPO (Zhang et al., 2024), SimNPO (Fan et al., 2024a), ICU (Pawelczyk et al., 2024), PURGE (Zaradoukas et al., 2026) (on `RWKU`), and LUNAR (Shen et al., 2025) (via its *deviation score* on its custom `TOFU` split; §C). Unlike prior work targeting single architectures, we test across four model families: DeepSeek 7B, Qwen2.5 7B, Llama 3.1 8B, and Phi 3 Mini (§C). **In summary, we evaluate CRU on 2 benchmarks (4 forget sets), 4 model families, and against 7 baselines.**

Hyperparameters. We use the default hyperparameters for the baselines on `TOFU` and `RWKU`. For PURGE, we use its published model checkpoints. For CRU, we use the recommended learning rate for `TOFU` and perform a small grid search for `RWKU` (§C.1). We set $k = 4$ and $d_r = 32$ for TOFU, $d_r = 256$ for RWKU.

5.1 Results

#1 *CRU has the best Model Utility-Forgetting trade-off on `TOFU` splits (statistically significant and not Pareto-dominated). On `RWKU`, CRU is the only one that removes the target knowledge against all attack levels.*

CRU achieves the best utility-forgetting trade-off across all forget sizes on TOFU. Figure 4 plots Model Utility against Forgetting (Rouge-L), where top-right is the ideal performance. On `forget01`, CRU outperforms all baselines on Llama, Phi, and Qwen, while matching top baselines (NPO, SimNPO) on DeepSeek. On `forget10`, high Forgetting scores for GA, NPO, and SimNPO are artifacts caused by utility collapsing to 0.00–0.21 (Figure 4). For example, on Llama, NPO/SimNPO reports 0.97 forgetting score at 0.00 utility, whereas CRU achieves 0.53 utility. Overall, CRU maintains the best trade-off without destroying generation capabilities. This trade-off advantage is statistically significant: i.e., a Friedman test (Demšar, 2006) across all (model, split) blocks rejects equivalence ($p \in [2.2\text{e}^{-4}, 3.5\text{e}^{-3}]$), with CRU holding the top mean rank (1.20–1.80) and beating every baseline in post-hoc Wilcoxon tests ($p \leq 0.039$). Crucially, CRU is never Pareto-dominated.

Additionally, we found out that Shen et al. (2025) use a bespoke forget set split on `TOFU`. For completeness, we compare CRU to LUNAR according to their introduced deviation score (distance from ideal point; lower is better) on their custom split. CRU achieves 7.1 on Qwen (averaged over 3 runs) compared to LUNAR's 13.2 and baselines' 50.4–92.9 (Figure 7).

CRU guarantees knowledge removal rather than hiding. While `TOFU` evaluates fictitious targets, `RWKU` tests real-world figures entangled with general knowledge. Figure 5 compares CRU against

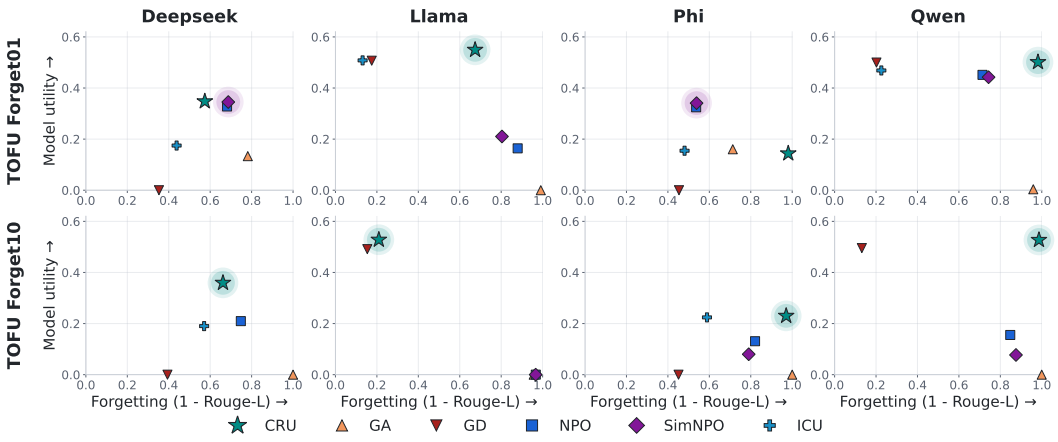


Figure 4: Model Utility and ROUGE-L Forgetting trade-off for every model and both TOFU splits. Higher is better for both metrics. The best method is highlighted.

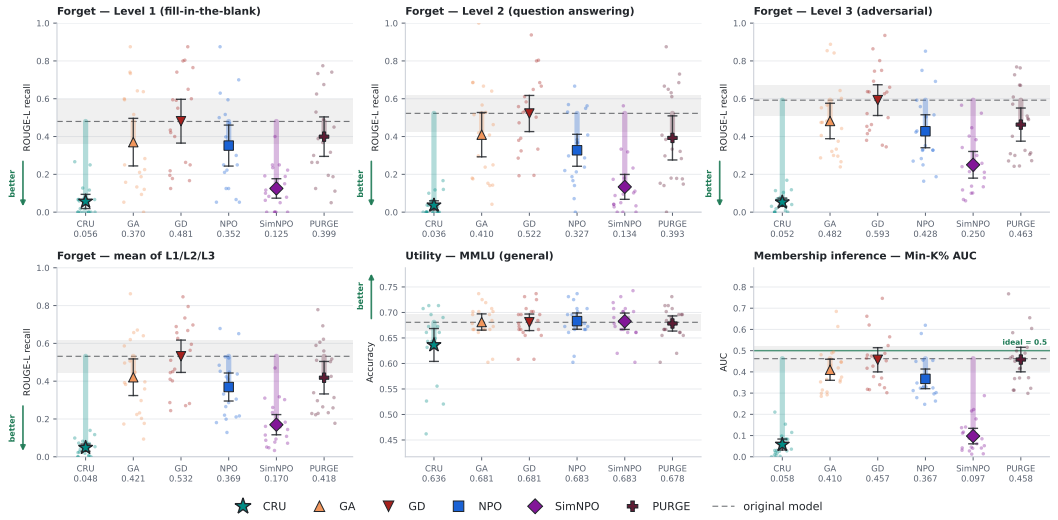


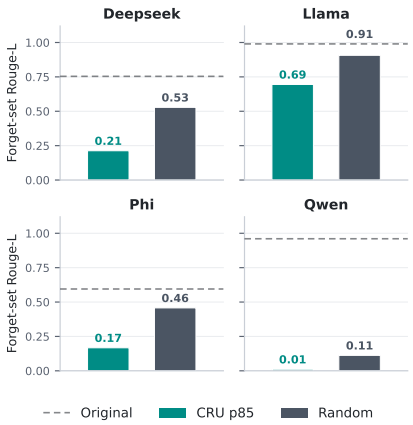
Figure 5: **CRU on RWKU against the baselines with Phi-3-mini, 20 targets.** Each marker is the mean over targets, and the whiskers are the 95% confidence interval; the dots behind are the individual targets, and the dashed line is the original model measured on the targets.

baselines across 20 targets on Phi-3-mini-4k-instruct. Unlike other methods that merely suppress surface forms, CRU is the only approach that truly removes target knowledge across all three probing levels: fill-in-the-blank (Level 1), QA (Level 2), and adversarial prompts (Level 3). Averaged over all levels, CRU reduces forget ROUGE-L recall from 0.532 (original model) to 0.048, outperforming SoTA and resisting adversarial extraction. Alas, this comes with a modest drop in utility on MMLU and a slightly higher vulnerability to membership inference attacks, especially in generation (§D).

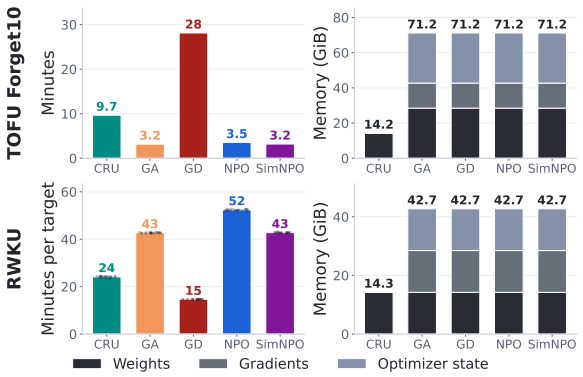
#2 *CRU's memory footprint is lower than all baselines. Still, CRU has comparable runtimes to SoTA and is orders of magnitude faster than retraining.*[2]

CRU is cheap. By eliminating gradients and optimizer states for the base model, CRU significantly reduces memory overhead compared to baselines. Figure 6b reports fp32 runtime and peak memory on Phi (excluding prompt-only ICU and PURGE, evaluated via published checkpoints). CRU

[2]According to several reports on the training of 7B+ parameter LLMs, such as the technical report of DeepSeek-V3 (DeepSeek-AI et al., 2025), which lists a training time of 54 days.



(a) Rouge-L (lower is better) for random neuron selection on `TOFU forget01`.



(b) Resources computed on Phi with fp32 tensors.

Figure 6: Random neuron selection (left), and runtime and peak memory (right).

demonstrates a significant memory advantage, using 14.2 GiB on `TOFU forget10` and 14.3 GiB on `RWKU`, versus 71.2 GiB and 42.7 GiB for all SoTA. CRU remains competitive in runtime by requiring 24 minutes per target (∼0.5 minutes for localization, the rest for routing) on `RWKU` compared to 52 minutes for NPO and 43 minutes for GA/SimNPO. On `TOFU`, CRU takes 9.7 minutes, trailing preference-based methods (3.2–3.5 minutes). We show full runtimes in Figure 13.

5.1.1 Ablations

#3 *Both phases of CRU are essential. Variance selection of Phase 1 accurately isolates concept-specific neurons. Phase 2 effectively discriminates between forget and retain sets.*

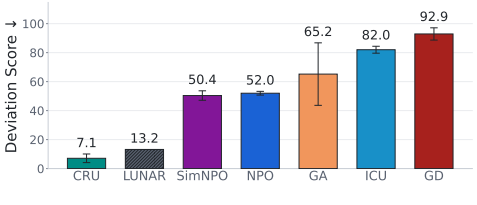


Figure 7: ROUGE Deviation Score (lower is better) on Qwen 2.5-7B, on the `TOFU` split defined by LUNAR.

Table 1: CRU variants over p on `TOFU forget01` (extrapolated from Table 4). Forget is measured as $(1 - \text{Rouge-L})$.

	Qwen		Llama		Phi		DeepSeek	
Setting	Forget↑	MU↑	Forget↑	MU↑	Forget↑	MU↑	Forget↑	MU↑
Original	0.04	0.48	0.01	0.52	0.07	0.41	0.25	0.42
$p = 85$	**0.99**	**0.55**	0.31	0.53	0.58	0.30	**0.79**	0.35
$p = 90$	0.98	0.53	0.25	0.53	0.42	0.36	0.68	**0.36**
$p = 95$	0.98	0.50	**0.67**	0.55	0.39	**0.38**	0.57	0.35
$p = 99$	0.97	0.45	0.54	**0.56**	**0.72**	0.08	0.68	0.34

The neurons we gate are concept-specific, not generically noisy. Because variance is partly an inherent property of individual neurons, Phase 1 might face the criticism that it merely selects units that are generically noisy across all inputs, rather than those tied to the target concept. To test this, we run Phase 1 twice on frozen weights – i.e., once on 40 forget-set examples ($\mathcal{C}_k^f$) and once on 40 retain-set examples ($\mathcal{C}_k^r$) – and compute their overlap coefficient $|\mathcal{C}_k^f \cap \mathcal{C}_k^r|/|\mathcal{C}_k^f|$ (Figure 8). Across gated blocks, the selected sets share only a minority of neurons: 27% on Llama, 38% on DeepSeek, 46% on Phi, and 59% on Qwen, with un-thresholded variance rankings showing similarly low-to-moderate Spearman correlations ($\rho = 0.18, 0.37, 0.44, 0.46$). Thus, the majority of gated neurons are specifically driven by the forget set. The remaining generic minority reflects baseline neuron variance, which likely accounts for the minor utility trade-offs observed on `RWKU`.

Random neuron selection is not effective (and, rightly so). We choose a random number of neurons in the same layers and of the same number that CRU would have selected through variance (Figure 6a). We compute the Forget set Rouge-L (lower is better) on `TOFU forget10`. Results are

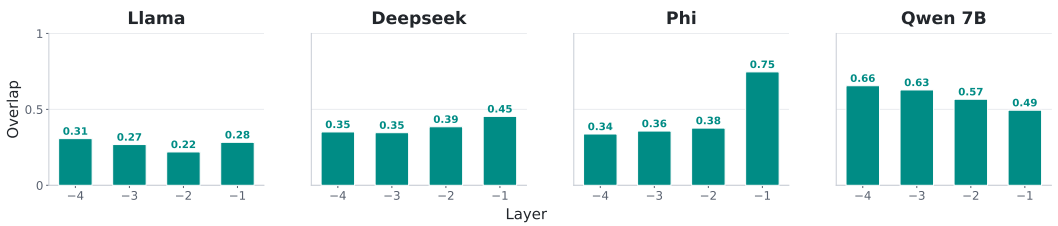


Figure 8: **Neuron overlap for the forget and retain sets.** Each bar is the fraction of the neurons in C_k^f that are also in C_k^r, $|C_k^f \cap C_k^r|/|C_k^f|$: how many of the gated neurons would have been picked without ever showing the model the forget set. Compued on `TOFU Forget01` with $p = 95$.

unambiguous: i.e., random selection with the same gating always performs worse, very close to the original model, as if no unlearning was done, across all models.

Gating fewer neurons does not produce a gentler intervention. Intuitively, adjusting the percentile threshold p to restrict $|\mathcal{C}_l|$ should yield a monotonic trade-off between forgetting and utility. However, on `RWKU`, $p = 90$ achieves the best performance on both axes (forget 0.055, utility 0.418), whereas $p = 99$, which gates only a fifth as many neurons, degrades both metrics (forget 0.101, utility 0.354). This occurs because the router compensates for the restricted set of neurons by closing the remaining gates more aggressively. The same non-monotonic pattern holds on `TOFU`, where Phi's utility drops to 0.08 at $p = 99$ compared to 0.38 at $p = 95$ (Table 1; see also Figure 9 and table 4). In short, restricting where suppression can land concentrates the intervention.

6 Conclusion

CRU is a streamlined and lightweight unlearning method that applies seamlessly to *any* LLM. One untrained forward pass over the forget set ranks neurons by how much their pooled activations vary, and small routing modules, then gate those neurons per token, closing on forget inputs and passing on retain ones. The base model is frozen throughout, so every change in behavior is attributable to the gated neurons rather than to parameter drift.

Our analysis shows three advantages. First, high forgetting scores are not evidence of unlearning: on `TOFU forget10`, GA, NPO, and SimNPO report up to 0.97 forgetting at null utility, which is a broken model reported as a success, and any comparison that does not read the two axes together will reward that. Second, suppressing a surface form is not the same as removing what produces it. Every baseline we tested loses ground as the probe gets harder, while CRU holds at 0.052 adversarial recall, compared to 0.250 for the next-best, which reflects the difference between knowledge that is gone and knowledge that is merely inconvenient to reach. Third, the cost of unlearning is mostly the cost of retraining: dropping gradients and optimizer states for the base model reduces the footprint from 71 GiB to 14 GiB at comparable runtime, putting per-concept unlearning within reach of a single consumer device.

Limitations. CRU leaves a few things open. Phase 1 uses variance as a cheap stand-in for attribution, and only a minority of the neurons it selects are generically variable rather than concept-specific. The gates withhold some activation on retain inputs, which accounts for the small utility drop we observe on `RWKU`. Our choice of the last four blocks follows the localization literature rather than our own criterion. Because each request trains its own routers on a frozen base, sequential and simultaneous unlearning of multiple concepts is a direction we leave for future work. We also leave the ablation of Equation (9) to support the choice of the two terms as a future avenue (§B).

Call-to-Action

We hope the broader point outlives our method. Most unlearning research has asked what to optimize and answered by changing the weights. We suggest that concepts can be reached via the neurons that express them, and that suppression can be determined per query rather than baked into the parameters. We encourage future work in unlearning to treat the scope of the intervention as a design choice in its own right.

AI USE STATEMENT

In this work, we used generative AI tools to formalize mathematical claims and to aid in implementing methods.

We have **not** used generative AI tools to assist with translation, develop theoretical models, provide critical ingredients for proving mathematical claims, provide feedback on research methodology, support qualitative and thematic data analysis, or interpret results. Finally, generating synthetic datasets, assisting in writing proofs, proposing hypotheses, and cleaning datasets are not applicable to this work.

Additionally, we used generative AI tools to create scientific figures and edit software code. We have reviewed all AI-assisted work: all co-authors have checked and revisited the text, including the mathematical formulations; LLM-generated code was verified and tested for correctness by 2 authors; and all figures were reviewed.

We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We made every effort to ensure the results were reproducible. Our repository is publicly available at `https://anonymous.4open.science/r/cru-iclr-4E56/README.md` and includes a detailed README with instructions for reproducing the experiments.

REFERENCES

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

California Department of Justice. California consumer privacy act (ccpa), 2026. URL `https://oag.ca.gov/privacy/ccpa`.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Hang Chen, Jiaying Zhu, Xinyu Yang, and Wenya Wang. Conflict-guided localization for LLM unlearning framework. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=jtRYvazBWv`.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Th6NyL07na`.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 933–941. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/dauphin17a.html`.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li,

Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL http://jmlr.org/papers/v7/demsar06a.html.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022. doi: 10.52202/068431-2198. URL https://neurips.cc/virtual/2022/poster/53018.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

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 *Neurips Safe Generative AI Workshop 2024*, 2024a. URL https://openreview.net/forum?id=pVACX02m0p.

Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *International Conference on Learning Representations*, volume 2024, pp. 53643–53673, 2024b.

GDPR Info. Article 17 – right to erasure ("right to be forgotten"), 2026. URL https://gdprinfo.eu/en-article-17.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

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

Lishuai Hou, Zixiong Wang, Gaoyang Liu, Chen Wang, Wei Liu, and Kai Peng. Decoupling memories, muting neurons: Towards practical machine unlearning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13978–13999, 2025.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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.

Jinghan Jia, Jiancheng Liu, Yihua Zhang, Parikshit Ram, Nathalie Baracaldo, and Sijia Liu. Wagle: Strategic weight attribution for effective and modular unlearning in large language models. *Advances in Neural Information Processing Systems*, 37:55620–55646, 2024.

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

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International conference on learning representations*, volume 2025, pp. 90960–90985, 2025.

Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning. In *Conference on Lifelong Learning Agents*, pp. 243–254. PMLR, 2022.

Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Yuguang Yao, Chris Yuhao Liu, Xiaojun Xu, Hang Li, et al. Rethinking machine unlearning for large language models. *Nature Machine Intelligence*, 7(2):181–194, 2025.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. 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. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Aashiq Muhamed, Jacopo Bonato, Mona T Diab, and Virginia Smith. Saes can improve unlearning: Dynamic sparse autoencoder guardrails for precision unlearning in llms. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025.

Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. In-context unlearning: language models as few-shot unlearners. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

William F. Shen, Xinchi Qiu, Meghdad Kurmanji, Alex Iacob, Lorenzo Sani, Yihong Chen, Nicola Cancedda, and Nicholas D. Lane. LLM unlearning via neural activation redirection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=teB4aqJsNP.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. In *International Conference on Learning Representations*, volume 2025, pp. 27797–27818, 2025.

Niklas Stoehr, Kevin Du, Vésteinn Snæbjarnarson, Robert West, Ryan Cotterell, and Aaron Schein. Activation scaling for steering and interpreting language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8189–8200, 2024.

Tianyi Tang, Wenyang Luo, Haoyang Huang, Dongdong Zhang, Xiaolei Wang, Wayne Xin Zhao, Furu Wei, and Ji-Rong Wen. Language-specific neurons: The key to multilingual capabilities in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5701–5715, 2024.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4593–4601, 2019.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX security symposium (USENIX Security 22)*, pp. 4007–4022, 2022.

William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. 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. 4527–4546, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.372. URL https://aclanthology.org/2021.emnlp-main.372/.

Vinícius Conte Turani, Otávio Parraga, João Vitor Boer Abitante, Kristen K Arguello, Joana Pasquali, Ramiro N Barros, Flavio du Pin Calmon, Christian Mattjie, Rodrigo C Barros, and Lucas S Kupssinskü. Inference-time machine unlearning via gated activation redirection. *arXiv preprint arXiv:2605.12765*, 2026.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11132–11152, 2022.

Yuanshun Yao and Xiaojun Xu. Large language model unlearning. *Advances in Neural Information Processing Systems*, 37:105425–105475, 2024.

Sangyeon Yoon, Yeachan Jun, and Albert No. Position: The term "machine unlearning" is overused in LLMs. In *Forty-third International Conference on Machine Learning Position Paper Track*, 2026. URL https://openreview.net/forum?id=GHUiYV4LTc.

Zeping Yu and Sophia Ananiadou. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3267–3280, 2024.

Efstratios Zaradoukas, Bardh Prenkaj, and Gjergji Kasneci. Reinforcement unlearning via group relative policy optimization. In *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 20–24, 2026*. OpenReview.net, 2026. URL https://openreview.net/forum?id=BjWwqPE7mk.

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.

A WHAT SOTA DOES WITH EQUATIONS EQUATION 2 AND EQUATION 3

A.1 THE OBJECTIVE

Recall Equation (2). GA sets $\ell_f = -\ell$ with $\lambda = 0$ (Yao & Xu, 2024); GD keeps $\ell_f = -\ell$ and takes $\lambda > 0$ (Liu et al., 2022). Preference-based objectives replace $-\ell$, which is unbounded below, with a saturating alternative. Writing σ for the logistic function and y_{idk} for a refusal response, the DPO formulation prefers refusal over the true completion (Rafailov et al., 2023),

$$\ell_f^{\mathrm{DPO}}(y \mid x; \theta) = -\log \sigma\left(\beta \log \frac{\pi_\theta(y_{\mathrm{idk}} \mid x)}{\pi_{\mathrm{o}}(y_{\mathrm{idk}} \mid x)} - \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{o}}(y \mid x)}\right), \tag{10}$$

whereas NPO retains only the dispreferred branch (Zhang et al., 2024),

$$\ell_f^{\mathrm{NPO}}(y \mid x; \theta) = -\frac{2}{\beta} \log \sigma\left(-\beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{o}}(y \mid x)}\right) = \frac{2}{\beta} \log\left(1 + \left[\frac{\pi_\theta(y \mid x)}{\pi_{\mathrm{o}}(y \mid x)}\right]^{\beta}\right), \tag{11}$$

which recovers GA ascent as $\beta \to 0$ but, being bounded below by zero, diverges exponentially more slowly. SimNPO discards the reference model and normalizes by response length (Fan et al., 2024a),

$$\ell_f^{\mathrm{SimNPO}}(y \mid x; \theta) = -\frac{2}{\beta} \log \sigma\left(-\frac{\beta}{|y|} \log \pi_\theta(y \mid x)\right). \tag{12}$$

Throughout, $\beta > 0$ is the inverse temperature of Equations (10) to (12) and λ the forget/retain weight of Equation (2).

The second template replaces likelihood targets with a scalar reward on sampled completions. With a reward $r(x, y)$ that penalizes any mention of the forbidden concept, unlearning becomes a KL-constrained policy-optimization problem (Zaradoukas et al., 2026),

$$\max_{\theta \in \Theta} \; \mathbb{E}_{x \sim \mathcal{D}_f} \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[r(x, y)] \; - \; \lambda_{\mathrm{KL}} \, \mathbb{E}_{x \sim \mathcal{D}_f}[\mathrm{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{o}}(\cdot \mid x))], \tag{13}$$

where the KL term plays the utility-preserving role that $\mathbb{E}_{\mathcal{D}_r}[\ell_r]$ plays in Equation (2), and advantages are estimated group-relatively over completions sampled per prompt rather than by a learned critic.

The third template does not optimize at all. Prompt-based methods leave the weights fixed and act on the context: for a context c,

$$\theta_{\mathrm{u}} = \theta_{\mathrm{o}}, \qquad \pi_{\mathrm{u}}(y \mid x) \; = \; \pi_{\mathrm{o}}(y \mid c \oplus x), \tag{14}$$

with $\oplus$ denoting concatenation (Pawelczyk et al., 2024). The consequence is visible in the notation: since $\theta_{\mathrm{u}} = \theta_{\mathrm{o}}$, forgetting is a property of the session rather than of the model, and any measurement taken with c removed returns π_{o} exactly.

A.2 THE SCOPE, AND WHERE CRU SITS

What $\mathbf{w}$ is allowed to hold. Equation (3) writes a parameter-space unlearning operator as $\theta_u = \theta_o + \mathbf{w} \odot \Delta$, and its two factors divide the labor: Δ is where the objective of §A.1 ends up, and $\mathbf{w}$ is everything a method has to say about scope. Read this way, the literature settles $\mathbf{w}$ in one of two ways. Global methods (GA, GD, NPO, SimNPO, PURGE) fix $\mathbf{w} = \mathbf{1}$ and leave scope to the objective. Localized ones (SalUn, WAGLE, CLUE, LUNAR) choose $\mathbf{w} \neq \mathbf{1}$ based on saliency, influence attribution, circuit discovery, or layer selection. In both cases, $\mathbf{w}$ is a constant vector, computed once while unlearning and applied unchanged to every query that follows. Equation (3) has no slot in which to write a quantity that depends on the input, so a method that wants one cannot be written in it at all.

CRU leaves the equation empty. CRU sets $\mathbf{w} \odot \Delta = \mathbf{0}$, so that $\theta_u = \theta_o$ exactly, and every weight of the base model survives unlearning bit for bit. What replaces $\mathbf{w}$ is a map from a hidden state to a full-width multiplier. For block l, with concept set $\mathcal{C}_l = \{j_1 < \cdots < j_{|\mathcal{C}_l|}\}$ and router parameters $\phi_l = (\mathbf{w}_1, \mathbf{b}_1, \mathbf{w}_2, \mathbf{b}_2)$, define $U_l : \mathbb{R}^d \to [0, 1]^d$ by

$$U_l(\mathbf{h})[j] \; = \; \begin{cases} R_l(\mathbf{h})[m], & j = j_m \in \mathcal{C}_l, \\ 1, & j \notin \mathcal{C}_l, \end{cases} \qquad R_l(\mathbf{h}) = \sigma(\mathbf{w}_2 \, \mathrm{ReLU}(\mathbf{w}_1 \mathbf{h} + \mathbf{b}_1) + \mathbf{b}_2), \tag{15}$$

so that Equation (8) is the product

$$\mathbf{h}'_{l,t} \;=\; U_l(\mathbf{h}_{l,t}) \odot \mathbf{h}_{l,t}. \tag{16}$$

U_l is determined by two things of different kinds. $\mathcal{C}_l$ is a constant of the map, fixed once per concept in Phase 1 and never differentiated; ϕ_l is trained. Coordinates outside $\mathcal{C}_l$ are pinned to 1 by construction, which is what makes the set a hard restriction on where suppression may land. The map carries no dependence on t: the same U_l is applied at every position.

Equation (16) is Equation (3) with three substitutions. The operand is an activation rather than a parameter vector; the product runs over model width at a single position rather than over $\dim \Theta$; and the multiplier is the image of a function rather than a stored vector. **Only the third is the claim of this paper.** The first two say where we act, and one could imagine acting there with a constant; the third says that what we do once we are there is not fixed until the query arrives.

A fixed selector and a learned one. CRU does not abolish the constant selector of Equation (3) so much as demote it. The set $\mathcal{C}_l$ is computed once per concept and is as fixed thereafter as any $\mathbf{w}$; what changes is its authority. It decides where suppression is permitted to land, never how much of it lands, and a neuron inside $\mathcal{C}_l$ is only a candidate. The magnitude is settled per token by R_l, which is why the same neuron can be closed at one position and passed at the next (§4.2).

Special cases. Because U_l is a function, the frozen model is recovered by restricting the class it is drawn from, and the restrictions form an order.

- $U_l \equiv \mathbf{1}_d$ returns the base model.
- U_l constant in $\mathbf{h}$, with entries in $\{0, 1\}$, is a fixed hard mask over a neuron subset, which is basically NeuMuter.
- U_l piecewise constant, taking $\mathbf{1}_d$ or a tuned magnitude α according to whether a similarity $s(\mathbf{h})$ clears a grid-searched threshold τ, gives CAST and DSG.
- U_l smooth and learned, as in Equation (15) (i.e., Equation (8)), is CRU.

The ordering is by how much of U_l is decided before the model is ever queried. The rule-based methods fix the magnitude and condition the trigger; Equations (8) and (15) learns both.

What this costs and returns. Since Δ is never formed, the unlearning operator is $\mathcal{U}(\pi_o, \mathcal{D}_f, \mathcal{D}_r) = (\theta_o, \phi)$. No gradient or optimizer state is ever allocated for θ_o, which is the whole of the memory result in §5.1 rather than an implementation detail. The operator also has an inverse, whose methods that write into θ do not. In other words, deleting ϕ returns π_o, not an approximation of it.

Why this is not ICU. Prompt-based unlearning also leaves $\theta_u = \theta_o$. Nevertheless, per Equation (14), the intervention lives in the context c, so forgetting is a property of the session. In Equation (16), the intervention lives in ϕ, which travels with the weights and not the input.

B On the shape of the two loss terms

Equation (9) pairs a linear forget term with a quadratic retain term. The asymmetry is not forced, so this section gives the reasons for it: what each shape does to the gradient reaching the router (§B.1), and what the four combinations predict for a router that cannot separate the two corpora (§B.2).

B.1 Gradients

Write z_n for the pre-activation of the router's output entry that produces gate $w_n = \sigma(z_n)$, and let $\bar{w}_l = \frac{1}{N}\sum_{n=1}^{N} w_n$ be the block mean over the N selected-neuron, non-padding entries of a batch. Every shape we consider reaches z_n through the same Jacobian,

$$\frac{\partial w_n}{\partial z_n} \;=\; w_n(1 - w_n), \tag{17}$$

so the shapes differ only in the scalar multiplying it. For the forget term,

$$\frac{\partial\,\bar{w}_l}{\partial z_n} = \frac{1}{N}\,w_n(1-w_n), \qquad \frac{\partial\,\bar{w}_l^{\,2}}{\partial z_n} = 2\bar{w}_l \cdot \frac{1}{N}\,w_n(1-w_n), \tag{18}$$

and for the retain term, with target 1,

$$\frac{\partial\,(1-\bar{w}_l)}{\partial z_n} = -\frac{1}{N}\,w_n(1-w_n), \qquad \frac{\partial\,(\bar{w}_l-1)^2}{\partial z_n} = 2(\bar{w}_l-1) \cdot \frac{1}{N}\,w_n(1-w_n). \tag{19}$$

The retain term should vanish at its target; the forget term should not. The two terms are not two instances of the same kind of goal. $\mathcal{L}_{\text{suppress}}$ has no target. In other words, a closed gate is better than a half-closed one, and there is no value of $\bar{w}_l^{\,f}$ at which the method would prefer to stop. $\mathcal{L}_{\text{pass}}$ has one, i.e., $\bar{w}_l^{\,r} = 1$, at which the router is passing retain activations unchanged. A penalty on a constraint should therefore release its gradient at the constraint, which Equation (19) does in the quadratic case and does not in the linear one: a linear retain term pushes toward 1 with undiminished force at $\bar{w}_l^{\,r} = 0.999$, so the equilibrium between the two terms would be set by their relative weight alone rather than by how far the retain gates have actually drifted. The quadratic makes retain pressure a function of the violation, which is the standard reason for a squared penalty and the reason here.

Why the forget term is not squared. The same reasoning does not transfer to $\mathcal{L}_{\text{suppress}}$, because the router already saturates on its own. The factor $w_n(1-w_n)$ of Equation (17) vanishes as a gate shuts, so the gradient reaching a nearly closed gate is small before any choice of loss shape. Squaring the forget term rescales it by a further $2\bar{w}_l^{\,f}$, a second factor that vanishes with the first: at the block means we observe on `TOFU forget10` the rescaling is $2\bar{w}_l^{\,f} \in [0.02, 0.12]$ (Figure 3), between roughly $8\times$ and $50\times$ smaller than the linear term depending on the model, and shrinking further as the gate approaches zero. The linear term holds the multiplier at 1 and leaves only the sigmoid's own saturation. Linearity is affordable here in a way it is not for gradient ascent (§A.1): since $\bar{w}_l^{\,f} \in [0,1]$, $\mathcal{L}_{\text{suppress}}$ is bounded below by construction.

Consequences for what the gates settle at. Because $\mathcal{L}_{\text{pass}}$ relaxes near 1 while $\mathcal{L}_{\text{suppress}}$ does not relax near 0, and the two act on the same router parameters over different batches, the retain gates come to rest short of their target: Figure 3 reports 0.79–0.85 rather than 1. This is the intended trade and not a training failure. The withheld activation is bounded by construction, since the base weights are untouched and the gate lies in $[0,1]$, so its cost appears as the small utility drop in `RWKU`, rather than as parameter drift. A linear retain term would hold the gates closer to 1, at the cost of making that pressure insensitive to how much was withheld.

A note on one-sidedness. Since $w_n \in [0,1]$ we have $\bar{w}_l^{\,r} \le 1$ throughout, so $(\bar{w}_l^{\,r}-1)^2 = (1-\bar{w}_l^{\,r})^2$ and the square never penalises overshoot. It is a one-sided penalty in effect, and the squared form is chosen for the vanishing gradient at the target rather than for two-sidedness.

B.2 What each combination predicts for a collapsed router

Proposition 1 treats the hypothesis that the router cannot distinguish the two corpora, so that a single gate value w is returned on both. The prediction it makes depends on the shapes, and not every pairing makes one.

Proposition 2. Let $\mathcal{L}(w) = f(w) + \lambda\, r(w)$ be the collapsed loss under a constant gate w, with $\lambda > 0$. Then:

1. $f(w) = w$, $r(w) = (w-1)^2$: $\mathcal{L}$ is strictly convex and minimized at $w^\star = 1 - \frac{1}{2\lambda}$, which lies in $(0,1)$ iff $\lambda > \frac{1}{2}$.
2. $f(w) = w^2$, $r(w) = (w-1)^2$: $w^\star = \frac{\lambda}{1+\lambda} \in (0,1)$ for every $\lambda > 0$.
3. $f(w) = w$, $r(w) = 1-w$: $\mathcal{L}(w) = (1-\lambda)w + \lambda$ is affine, so the minimum over $[0,1]$ is attained at an endpoint according to the sign of $1-\lambda$, and every w is optimal at $\lambda = 1$.

4. $f(w) = w^2$, $r(w) = 1 - w$: $w^\star = \frac{\lambda}{2}$, which lies in $(0, 1)$ iff $\lambda < 2$.

Proof. Each $\mathcal{L}$ is an affine or quadratic polynomial in w; set $\mathcal{L}'(w) = 0$ and check the sign of $\mathcal{L}''$. In case 3, $\mathcal{L}'' = 0$ and the minimum is attained at an endpoint of $[0, 1]$. □

Case 3 is the reason we would not use a linear retain term, even setting §B.1 aside. With both terms linear, the collapsed loss is affine, its optimum sits at a boundary of the box, and at $\lambda = 1$ the loss is constant in w: the null model predicts nothing, and a measurement of the gate means cannot be compared against it. Case 1, the pairing we use, gives the null an interior value (0.5 at $\lambda = 1$) that our measurements contradict by more than an order of magnitude on the forget side. Case 2 also gives an interior value, but at the cost of the gradient rescaling of Equation (18). Case 4 makes the location of the null grow linearly in λ and leaves $[0, 1]$ at $\lambda = 2$, so for the retained weights, we use it to predict a boundary again.

What we have and have not shown. The argument above is analytical. It says what each pairing does to the gradient reaching the router and what each predicts for a collapsed router, and on both counts the linear-quadratic pairing is the one we can reason about: the forget term avoids compounding the sigmoid's own saturation, the retain term releases its gradient at the constraint it encodes, and the null model of Proposition 1 lands at an interior value our measurements contradict. We have not run the corresponding ablation, so we do not claim that the alternatives fail empirically, nor that CRU's results depend on this choice. The gate statistics we report Figure 3) are consistent with the mechanism described here, but do not discriminate among the four shapes; a sweep over them is the natural next check.

C Detailed Experimental Settings

Hardware. We carry out all our experiments on a single NVIDIA GB10 Grace Blackwell system equipped with a 20-core Arm CPU and 128 GB of unified memory shared between the CPU and the integrated Blackwell GPU.

Models. Table 2 shows the family, number of parameters, number of layers, and context (in terms of thousands of tokens) of the models we employ in our experimental settings.

Benchmark metrics. Table 3 shows all the metrics evaluated for each dataset. For `TOFU`, utility metrics are evaluated for three sets: the retain set, i.e., the training set without the forget set, and two held-out sets called "Real Authors" and "World Facts" (Maini et al., 2024). These are not part of the model's fine-tuning and contain information assumed to be already known.

Table 2: The four model families we evaluate on. Sizes are the total parameter counts of the released checkpoints.

Short name	Family	Params	Layers	Context
DeepSeek	DeepSeek LLM	6.9B	30	4K
Qwen	Qwen2.5	7.6B	28	32K
Llama	Llama 3.1	8.0B	32	128K
Phi	Phi-3	3.8B	32	4K

Table 3: Evaluation metrics of the two benchmarks.

Metric	What it measures
`TOFU`	
Probability	Normalized likelihood $P(a \mid q)^{1/\lvert a\rvert}$ of the ground-truth answer (multiple-choice variant on Real Authors / World Facts). Forget ↓, utility ↑.
ROUGE-L recall	Overlap between the generated answer and the reference answer. Forget ↓, utility ↑.
Truth Ratio	Likelihood of perturbed (false) answers over that of a correct paraphrase: how much the model now prefers a wrong answer. Forget ↑, utility ↓.
Forget Quality ↑	KS-test p-value against a model retrained on the retain set only: high p = indistinguishable from never having seen the forgotten data.
Model Utility ↑	Harmonic mean of the nine utility scores (probability, ROUGE-L, rescaled truth ratio × retain / Real Authors / World Facts).
`RWKU`	
Forget FB / QA / AA ↓	ROUGE-L recall on cloze, question, and adversarial (jailbreak-style) probes about the target; AA exposes knowledge that is only suppressed at the surface.
Neighbor FB / QA ↑	The same probes on closely related entities that must be kept: edit locality.
MIA (LOSS, Zlib, Min-K%++)	Membership-inference scores on held-out fragments about the target (FM ↑) and about unrelated subjects (RM, unchanged); the AUC separating the two should return to chance ($\to 0.5$).
Utility ↑	MMLU 5-shot accuracy (general), BBH 3-shot CoT exact match (reasoning), TruthfulQA MC1 (truthfulness), TriviaQA F1 (factuality) and AlpacaEval n-gram entropy (fluency, i.e. non-degenerate text).

Comparison with LUNAR. The authors of LUNAR, in their original paper (Shen et al., 2025), use a custom split of TOFU that is not part of the original benchmark. Specifically, they use a forget set consisting of a single author; we conformed to this setting by creating an equal custom split on which we evaluate their own metric, *deviation score*. The deviation score is defined as:

$$\mathrm{DS} = 100 \times \sqrt{\mathrm{ROUGE1}_{\mathrm{forget}}^2 + \left(1 - \mathrm{ROUGE1}_{\mathrm{retain}}\right)^2}.$$

That is the Euclidean distance between a method's operating point and the ideal outcome of unlearning, at which the model reproduces none of the forget set ($\mathrm{ROUGE1}_{\mathrm{forget}} = 0$) while remaining

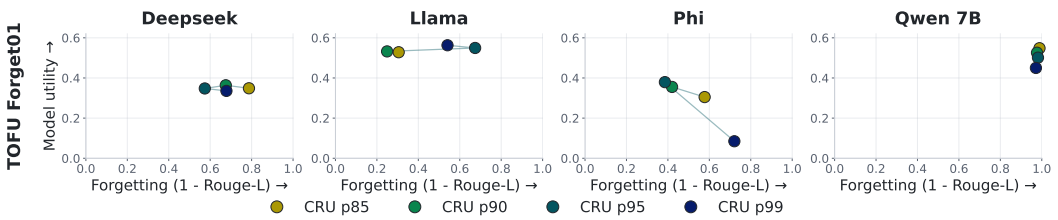


Figure 9: Hyperparameter p study for CRU on TOFU.

Table 4: CRU percentile ablation on TOFU forget01. "Original" is the model before unlearning. Best method per column, within each model, in bold.

		Unlearning Efficacy (Forget Set)			Utility Preservation									
					Real Authors			World Facts			Retain Set			
Model	Method	(1-Rouge-L)↑	(1-Prob.)↑	Truth ratio↓	Rouge-L↑	Prob.↑	Truth ratio↑	Rouge-L↑	Prob.↑	Truth ratio↑	Rouge-L↑	Prob.↑	Truth ratio↑	MU↑
Qwen	Original	0.04	0.01	–	0.83	0.31	0.32	0.94	0.37	0.29	0.98	0.99	0.59	0.48
	CRU p85	**0.99**	**1.00**	**0.23**	0.63	**0.38**	**0.53**	0.85	**0.39**	0.39	0.83	0.91	0.58	**0.55**
	CRU p90	0.98	1.00	0.34	0.74	0.36	0.44	**0.88**	0.37	0.32	**0.94**	**0.96**	**0.60**	0.53
	CRU p95	0.98	0.99	0.33	**0.77**	0.32	0.37	0.84	0.37	0.33	0.94	0.95	0.56	0.50
	CRU p99	0.97	0.92	0.37	0.41	0.37	0.49	0.52	0.37	**0.39**	0.51	0.53	0.58	0.45
Llama	Original	0.01	0.01	–	0.76	0.33	0.41	0.88	0.39	0.34	0.98	0.99	0.56	0.52
	CRU p85	0.31	0.23	0.49	0.74	0.34	0.44	0.86	0.38	0.35	**0.98**	**0.99**	**0.55**	0.53
	CRU p90	0.25	0.18	0.48	**0.77**	0.35	0.45	**0.88**	0.38	0.35	0.98	0.98	0.55	0.53
	CRU p95	**0.67**	**0.79**	**0.47**	0.53	**0.49**	**0.68**	0.79	**0.41**	**0.44**	0.59	0.66	0.55	0.55
	CRU p99	0.54	0.54	0.49	0.62	0.45	0.64	0.81	0.40	0.41	0.72	0.81	0.55	**0.56**
Phi	Original	0.07	0.14	–	0.41	0.31	0.24	0.80	0.33	0.32	0.95	0.82	0.40	0.41
	CRU p85	0.58	0.84	**0.24**	0.20	0.30	0.21	0.58	0.30	0.23	0.77	0.38	0.33	0.30
	CRU p90	0.42	0.71	0.33	0.27	0.31	0.24	0.73	0.33	0.30	0.81	0.40	**0.37**	0.36
	CRU p95	0.39	0.62	0.35	**0.30**	**0.31**	**0.26**	**0.79**	**0.33**	**0.32**	**0.85**	**0.53**	0.37	**0.38**
	CRU p99	**0.72**	**0.87**	0.32	0.01	0.30	0.24	0.13	0.32	0.27	0.18	0.11	0.37	0.08
DeepSeek	Original	0.25	0.33	–	0.79	0.32	0.28	0.89	0.32	0.28	0.67	0.64	0.42	0.42
	CRU p85	**0.79**	**0.89**	**0.28**	**0.83**	0.31	0.25	0.87	0.31	0.25	0.62	0.21	0.38	0.35
	CRU p90	0.68	0.87	0.31	0.76	0.32	0.27	0.87	**0.32**	**0.28**	0.62	**0.22**	0.41	**0.36**
	CRU p95	0.57	0.88	0.32	0.80	**0.32**	**0.27**	**0.88**	0.32	0.28	**0.66**	0.16	**0.43**	0.35
	CRU p99	0.68	0.83	0.34	0.63	0.32	0.27	0.85	0.31	0.27	0.52	0.17	0.40	0.34

unchanged on the retain set ($\mathrm{ROUGE1}_{\mathrm{retain}} = 1$). It is therefore a quantity to be minimized, with 0 denoting perfect unlearning and a fine-tuned model that has not been unlearned scoring close to 100. Following LUNAR, we score ROUGE-1 recall rather than the ROUGE-L recall used elsewhere in this work. We report the result on Qwen2.5-7B, the only model common to both papers.

Comparison with PURGE. For RWKU, we adapt our settings to the ones of PURGE (Zaradoukas et al., 2026) for a faithful comparison: the forget set is composed of one author at a time, and we report metrics that are the mean of 20 authors. Moreover, we use the publicly released model checkpoints from the original paper; their download is integrated into our evaluation code.

C.1 Hyperparameters

For the TOFU benchmark, we adhere to the recommended hyperparameters in the paper (Maini et al. (2024)) for all baselines. For CRU, we perform a small hyperparameter study on p in the setting of TOFU forget01, illustrated in Figure 9. All metrics for this same ablation are reported in Table 4.

For RWKU, we strictly adhere to the best hyperparameters found by PURGE Zaradoukas et al. (2026) for all baselines to ensure a fair comparison. The paper itself Jin et al. (2024) reports the number of warmup steps (20) before applying unlearning. For PURGE, since we use the released checkpoints, there are no hyperparameters to tune. For CRU, we perform a small grid search on the learning rate and the value of λ on a 3-identities held-out set. We sweep λ at the default $p = 95$ and then sweep p at the λ the first sweep picks.



Figure 10: TOFU's Forget Quality across all models and settings. Values above ≥ 0.05 are statistically indistinguishable from the Gold Model.

D Full Results on Benchmarks

D.1 TOFU

Figure 10 plots the Forget Quality from TOFU across all models and settings. The forget quality is essentially the p-value of the KS test against the Gold Model (see Table 3). On `forget01`, CRU achieves a Forget Quality above the $p = 0.05$ threshold for all models except Llama (which is still very close). Only NPO and SimNPO can equal or surpass CRU in this metric. On `forget10`, specifically Phi, CRU and GD are the only methods able to surpass the threshold.

D.2 RWKU

Figure 11 shows the hyperparameter selection for RWKU on Phi-3. Forgetting is saturated everywhere. Mean forget ROUGE-L stays between 0.055 and 0.101 across all nine configurations, compared to 0.484 for the original model, so it does not separate them, and the choice has to be made based on what that forgetting costs.

Along λ, mean utility rises from 0.333 at $\lambda = 1$ to 0.402 at 50 and then falls back to 0.377 at 100.

Along p, the value 90 is the best point on both axes simultaneously, with the lowest forget score (0.055) and the highest utility (0.418). The failure at $p = 99$ is the informative end of that sweep. Gating a fifth as many neurons does not make the intervention gentler, because the router compensates by closing harder on the units it still has, resulting in worse forgetting (0.101) and lower utility (0.354). Following this analysis, we select $\lambda = 50$ and $p = 90$ for testing the full set on RWKU for Phi-3.

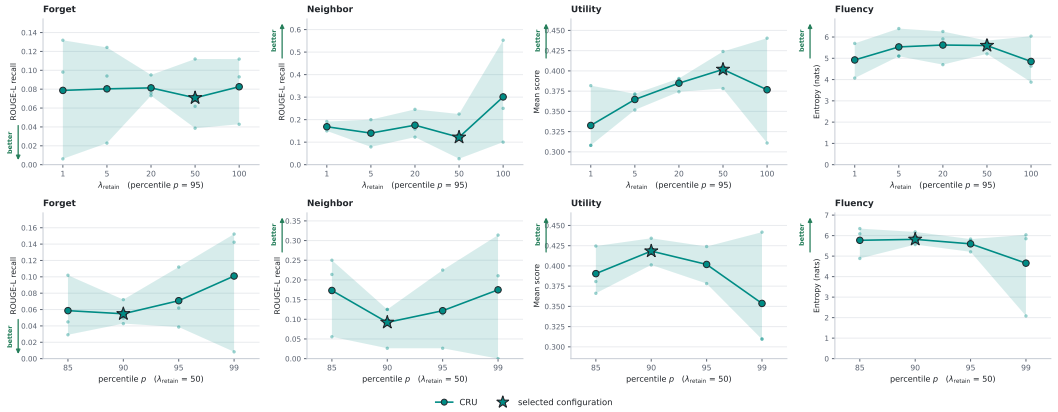


Figure 11: Hyperparameter selection for CRU on RWKU on Phi-3, measured on the three held-out targets (Beyoncé, Evel Knievel, Jim Morrison) that are disjoint from the test set. Top row: λ at percentile $p = 95$. Bottom row: p at $\lambda = 50$. The line is the mean over the three targets, the dots are the targets themselves, and the band is their range.

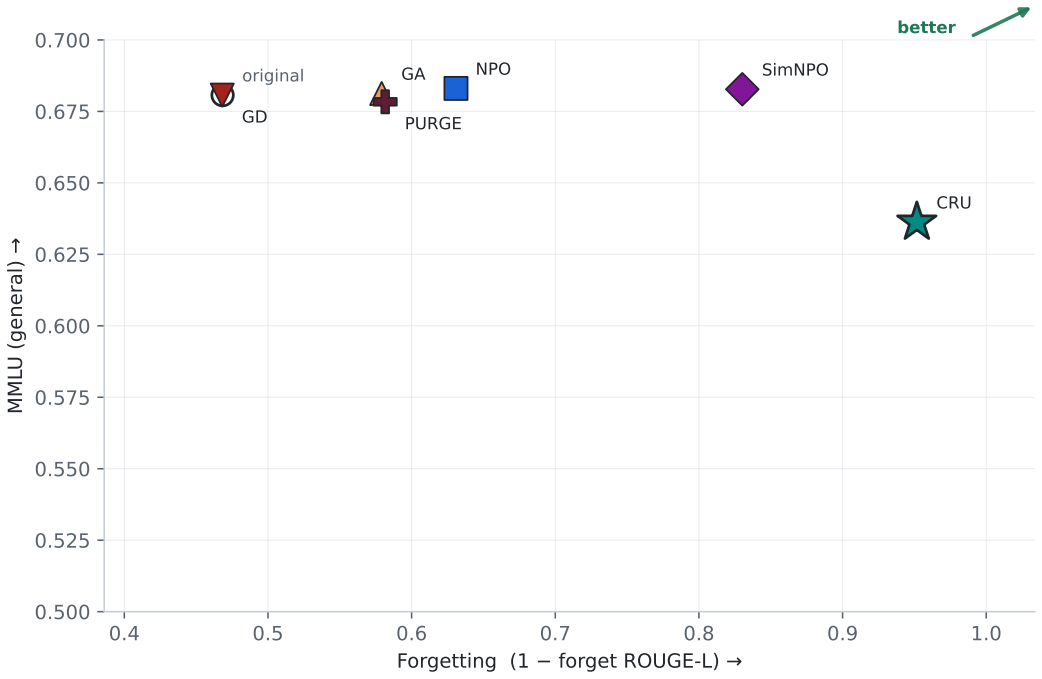


Figure 12: Forgetting against retained general knowledge on RWKU on Phi-3, both as means over the twenty targets. The open circle is the original model.

Figure 12 shows the tradeoff. CRU is alone on the right at 0.952 forgetting, with SimNPO next at 0.830 and everything else between 0.468 and 0.631. It pays for that with MMLU, at 0.636 against an original 0.681, while the baselines all stay within a point of the original. No method dominates CRU, in the sense that none reaches both more forgetting and higher MMLU.

CRU takes 24 minutes per target, against 52 for NPO, 43 for GA and SimNPO, and 15 for GD (Figure 6). Of our 24 minutes, localization accounts for about half a minute: the concept neurons come from a single forward pass over the forget corpus and train nothing, and the rest is spent fitting the routing modules. PURGE is not present here because we fetch the published checkpoints. Please refer to the original PURGE paper for running times (Zaradoukas et al., 2026).

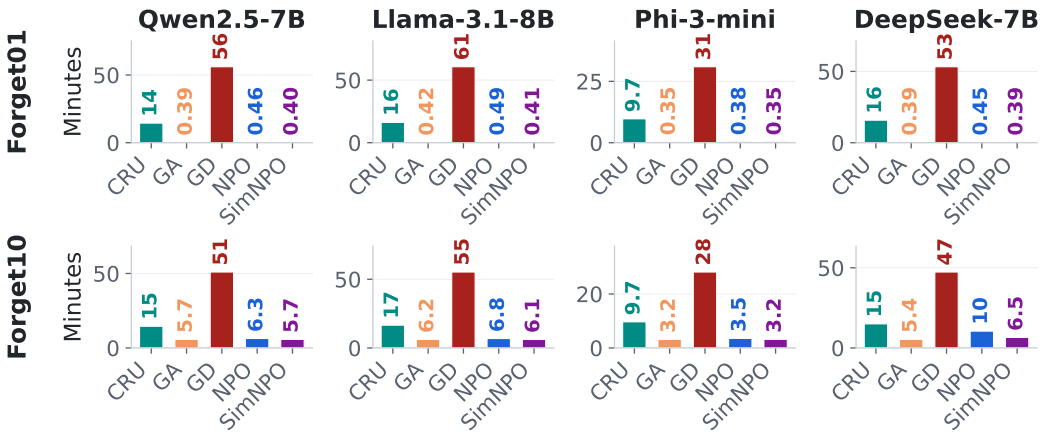


Figure 13: Runtime on both TOFU splits for all models.

D.2.1 RESOURCES

Figure 13 shows the full runtime results on both TOFU splits across all models. CRU is several times faster than GD, but slower than other baselines, especially on Forget01. However, as shown in §5.1, CRU is faster than most baselines on RWKU, and uses much less memory across all benchmarks. Moreover, using CRU is still orders of magnitude faster than retraining an LLM from scratch.

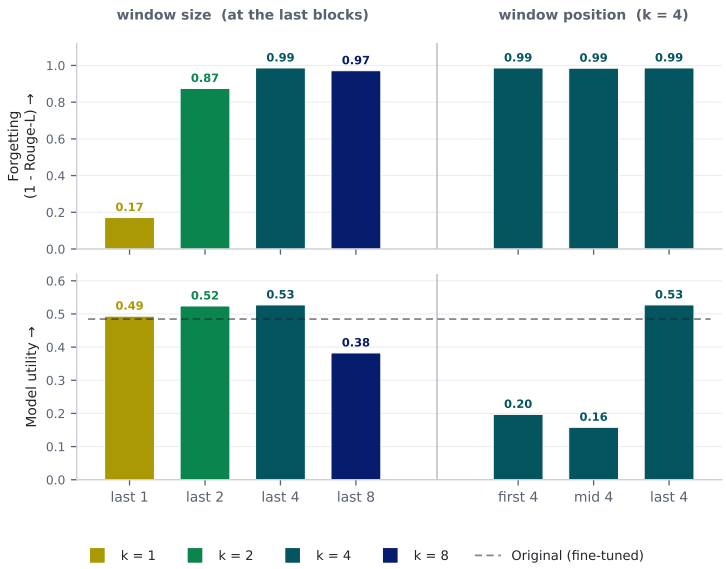


Figure 14: Ablation on the position and number of blocks where CRU selects neurons.

D.3 Full Ablations

D.3.1 Overlap of high-variance neurons

Phase 1 of CRU selects neurons by the empirical variance $\sigma_k^2(j)$ of their activations over the forget set. Variance, however, is in part a property of a neuron. A natural objection is therefore that CRU finds the generically high-variance units of layer k regardless of what we asked the model to forget. This would mean that the gate in Phase 2 succeeds by suppressing them rather than by suppressing the concept.

To disprove this, we run the following experiment: we run Phase 1 twice on the same fine-tuned weights, once on 40 forget-set examples and once on 40 retain-set examples, and measure the overlap in neurons that appear in both lists. Write C_k^f for the concept neuron set that CRU computes, over $n_f = 40$ forget-set examples, and C_k^r for the set that the identical procedure returns when those examples are replaced by $n_r = 40$ retain-set examples, with the same frozen fine-tuned weights, the same layer k, and the same α. We aim to compute the overlap coefficient $|C_k^f \cap C_k^r|/|C_k^f|$.

Figure 8 shows the results. Averaged over the K layers, the two lists share 27% of their neurons on Llama, 38% on DeepSeek, 46% on Phi, and 59% on Qwen: well short of being the same list. The same ordering appears without any threshold at all, in the rank Spearman correlation between the two sets of variances ($\rho = 0.18, 0.37, 0.44, 0.46$). Thus, we can claim that the vast majority of the neurons that CRU gates are specific to the forget set; a sizable minority is not, and depends on the inherent variance of each neuron.

D.3.2 Ablation on Block size and position

Figure 14 shows the ablation on the number and position of blocks where CRU selects the most variant neurons. The ablation was run on `TOFU forget10` on Qwen-2.5-7B. On the left column, Forgetting and Model Utility when selecting the last $\{1, 2, 4, 8\}$ blocks to select neurons. Our choice of $k = 4$ is the best both in Forgetting and Model Utility. Clearly, selecting more blocks (e.g., 8) means that neurons in earlier layers (and thus encoding more general knowledge) are gated and suppressed, thereby hindering the model's utility.

On the right, Forgetting and Model Utility when the window position changes. While Forgetting remains roughly the same (possibly due to model collapse and catastrophic forgetting), selecting the first 4 blocks or the middle 4 blocks severely hurts model utility, for the same reason described before. As such, we found that selecting the last $k = 4$ layers is optimal.

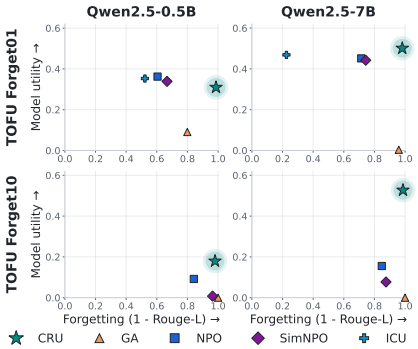


Figure 15: Ablation on the size of the model, using Qwen-2.5 with 7 billion parameters and half a billion parameters.

D.3.3 Ablation on Model Size

We also ablate the model size to see if CRU is succeeding only on big models, or if results can be generalized. To test this, we employ the same Qwen 2.5 model in both its 7-billion-parameter (used in our experiments) and 0.5-billion-parameter versions. We perform the test on both `TOFU` `forget01` and `forget10`. Results are visualized in Figure 15, following the same schema as Figure 4.

CRU has the best Forgetting-Utility tradeoff across all settings, for both models, and both `TOFU` splits, surpassing all baselines. The success of CRU is therefore not merely a byproduct of model size, but can be generalized to smaller models too.