K-FORGE: Fisher-Guided Initialization for Preference-Based LLM Unlearning

Daniil Moskovskiy
AIRI, Skoltech
Daniil.Moskovskiy@skol.tech

Alexander Panchenko
Skoltech, AIRI
A.Panchenko@skol.tech

Abstract

Preference-based LLM unlearning methods such as NPO and SimNPO must discover directions that suppress forget-set behavior without damaging retained capability. K-FORGE uses forget- and retain-set Fisher information to construct a retain-aware low-rank edit, then starts NPO or SimNPO from the edited checkpoint. On the TOFU `forget10` split, where 10% of the original dataset is designated for forgetting, stronger one-shot edits lower both Forget Q/A Probability and Model Utility. With Llama 3.2 1B, K-FORGE reduces Forget Q/A Probability by 22.3% under NPO while improving utility and using less total time and compute than scratch. SimNPO shows a smaller descriptive probability reduction with lower utility. On the same split, random and weight-SVD initializers yield higher forget probability than K-FORGE at 50 and 100 steps. Experiments with Gemma 3 1B also yield lower mean forget probability for both optimizers at 100 steps. MUSE results vary by metric, and direct relearning recovers forgotten behavior.

1 Introduction

LLM unlearning seeks to remove behavior attributable to specified training data without retraining from scratch (Maini et al., 2024; Li et al., 2024; Shi et al., 2025). Iterative preference-based methods such as NPO (Zhang et al., 2024) and SimNPO (Fan et al., 2024) repeatedly optimize a one-sided preference objective. Representation methods such as RMU (Li et al., 2024) instead redirect intermediate activations. Structured alternatives include diagonal-Fisher dampening (Foster et al., 2024), Fisher-weighted LoRA initialization (Cha et al., 2025), and few-step Kronecker-factored Gauss–Newton updates (McKinney et al., 2026).

The central difference is adaptivity. An iterative method can revise its direction after each update, whereas a one-shot method commits to one local surrogate before observing the resulting model behavior. In the reported one-shot sweep, increasing strength reduces forget probability while progressively spending utility. We therefore ask whether a closed-form edit is more useful as the starting direction for iterative unlearning than as the final unlearned model.

We combine K-FAC-style layerwise Fisher factors (Martens and Grosse, 2015) with Fisher-weighted low-rank factorization (Chekalina et al., 2026). We estimate one factor pair on forget data and one on retain data, simultaneously diagonalize the pairs, and derive a retain-aware Wiener edit in that basis. We call the resulting construction K-FORGE (Kronecker-Fisher One-shot Rank-r Generalized Erasure).

On TOFU `forget10` with Llama 3.2 1B Instruct, K-FORGE lowers the early Forget Q/A Probability of NPO and SimNPO. With the complete setup included in wall-clock time and FLOPs, the NPO comparison yields lower forget probability and higher utility; SimNPO has a smaller probability reduction with lower utility. Random and weight-SVD initializers yield higher forget probability than K-FORGE at 50 and 100 steps on this split. At 100 steps on Gemma 3 1B, both optimizers have lower mean forget probability. Qwen results are nearly unchanged, and MUSE improvements vary by metric.

Our contributions are three-fold:

1. We introduce K-FORGE, a closed-form Fisher-guided initializer for preference-based LLM unlearning. It uses separate forget and retain curvature estimates to construct a low-rank edit applied before NPO or SimNPO training.

2. We derive the exact full-rank solution of a two-Fisher Wiener objective and a separable relaxation for the general rank-constrained problem.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 4233–4248
November 6-10, 2026 ©2026 Association for Computational Linguistics

3. We show that K-FORGE improves the early Forget Q/A Probability trajectory of fixed NPO and SimNPO optimizers on TOFU `forget10`. We also measure setup time and FLOPs, evaluate model-family and benchmark transfer, and test relearning and quantized loading.

Code is available at `https://github.com/etomoscow/kron_unlearn`.

2 Related Work

LLM unlearning methods. Preference-based unlearning is the immediate setting for our work. NPO (Zhang et al., 2024) regularizes forget-set ascent through a reference-model preference term, while SimNPO (Fan et al., 2024) removes the reference model. Other families change the unlearning rule itself, for example by redirecting activations as in RMU (Li et al., 2024), combining forget and retain gradients, or subtracting task vectors. We initialize NPO and SimNPO with an edit computed from forget and retain curvature, then keep their training objectives unchanged. We evaluate on TOFU (Maini et al., 2024) and MUSE (Shi et al., 2025) using OpenUnlearning (Dorna et al., 2025).

Fisher-guided initialization and second-order unlearning. FILA (Cha et al., 2025) uses relative-Fisher weighted low-rank approximation to initialize LoRA; VILA (Kim et al., 2025) improves its estimation and efficiency; ReGLU (Xiao et al., 2026) combines representation-subspace initialization with an orthogonality loss. Other second-order methods change the update rule: SOUL (Jia et al., 2024) uses a Sophia-style optimizer (Liu et al., 2024), K-FADE (McKinney et al., 2026) uses Kronecker-factored Gauss–Newton steps, and WIN-U (Zhao et al., 2026) uses Woodbury-scaled Newton updates. K-FORGE derives a two-Fisher Kronecker Wiener edit with an exact full-rank solution and a separable low-rank relaxation, folds it into the base checkpoint, and leaves NPO or SimNPO unchanged.

Fisher information approximations. K-FAC (Martens and Grosse, 2015) and EK-FAC (George et al., 2018) approximate layer-wise Fisher information with Kronecker factors, making structured curvature computations tractable. Kronecker-Fisher approximations have also been used for influence functions at LLM scale (Grosse et al., 2023) and for compression via matrix-free Fisher factorization and GFWSVD (Chekalina et al., 2026). K-FORGE uses K-FAC-style covariance factors and extends the Fisher-weighted low-rank construction to separate forget and retain metrics.

3 Method

3.1 Preliminaries

Let $\mathbf{W} \in \mathbb{R}^{n\times m}$ denote the weight matrix of a single linear layer in a pretrained model with parameters $\theta^\star$. We denote by $\mathcal{L}_f(\theta)$ and $\mathcal{L}_r(\theta)$ the expected losses on a forget set $\mathcal{D}_f$ and a retain set $\mathcal{D}_r$, respectively. For a loss-bearing token t, let $\mathbf{x}_t \in \mathbb{R}^m$ be the layer input and $\mathbf{g}_t \in \mathbb{R}^n$ the gradient with respect to its output. The contribution of this application of $\mathbf{W}$ to its gradient is $\mathbf{g}_t\mathbf{x}_t^\top$. We use the K-FAC-style approximation (Martens and Grosse, 2015)

$$\begin{aligned} \mathcal{I}_F(\mathbf{W}) &\approx \mathbf{A}\otimes\mathbf{B}, \\ \mathbf{A} &= \frac{1}{N}\sum_{t=1}^{N}\mathbf{x}_t\mathbf{x}_t^\top, \\ \mathbf{B} &= \frac{1}{N}\sum_{t=1}^{N}\mathbf{g}_t\mathbf{g}_t^\top. \end{aligned} \tag{1}$$

We stream these covariances without materializing the $(mn)\times(mn)$ Fisher and mask padding, prompt, and other non-loss tokens. For N unmasked token positions, factor accumulation costs $\mathcal{O}(N(m^2+n^2))$ operations and $\mathcal{O}(m^2+n^2)$ memory; the subsequent dense factorizations cost $\mathcal{O}(m^3+n^3)$. We therefore evaluate and report the one-layer setup cost directly rather than extrapolating broad many-layer scaling.

In this work, we estimate the factors twice per layer: once on $\mathcal{D}_f$ to obtain $\mathbf{A}_f, \mathbf{B}_f$, and once on $\mathcal{D}_r$ to obtain $\mathbf{A}_r, \mathbf{B}_r$. Each $d\times d$ factor $\mathbf{C}$ is symmetrized and trace-scale damped as $\mathbf{C} \leftarrow \mathbf{C}+\varepsilon\max(\mathrm{tr}(\mathbf{C})/d, 10^{-12})\mathbf{I}$ before Cholesky factorization:

$$\mathbf{A} = \mathbf{L}_A\mathbf{L}_A^\top, \qquad \mathbf{B} = \mathbf{L}_B\mathbf{L}_B^\top. \tag{2}$$

For positive semidefinite empirical factors and $\varepsilon > 0$, this damping produces symmetric positive definite (SPD) factors in exact arithmetic. All subsequent factor symbols and quadratic norms refer to these damped matrices, which are held fixed while computing the edit.

We use float64 intermediates for covariance accumulators. Prompt and padding positions are as-

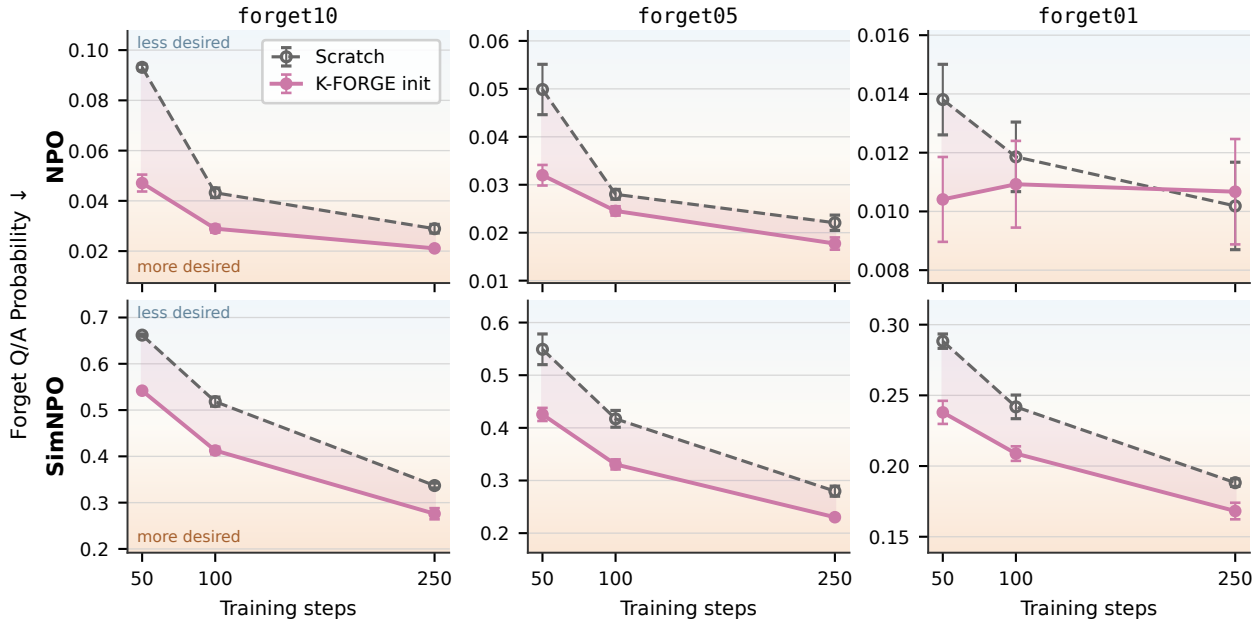


Figure 1: Matched downstream-step forget-probability trajectories on TOFU. Points show means with standard-deviation bars ($n = 3$); panels use split-specific y-axis ranges. Gains are clearest on `forget10`/`forget05` and for SimNPO; NPO on `forget01` is near the noise floor and reverses slightly at 250 steps. Additional initialization controls are reported in Tables 3 and 6.

signed label -100; after the causal shift, only positions with labels different from -100 are retained. The model loss is mean-reduced over valid target tokens, then multiplied by the number of valid target tokens before backpropagation. Thus $\mathbf{g}_t$ is the gradient with respect to the layer output at an unmasked causal-loss position. With factor normalization enabled, both covariance accumulators are divided by the total number of retained token rows.

We do not retry Cholesky factorization with an alternative numerical method. If an exception occurs while computing a module edit, that module is skipped and a warning is logged. We did not observe any skipped modules in our experiments.

We use column-wise vectorization: $\mathrm{vec}(\mathbf{X})$ stacks the columns of $\mathbf{X}$. The rank constraint is $\mathrm{rank}(\mathbf{\Delta}) \le r$, where $r \in \{0, \dots, \min(m, n)\}$. Throughout, $\mathrm{SVD}_r(\mathbf{X})$ denotes a best rank-at-most-r approximation in Frobenius norm.

3.2 Two-Fisher Wiener Edit

Following Fisher-weighted low-rank approximation, we treat weight components with high forget-Fisher energy as directions strongly expressed on the forget distribution. We therefore shrink the post-edit weight in that metric while penalizing movement in the retain metric. This is a local design surrogate, not a guarantee that answer probability decreases; the one-shot and initialization experiments test that link empirically:

$$J(\mathbf{\Delta}) = \|\mathbf{W} + \mathbf{\Delta}\|_{F_f}^2 + \lambda_{\mathrm{ret}} \|\mathbf{\Delta}\|_{F_r}^2, \qquad (3)$$

where $\|\mathbf{X}\|_{F_\bullet}^2 = \mathrm{vec}(\mathbf{X})^\top (\mathbf{A}_\bullet \otimes \mathbf{B}_\bullet)\,\mathrm{vec}(\mathbf{X})$. We define cross-Cholesky maps

$$\mathbf{K}_A = \mathbf{L}_{A_r}^{-1} \mathbf{L}_{A_f}, \qquad \mathbf{K}_B = \mathbf{L}_{B_r}^{-1} \mathbf{L}_{B_f}, \qquad (4)$$

with SVDs $\mathbf{K}_A = \mathbf{U}_A \mathbf{\Sigma}_A \mathbf{V}_A^\top$ and $\mathbf{K}_B = \mathbf{U}_B \mathbf{\Sigma}_B \mathbf{V}_B^\top$. The rotated forget-whitened weight is

$$\mathbf{R} = \mathbf{V}_B^\top \mathbf{L}_{B_f}^\top \mathbf{W} \mathbf{L}_{A_f} \mathbf{V}_A. \qquad (5)$$

Define the simultaneous diagonalizers $\mathbf{Q}_A = \mathbf{L}_{A_r}^{-\top} \mathbf{U}_A$ and $\mathbf{Q}_B = \mathbf{L}_{B_r}^{-\top} \mathbf{U}_B$, which jointly diagonalize $(\mathbf{A}_f, \mathbf{A}_r)$ and $(\mathbf{B}_f, \mathbf{B}_r)$, and let $\mathbf{H} = \mathbf{Q}_B^{-1} \mathbf{\Delta}\, \mathbf{Q}_A^{-\top}$ denote the perturbation in this basis.

Theorem 1 (Unconstrained K-FORGE Wiener edit)**.** *For $\lambda_{\mathrm{ret}} \ge 0$, the unique minimizer of Eq. 3 over $\mathbf{\Delta} \in \mathbb{R}^{n \times m}$ is*

$$\mathbf{\Delta}^\star = -\mathbf{L}_{B_r}^{-\top} \mathbf{U}_B (\mathbf{G} \odot \mathbf{R}) \mathbf{U}_A^\top \mathbf{L}_{A_r}^{-1}, \qquad (6)$$

where

$$G_{ji} = \frac{\sigma_{A,i} \sigma_{B,j}}{\sigma_{A,i}^2 \sigma_{B,j}^2 + \lambda_{\mathrm{ret}}}. \qquad (7)$$

For $r = \min(m, n)$ the rank constraint is inactive, so $\mathbf{\Delta}^\star$ is also the rank-r minimizer.

Proposition 1 (Exact rank-r solution at zero retain penalty)**.** *At $\lambda_{\mathrm{ret}} = 0$, an exact rank-$r$ minimizer of Eq. 3 is obtained by the rescaling $\mathbf{Y} = \mathbf{\Sigma}_B \mathbf{H} \mathbf{\Sigma}_A$, which converts the objective into the unweighted Frobenius problem $\|\mathbf{Y} + \mathbf{R}\|_F^2$. Solving via Eckart–Young, $\mathbf{Y}_r = \mathrm{SVD}_r(-\mathbf{R})$, unscaling $\mathbf{H}_r = \mathbf{\Sigma}_B^{-1} \mathbf{Y}_r \mathbf{\Sigma}_A^{-1}$, and mapping back yields*

$$\mathbf{\Delta}_0^{(r)} = -\mathbf{L}_{B_r}^{-\top} \mathbf{U}_B\, \mathbf{\Sigma}_B^{-1}\, \mathrm{SVD}_r(\mathbf{R}) \mathbf{\Sigma}_A^{-1} \mathbf{U}_A^\top \mathbf{L}_{A_r}^{-1} \qquad (8)$$

General case: separable relaxation. Let $s_{ji} = \sigma_{A,i}\sigma_{B,j}$. For $\lambda_{\text{ret}} > 0$ and $r < \min(m, n)$, the coefficients $s_{ji}^2 + \lambda_{\text{ret}}$ generally do not factor into a row weight and a column weight, so ordinary SVD truncation does not in general solve the resulting weighted approximation problem (Srebro and Jaakkola, 2003). We adopt the separable relaxation that keeps the Wiener-filtered target but replaces the weights with their separable surrogate $\sigma_{A,i}^2\sigma_{B,j}^2$. Concretely,

$$\widetilde{G}_{ji} = \frac{\sigma_{A,i}^2\sigma_{B,j}^2}{\sigma_{A,i}^2\sigma_{B,j}^2 + \lambda_{\text{ret}}}, \quad \mathbf{Y}^\star = -(\widetilde{\mathbf{G}} \odot \mathbf{R}), \quad (9)$$

followed by $\mathbf{Y}_r = \text{SVD}_r(\mathbf{Y}^\star)$, $\mathbf{H}_r = \mathbf{\Sigma}_B^{-1}\mathbf{Y}_r\mathbf{\Sigma}_A^{-1}$, and

$$\mathbf{\Delta}^{(r)} = \mathbf{L}_{B_r}^{-\top}\mathbf{U}_B\mathbf{H}_r\,\mathbf{U}_A^\top\mathbf{L}_{A_r}^{-1}. \quad (10)$$

(The sign is absorbed into $\mathbf{Y}^\star$.)

The relaxation is exact either at full rank for any $\lambda_{\text{ret}} \geq 0$, or at $\lambda_{\text{ret}} = 0$ for any r (Proposition 2).

Interpretation. Let $\mathbf{M} = \mathbf{Q}_B^{-1}\mathbf{W}\mathbf{Q}_A^{-\top}$. The quantity s_{ji}^2 is the forget-to-retain curvature ratio along coordinate (j, i) in the simultaneous basis. The full-rank solution in that basis is

$$H_{ji}^\star = -\frac{s_{ji}^2}{s_{ji}^2 + \lambda_{\text{ret}}} M_{ji}. \quad (11)$$

At unit edit strength, the corresponding post-edit coordinate is

$$M_{ji} + H_{ji}^\star = \frac{\lambda_{\text{ret}}}{s_{ji}^2 + \lambda_{\text{ret}}} M_{ji}. \quad (12)$$

For $\lambda_{\text{ret}} > 0$, coordinates with $s_{ji}^2 \gg \lambda_{\text{ret}}$ are strongly reduced, while those with $s_{ji}^2 \ll \lambda_{\text{ret}}$ remain close to their original values. When $s_{ji}^2 = \lambda_{\text{ret}}$, the full-rank edit removes half of the coordinate.

The reduced-rank construction truncates the Wiener-filtered matrix $\mathbf{Y}^\star$ from Eq. 9. It keeps the leading singular components of this transformed target before mapping the edit back to the original weight coordinates. The invertible changes of coordinates preserve rank, so the resulting edit has rank at most r.

Diagonal and forget-only variants. When all four Fisher factors are diagonal, a coordinate-aligned simultaneous basis can be chosen. The full-rank solution then applies a coordinatewise Wiener update directly to the original weights, as given in Eq. 15. Selective Synaptic Dampening (Foster et al., 2024) instead selects parameters using a forget-to-training importance threshold and applies a clipped training-to-forget importance multiplier to the selected parameters. We use the diagonal-factor variant as a control in Section 4. Setting $\mathbf{A}_r = \mathbf{I}$ and $\mathbf{B}_r = \mathbf{I}$ replaces the retain-side metric by an isotropic penalty and yields the forget-only variant.

Algorithm 1 K-FORGE

Require: Weight matrix $\mathbf{W}$, forget/retain sets $\mathcal{D}_f, \mathcal{D}_r$, rank r, factor damping ε, retain penalty λ_{ret}, strength α

$(\mathbf{A}_f, \mathbf{B}_f) \leftarrow \text{KronFisher}(\mathbf{W}, \mathcal{D}_f)$; $(\mathbf{A}_r, \mathbf{B}_r) \leftarrow \text{KronFisher}(\mathbf{W}, \mathcal{D}_r)$

Trace-scale damp factors by ε and compute $\mathbf{L}_{A_\bullet}, \mathbf{L}_{B_\bullet}$

$\mathbf{K}_A \leftarrow \mathbf{L}_{A_r}^{-1}\mathbf{L}_{A_f}$

$\mathbf{K}_B \leftarrow \mathbf{L}_{B_r}^{-1}\mathbf{L}_{B_f}$

$\text{SVD}(\mathbf{K}_A) = \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^\top$

$\text{SVD}(\mathbf{K}_B) = \mathbf{U}_B\mathbf{\Sigma}_B\mathbf{V}_B^\top$

$\mathbf{R} \leftarrow \mathbf{V}_B^\top\mathbf{L}_{B_f}^\top\mathbf{W}\mathbf{L}_{A_f}\mathbf{V}_A$

$\widetilde{G}_{ji} \leftarrow \sigma_{A,i}^2\sigma_{B,j}^2/(\sigma_{A,i}^2\sigma_{B,j}^2 + \lambda_{\text{ret}})$

$\mathbf{Y}^\star \leftarrow -(\widetilde{\mathbf{G}} \odot \mathbf{R})$

$\mathbf{Y}_r \leftarrow \text{SVD}_r(\mathbf{Y}^\star)$

$\mathbf{H}_r \leftarrow \mathbf{\Sigma}_B^{-1}\mathbf{Y}_r\mathbf{\Sigma}_A^{-1}$

$\mathbf{\Delta}^{(r)} \leftarrow \mathbf{L}_{B_r}^{-\top}\mathbf{U}_B\mathbf{H}_r\mathbf{U}_A^\top\mathbf{L}_{A_r}^{-1}$

return $\mathbf{W} + \alpha\,\mathbf{\Delta}^{(r)}$

3.3 The K-FORGE Algorithm

Damping changes the numerical conditioning and the quadratic metrics used by the edit. The retain penalty $\lambda_{\text{ret}} \geq 0$ controls the weight assigned to changes in the retain-side metric in Eq. 3. Before strength scaling, $\mathbf{\Delta}^{(r)}$ is the unique minimizer when the rank budget is unrestricted, and an exact rank-constrained minimizer when $\lambda_{\text{ret}} = 0$, by Theorem 1 and Proposition 1. Otherwise, the SVD step solves the separable surrogate; a decrease in the original objective is not guaranteed. The edit strength $\alpha \geq 0$ scales this direction before it is applied to the checkpoint. For $\alpha \neq 1$, the scaled edit is not in general a minimizer of Eq. 3. Computing the edit requires calibration forward and backward passes and matrix factorizations, but no iterative parameter-update loop.

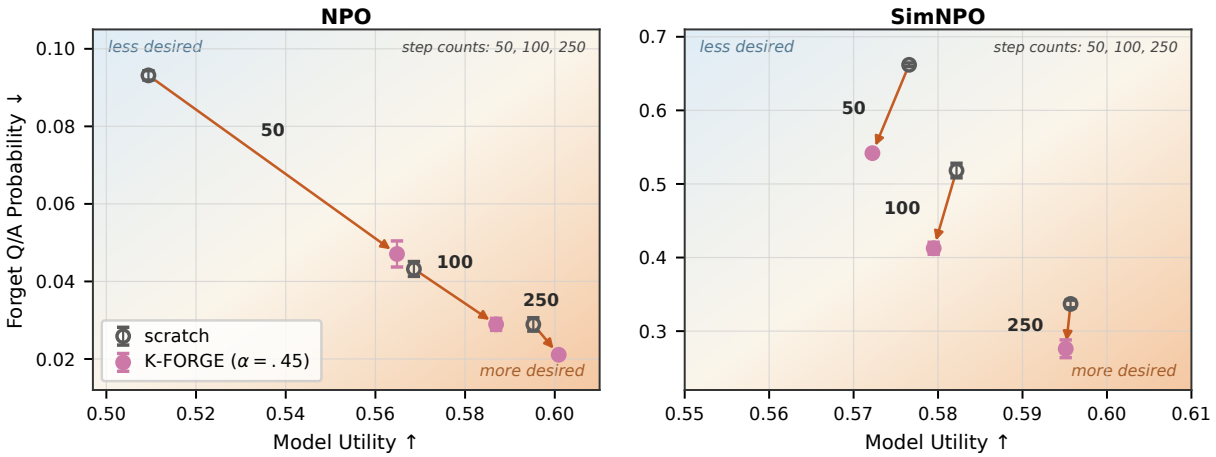


Figure 2: Matched downstream-step shifts on TOFU `forget10`. Arrows connect scratch runs to K-FORGE-initialized runs at the same optimizer-step budget; warmer background regions indicate higher utility and lower forget probability. Points show means; error bars indicate forget-probability standard deviations ($n = 3$).

3.4 K-FORGE as Initialization

After computing the low-rank edit, we replace the selected weight matrix by

$$\widehat{\mathbf{W}} = \mathbf{W} + \alpha \mathbf{\Delta}^{(r)}. \tag{13}$$

All other model parameters remain unchanged. We denote the resulting full-model parameters by $\widehat{\theta}$. Starting from $\widehat{\theta}$, we run downstream training with NPO (Zhang et al., 2024) or SimNPO (Fan et al., 2024). We compare these runs with runs started from the original checkpoint $\theta^\star$, using the same downstream loss formulation.

4 Experimental Setup

Our primary experiments use Llama 3.2 1B Instruct. We test model-family transfer on Gemma 3 1B-IT and report limited Llama 3.2 3B Instruct and Qwen2.5 1.5B Instruct checks. The MUSE experiments use Llama 2 7B. We use OpenUnlearning (Dorna et al., 2025) for training and evaluation throughout.

4.1 Benchmarks

We evaluate on TOFU (Maini et al., 2024), using the `forget10`, `forget05`, and `forget01` splits, and on the News and Books domains of MUSE (Shi et al., 2025).

Repeated runs. Unless stated otherwise, downstream comparisons report averages over three runs with different random seeds. The Gemma initialization and quantized-loading comparisons use four runs; its relearning comparisons use three. In tables and figure captions, n denotes the number of runs per experimental configuration. The comparison in Table 2 uses seeds $0, 1, 2$. In performance columns, bold marks the best displayed mean within each comparison block, including all ties at the reported precision. A displayed standard deviation of 0.000 may reflect rounding.

4.2 Metrics

Our primary endpoint is Forget Q/A Probability; lower values indicate less probability assigned to the gold forget answers. For each item, we compute the mean cross-entropy over non-ignored target tokens and report the exponential of its negative. When an item has multiple answers, values are averaged within the item before averaging over items.

The TOFU results also report Model Utility, Extraction Strength, and Forget Q/A ROUGE. Forget Q/A ROUGE uses ROUGE-L recall. Model Utility is the harmonic mean of the nine configured components: `retain_Q_A_Prob`, `retain_Q_A_ROUGE`, `retain_Truth_Ratio`, `ra_Q_A_Prob_normalised`, `ra_Q_A_ROUGE`, `ra_Truth_Ratio`, `wf_Q_A_Prob_normalised`, `wf_Q_A_ROUGE`, and `wf_Truth_Ratio`. Extraction Strength is the mean $1 - k/L$, where k is the first target-suffix position from which greedy predictions exactly match the target suffix and L is the target length.

For MUSE, KnowMem and VerbMem are ROUGE-L F1 scores on the forget knowledge-memory and verbal-memory sets, respectively, while Retain is the ROUGE-L F1 score on the retain set. All displayed $\pm$ values are sample standard deviations across seed-level summaries.

Reported p-values use two-sided paired t-tests on matched seed-level values. The relearning p-value is computed on the per-seed changes from pre-relearning to post-relearning values. These tests are reported as unadjusted descriptive statistics.

4.3 Baselines

Pretrained checkpoint (scratch). Our primary baseline runs NPO (Zhang et al., 2024) or SimNPO (Fan et al., 2024) from the pretrained checkpoint $\theta^\star$, with no preceding edit. We use `scratch` to denote this initialization throughout.

Random rank-2 edit. We apply a random rank-2 perturbation before downstream unlearning. For each matched linear module, we draw $\mathbf{U} \in \mathbb{R}^{n\times 2}$ and $\mathbf{V} \in \mathbb{R}^{2\times m}$ with independent standard-normal entries using a seeded random generator, and form $\mathbf{\Delta}_{\text{raw}} = \mathbf{UV}$. We rescale this perturbation so that

$$\|\mathbf{\Delta}_{\text{raw}}\|_F = \|\mathbf{W}_{\text{template}} - \mathbf{W}_{\text{base}}\|_F, \qquad (14)$$

where the template is the corresponding K-FORGE checkpoint and the base is the original checkpoint. The rescaled perturbation is then folded into the base model. The reported main controls use one matched mlp.down_proj module.

Weight-SVD edit. For the weight-SVD control, let $\mathbf{W}_{\text{base}} = \mathbf{U\Sigma V}^\top$. Let $\mathbf{U}_r$, $\mathbf{\Sigma}_r$, and $\mathbf{V}_r$ denote the leading r singular components. We construct $\mathbf{\Delta}_{\text{raw}} = -\mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^\top$ with $r = 2$, and rescale it to the same Frobenius norm as the corresponding K-FORGE edit. The resulting perturbation is folded into the same matched linear module before downstream unlearning.

Diagonal Fisher factors. The diagonal variant restricts $\mathbf{A}_f$, $\mathbf{B}_f$, $\mathbf{A}_r$, and $\mathbf{B}_r$ to diagonal matrices, as described in Section 3.2. Its quadratic weights depend only on the diagonal entries of these factors. Let $d_{\bullet,ji} = (\mathbf{A}_\bullet)_{ii}(\mathbf{B}_\bullet)_{jj}$. The full-rank minimizer of the resulting objective is

$$\Delta_{ji}^{\text{diag},\star} = -\frac{d_{f,ji}}{d_{f,ji} + \lambda_{\text{ret}} d_{r,ji}} W_{ji}. \qquad (15)$$

This expression makes the coordinatewise weighting explicit; off-diagonal entries of the Fisher factors do not enter the objective. We denote this control by `diagonal` in the tables. For the reported diagonal control, we use `factor_mode=diagonal`, retain Fisher factors, the same damping, $\lambda_{\text{ret}} = 0.01$, and the same rank-2 SVD truncation in $\mathbf{Y}$-space as in Eq. 9. Thus Eq. 15 gives the full-rank limit, while the reported control uses its rank-constrained implementation. Its relation to Selective Synaptic Dampening (Foster et al., 2024) is discussed in Section 3.2.

Forget-only factors. The forget-only variant sets $\mathbf{A}_r = \mathbf{I}$ and $\mathbf{B}_r = \mathbf{I}$. Substituting these identities into Eq. 3 gives

$$J_{\text{forget-only}}(\mathbf{\Delta}) = \|\mathbf{W} + \mathbf{\Delta}\|_{F_f}^2 + \lambda_{\text{ret}}\|\mathbf{\Delta}\|_F^2. \qquad (16)$$

The retain-side term becomes an isotropic penalty on the edit. The forget-side term still uses the factors estimated on $\mathcal{D}_f$.

4.4 K-FORGE configuration

Unless stated otherwise, K-FORGE uses rank $r = 2$. We fold the edit into the checkpoint before running NPO or SimNPO.

For the main Llama 3.2 1B `forget10` runs, we use $\varepsilon = 0.001$, $\lambda_{\text{ret}} = 0.01$, `factor_mode=kron`, retain Fisher factors, factor normalization, and the `wiener_v2` edit variant. The target is the single module `model.layers.0.mlp.down_proj`, with `max_target_modules=1`.

K-FORGE calibration uses at most 32 batches with per-device batch size 8 and gradient accumulation 4. The calibration contains 256 forget examples and 256 retain examples, hence 512 examples in total. The representative run contains 9,515 and 9,099 retained factor-token rows for the forget and retain splits, respectively. The 47.7k input-token value in the compute table is a total over both splits.

The downstream runs update the full model without LoRA adapters or a parameter-freezing regex. They use `adamw_torch`, per-device batch size 4, gradient accumulation 8, learning rate 10^{-5}, gradient checkpointing, and bf16 autocasting. NPO uses $(\beta, \alpha, \gamma) = (0.1, 1, 1)$ with an NLL retain term and the fixed original checkpoint as its reference model. SimNPO uses $(\delta, \beta, \alpha, \gamma) = (0, 4.5, 1, 0.125)$ with an NLL retain term and does not use a reference model.

4.5 One-Shot K-FORGE

Table 1 shows the one-shot frontier on TOFU `forget10`.

Increasing α from 0.100 to 0.800 lowers forget probability from 0.878 to 0.411, while utility falls from 0.599 to 0.513.

5 Results

K-FORGE lowers mean Forget Q/A Probability at all three NPO step budgets (Table 2). At 50 steps, $\alpha = 0.600$ reduces forget probability from 0.093 to 0.045 and raises utility from 0.509 to 0.574. The probability gap narrows by 250 steps. Extraction

Method	Util.↑	Forget Pr.↓	ROUGE↓	Extr.↓
Base model	**0.599**	0.881	0.820	0.706
K-FORGE $\alpha=0.100$	**0.599**	0.878	0.814	0.698
K-FORGE $\alpha=0.300$	0.595	0.837	0.721	0.592
K-FORGE $\alpha=0.350$	0.594	0.821	0.692	0.542
K-FORGE $\alpha=0.450$	0.582	0.774	0.613	0.390
K-FORGE $\alpha=0.600$	0.556	0.657	0.539	0.262
K-FORGE $\alpha=0.800$	0.513	**0.411**	**0.427**	**0.130**

Table 1: One-shot results on TOFU `forget10` using Llama 3.2 1B Instruct for strengths from $\alpha = 0.100$ to 0.800. Results at $\alpha \in \{1, 3, 10\}$ are provided in the figure data; these strengths collapse utility. Best displayed values are bolded, including ties.

Strength is higher with K-FORGE at every NPO budget.

For SimNPO, K-FORGE lowers mean forget probability and Extraction Strength at 50, 100, and 250 steps (Table 2). With $\alpha = 0.450$, mean utility is 0.572 versus 0.577 for scratch at 50 steps and 0.595 versus 0.596 at 250 steps. Section 5.6 reports the separate comparison that includes setup time and FLOPs.

5.1 Llama 3.2 3B

We repeat all three budgets on Llama 3.2 3B Instruct (Appendix Table 5). SimNPO improves in utility, forget probability, extraction, and ROUGE at 50, 100, and 250 steps. NPO improves utility and forget probability strongly at 50 steps, but its forgetting metrics are worse at 100 and 250 steps.

5.2 Model-family transfer

At 100 downstream steps on Gemma 3 1B IT, K-FORGE with $\alpha = 0.800$ lowers mean NPO forget probability from 0.066 to 0.053 and extraction from 0.055 to 0.033 (Appendix Table 8). Mean utility is 0.401 for both initializers at the reported precision, while Forget ROUGE increases from 0.298 to 0.365. For SimNPO, forget probability decreases from 0.272 to 0.269 and utility from 0.410 to 0.409. This comparison does not match total compute.

In the Qwen2.5 1.5B SimNPO pilot, scratch and K-FORGE have the same rounded mean forget probability (0.262) and utility (0.345) at both 50 and 100 steps (Appendix Table 9).

5.3 Transfer beyond TOFU

We additionally evaluate SimNPO on the News and Books domains of MUSE with Llama 2 7B (Appendix Table 7). At 50 steps on News, K-FORGE lowers KnowMem and VerbMem, raises extraction, and changes retain quality from 0.534 to 0.532. At 100 steps, $\alpha = 1.000$ lowers extraction from 0.303 to 0.301 and VerbMem from 0.573 to 0.551; retain quality falls from 0.532 to 0.520, and KnowMem is 0.629 for both initializers at the reported precision. On Books at 50 steps, KnowMem and VerbMem increase and retain quality decreases. At 100 steps, KnowMem decreases from 0.365 to 0.357, while extraction and VerbMem increase and retain quality decreases.

5.4 Ablations

Table 3 compares the initialization controls on Llama 3.2 1B Instruct at 50 and 100 downstream steps. K-FORGE has the lowest mean Forget Q/A Probability in each optimizer and budget block.

For NPO at 50 steps, the random and weight-SVD edits reduce mean forget probability from 0.093 for scratch to 0.082 and 0.084. The diagonal and forget-only variants reduce it further to 0.058 and 0.062, while K-FORGE reaches 0.047. At 100 steps, K-FORGE has mean forget probability 0.029, compared with 0.033 for the diagonal variant and 0.032 for forget-only. Weight-SVD and K-FORGE have the same rounded utility, 0.587, but weight-SVD has higher forget probability, 0.043.

For SimNPO at 100 steps, weight-SVD has lower forget probability than the diagonal variant, 0.488 versus 0.500. K-FORGE reduces it to 0.413, with utility 0.579 compared with 0.583–0.585 for the other initialization controls.

Appendix Table 6 reports the controls on the smaller TOFU splits. On `forget05` at 100 NPO steps, the random edit and K-FORGE have the same rounded forget probability, 0.025. Weight-SVD has higher utility than K-FORGE at that budget, 0.592 versus 0.586. On `forget01` at 100 NPO steps, K-FORGE has forget probability 0.011, compared with 0.012 for scratch, random, and weight-SVD.

5.5 Relearning and Quantized Loading

We evaluate how the models change under direct relearning and after quantized loading. For NPO relearning, Appendix Table 11 uses a separate Llama 3.2 1B series: scratch after 100 downstream steps and K-FORGE with $\alpha = 0.600$ after 50 steps. We evaluate the checkpoints before relearning and after 13 and 39 updates. The relearning comparison uses a different checkpoint series from the main matched-step and quantized-loading comparisons.

Steps	Method	Initialization	Utility↑	Forget Probability↓	Extraction↓
50	NPO	scratch	0.509±0.004	0.093±0.001	**0.068**±0.002
		K-FORGE (0.450)	0.565±0.003	0.047±0.003	0.072±0.003
		K-FORGE (0.600)	**0.574**±0.004	**0.045**±0.003	0.071±0.001
	SimNPO	scratch	**0.577**±0.006	0.662±0.003	0.270±0.010
		K-FORGE (0.450)	0.572±0.007	0.542±0.002	0.192±0.002
		K-FORGE (0.600)	0.571±0.005	**0.534**±0.002	**0.188**±0.005
100	NPO	scratch	0.569±0.009	0.043±0.002	**0.065**±0.003
		K-FORGE (0.450)	0.587±0.002	**0.029**±0.002	0.073±0.003
		K-FORGE (0.600)	**0.593**±0.005	0.031±0.002	0.073±0.003
	SimNPO	scratch	**0.582**±0.002	0.518±0.010	0.172±0.013
		K-FORGE (0.450)	0.579±0.003	0.413±0.008	**0.143**±0.002
		K-FORGE (0.600)	**0.582**±0.002	**0.407**±0.011	0.144±0.005
250	NPO	scratch	0.595±0.009	0.029±0.002	**0.064**±0.003
		K-FORGE (0.450)	0.601±0.003	**0.021**±0.001	0.072±0.003
		K-FORGE (0.600)	**0.603**±0.004	0.022±0.001	0.074±0.001
	SimNPO	scratch	**0.596**±0.003	0.337±0.005	0.116±0.004
		K-FORGE (0.450)	0.595±0.001	0.276±0.012	0.111±0.003
		K-FORGE (0.600)	0.592±0.004	**0.272**±0.008	**0.110**±0.001

Table 2: K-FORGE initialization (α value in parentheses) against scratch on TOFU `forget10`. Entries are mean ± sample standard deviation from standalone FP32 evaluation summaries ($n = 3$).

Method	Init	S50		S100	
		Utility↑	Forget Probability↓	Utility↑	Forget Probability↓
NPO	scratch	0.509±0.004	0.093±0.001	0.569±0.009	0.043±0.002
	random	0.516±0.004	0.082±0.005	0.579±0.008	0.038±0.005
	weight-SVD	0.538±0.012	0.084±0.006	**0.587**±0.003	0.043±0.001
	diagonal	0.561±0.007	0.058±0.007	0.584±0.011	0.033±0.001
	forget-only	0.543±0.004	0.062±0.004	0.582±0.003	0.032±0.001
	K-FORGE	**0.565**±0.003	**0.047**±0.003	**0.587**±0.002	**0.029**±0.002
SimNPO	scratch	0.577±0.006	0.662±0.003	0.582±0.002	0.518±0.010
	random	**0.578**±0.011	0.650±0.004	0.584±0.001	0.515±0.002
	weight-SVD	0.574±0.005	0.619±0.006	**0.585**±0.003	0.488±0.005
	diagonal	0.570±0.003	0.607±0.002	0.583±0.003	0.500±0.007
	forget-only	0.572±0.005	0.619±0.006	0.583±0.001	0.491±0.004
	K-FORGE	0.572±0.007	**0.542**±0.002	0.579±0.003	**0.413**±0.008

Table 3: Initialization controls on TOFU `forget10` with Llama 3.2 1B Instruct ($n = 3$). S50 and S100 denote 50 and 100 downstream steps. Best displayed means are bolded within each method and step-budget block, including ties.

After 13 relearning updates, mean forget probability increases by 0.494 for K-FORGE and by 0.329 for scratch. The reported $p < 0.001$ is from a two-sided paired t-test on the three per-seed changes in forget probability.

In the separate three-epoch runs, final forget probability is 0.911 for K-FORGE and 0.730 for scratch. After 39 updates, K-FORGE also has higher mean forget probability than scratch for Llama SimNPO and both Gemma optimizers (Appendix Table 13). Direct relearning increases forget probability for both initializers.

For Llama NPO, quantized loading is applied to the scratch S100 checkpoint and the K-FORGE S50 checkpoint. For Llama SimNPO, both initializers use S50 checkpoints. Quantized evaluations are paired with FP32 evaluations of the same source checkpoints.

The corresponding FP32-to-quantized changes in Forget Q/A Probability for scratch/K-FORGE are $+0.0018/+0.0016$ for NPO at 8-bit loading and $+0.0084/+0.0117$ for NPO at 4-bit loading. The SimNPO changes are $-0.0090/-0.0074$ at 8-bit and $-0.1397/-0.0959$ at 4-bit loading.

In the NPO quantization series, scratch/K-FORGE mean forget probability is 0.049/0.054 after 8-bit loading and 0.056/0.064 after 4-bit loading (Appendix Table 12). SimNPO has lower mean for-

Size	Edited W $n \times m$	Calibration examples / input tokens	Time (s) calib./edit/ end-to-end	FLOPs	Eq. steps NPO/SimNPO
1B	2048×8192	512/47.7k	18/62/≤81	$3.6 \cdot 10^{14}$	7.0/8.2
3B	3072×8192	512/47.7k	51/104/≤145	$9.3 \cdot 10^{14}$	6.9/8.1

Table 4: One-time K-FORGE setup cost for one edited layer. Analytical step equivalents include prompt and answer input tokens. Timing intervals are nested: edit time includes calibration, and end-to-end time additionally includes startup and serialization. The 512 calibration examples and 47.7k estimated input tokens are totals over the forget and retain splits; the corresponding factor-token rows are 9,515 and 9,099.

get probability with K-FORGE at both precisions. On Gemma, NPO forget probability is 0.070 for both initializers at the reported precision after 8-bit loading; after 4-bit loading it is 0.058 for scratch and 0.061 for K-FORGE. Gemma SimNPO has lower mean forget probability with K-FORGE at both precisions (Appendix Table 14).

5.6 K-FORGE compute overhead

K-FORGE adds one-time calibration, factorization, and edit cost. Table 4 reports the analytical setup estimate for one edited MLP down-projection. The edit time spans calibration through application; end-to-end also includes startup and serialization. At 1B, the complete 80.2-second setup is 40.7%/61.5% of evaluation-free NPO/SimNPO S50 training. FLOPs include prompt and answer tokens and exact factor rows; model passes contribute 97.7%/99.1% at 1B/3B. The remaining $\mathcal{O}(N(m^2+n^2)+m^3+n^3)$ factor cost limits wide or multi-layer edits.

For Llama 3.2 1B, setup FLOPs equal about 7.0 NPO or 8.2 SimNPO training steps; the dynamic-padding sensitivity calculation remains below 10 steps. The runs in Table 2 use equal downstream-step budgets.

In a separate evaluation-free FP32 resource comparison, K-FORGE runs for 50 downstream steps after setup, while scratch uses 73 NPO steps or 86 SimNPO steps. The K-FORGE totals include the full 80.2-second setup and its FLOPs. K-FORGE has lower mean total wall time than scratch for both NPO and SimNPO. Because the optimizer-step budgets differ, this comparison is resource-covering rather than step-matched.

For the compute comparison, percentage changes are computed from group means:

$$100\,\frac{\bar{p}_{\text{scratch}} - \bar{p}_{\text{K-FORGE}}}{\bar{p}_{\text{scratch}}}. \tag{17}$$

In this comparison, K-FORGE reduces NPO forget probability by 22.3% ($p = 0.007$, descriptive) and increases utility by 0.028 ($p < 0.001$), while extraction and ROUGE increase. For SimNPO, K-FORGE has lower forget probability in each paired comparison, with a mean reduction of 2.8% ($p = 0.126$, descriptive). Mean extraction and ROUGE decrease, and utility decreases by 0.011. With setup included, K-FORGE uses less total time and compute in both comparisons.

6 Conclusion

We introduced K-FORGE, a Fisher-guided initializer with an exact full-rank Wiener solution and a separable low-rank relaxation. It improves the early Forget Q/A Probability trajectory of NPO and SimNPO on TOFU; the NPO gain remains when setup FLOPs and time are included, while SimNPO has a smaller probability reduction and lower utility. On Gemma 3 1B IT, both optimizers have lower mean forget probability at 100 steps. Qwen's mean forget probability and utility are unchanged at the reported precision, and MUSE results vary by metric.

7 Limitations

K-FORGE requires downstream training and offers no certified-erasure guarantee. Direct relearning recovers forgotten behavior. The lower Forget Q/A Probability observed with NPO can coincide with higher Extraction Strength and Forget Q/A ROUGE. Transfer results vary across model families and MUSE metrics. For one 1B layer, complete setup costs 40.7% of an evaluation-free NPO S50 run and 61.5% of a SimNPO S50 run by wall-clock time, versus 14.0% and 16.4% by analytical FLOPs. Multi-layer and broader large-model scaling remain open.

References

Sungmin Cha, Sungjun Cho, Dasol Hwang, and Moontae Lee. 2025. Towards Robust and Parameter-Efficient Knowledge Unlearning for LLMs. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Viktoriia A. Chekalina, Daniil Moskovskiy, Tatyana Matveeva, Andrey Kuznetsov, and Evgeny Frolov. 2026. Scalable Kronecker-Factored Fisher Approximation for Neural Network Parameter Sensitivity. In *Proceedings of the 43rd International Conference on Machine Learning (ICML 2026)*, Proceedings of Machine Learning Research. PMLR. Accepted to ICML 2026.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zico Kolter, Zachary Lipton, and Pratyush Maini. 2025. OpenUnlearning: Accelerating LLM Unlearning via Unified Benchmarking of Methods and Metrics. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc.

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. 2024. Simplicity Prevails: Rethinking Negative Preference Optimization for LLM Unlearning. In *Safe Generative AI Workshop (Poster), 2024*. Poster; workshop paper.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. 2024. Fast Machine Unlearning without Retraining through Selective Synaptic Dampening. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pages 12043–12051. AAAI Press.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. 2018. Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9573–9583.

Roger B. Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamile Lukosiute, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. 2023. Studying Large Language Model Generalization with Influence Functions. *CoRR*, abs/2308.03296.

Jinghan Jia, Yihua Zhang, Yimeng Zhang, Jiancheng Liu, Bharat Runwal, James Diffenderfer, Bhavya Kailkhura, and Sijia Liu. 2024. SOUL: Unlocking the Power of Second-Order Optimization for LLM Unlearning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 4276–4292. Association for Computational Linguistics.

Yejin Kim, Eunwon Kim, Buru Chang, and Junsuk Choe. 2025. Improving Fisher Information Estimation and Efficiency for LoRA-based LLM Unlearning. In *Second Conference on Language Modeling*, Montreal, Canada.

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, and 27 others. 2024. The WMDP Benchmark: Measuring and Reducing Malicious Use with Unlearning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28525–28550. PMLR.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. 2024. Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary C. Lipton, and J. Zico Kolter. 2024. TOFU: A Task of Fictitious Unlearning for LLMs. In *Conference on Language Modeling*, Philadelphia, PA, USA.

James Martens and Roger B. Grosse. 2015. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2408–2417. JMLR.org.

Lev McKinney, Anvith Thudi, Juhan Bae, Tara Rezaei, Nicolas Papernot, Sheila A. McIlraith, and Roger B. Grosse. 2026. Gauss-Newton Unlearning for the LLM Era. *CoRR*, abs/2602.10568.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A. Smith, and Chiyuan Zhang. 2025. MUSE: Machine Unlearning Six-Way Evaluation for Language Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Nathan Srebro and Tommi S. Jaakkola. 2003. Weighted Low-Rank Approximations. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 720–727. AAAI Press.

Zeguan Xiao, Lang Mo, Yun Chen, Lei Yang, Jiehui Zhao, Lili Yang, and Guanhua Chen. 2026. Representation-Guided Parameter-Efficient LLM Unlearning. In *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pages 14602–14616. Association for Computational Linguistics.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. 2024. Negative Preference Optimization: From Catastrophic Collapse to Effective Unlearning. In *First Conference on Language Modeling*, Philadelphia, PA, USA.

Xingjian Zhao, Mohammad Mohammadi Amiri, and Malik Magdon-Ismail. 2026. WIN-U: Woodbury-Informed Newton-Unlearning as a retain-free Machine Unlearning Framework. *CoRR*, abs/2604.13438.

A Proofs

Throughout this appendix we use the notation of Section 3. Let $\mathbf{W} \in \mathbb{R}^{n\times m}$, $\mathbf{A}_\bullet \in \mathbb{R}^{m\times m}$, $\mathbf{B}_\bullet \in \mathbb{R}^{n\times n}$, with Cholesky factors $\mathbf{L}_{A_\bullet}$, $\mathbf{L}_{B_\bullet}$ and cross-Cholesky SVDs

$$\mathbf{K}_A = \mathbf{L}_{A_r}^{-1}\mathbf{L}_{A_f} = \mathbf{U}_A\mathbf{\Sigma}_A\mathbf{V}_A^\top,$$
$$\mathbf{K}_B = \mathbf{L}_{B_r}^{-1}\mathbf{L}_{B_f} = \mathbf{U}_B\mathbf{\Sigma}_B\mathbf{V}_B^\top.$$

The simultaneous diagonalizers are $\mathbf{Q}_A = \mathbf{L}_{A_r}^{-\top}\mathbf{U}_A$ and $\mathbf{Q}_B = \mathbf{L}_{B_r}^{-\top}\mathbf{U}_B$. Standard algebra gives $\mathbf{Q}_A^\top\mathbf{A}_f\mathbf{Q}_A = \mathbf{\Sigma}_A^2$, $\mathbf{Q}_A^\top\mathbf{A}_r\mathbf{Q}_A = \mathbf{I}$, and analogously for $\mathbf{Q}_B$. We parametrize the perturbation as $\mathbf{\Delta} = \mathbf{Q}_B\mathbf{H}\mathbf{Q}_A^\top$ with $\mathbf{H} \in \mathbb{R}^{n\times m}$.

A.1 Lemma: change of coordinates

Lemma 1. *Let* $\mathbf{M} := \mathbf{Q}_B^{-1}\mathbf{W}\mathbf{Q}_A^{-\top}$. *Then*

$$J(\mathbf{\Delta}) = \sum_{j,i}\Big[\sigma_{A,i}^2\sigma_{B,j}^2(M_{ji}+H_{ji})^2 + \lambda_{\text{ret}}H_{ji}^2\Big],$$

and $\mathbf{M} = \mathbf{\Sigma}_B^{-1}\mathbf{R}\mathbf{\Sigma}_A^{-1}$, *where* $\mathbf{R} = \mathbf{V}_B^\top\mathbf{L}_{B_f}^\top\mathbf{W}\mathbf{L}_{A_f}\mathbf{V}_A$.

Proof. By definition of $\mathbf{M}$, $\mathbf{W} = \mathbf{Q}_B\mathbf{M}\mathbf{Q}_A^\top$, so $\mathbf{W}+\mathbf{\Delta} = \mathbf{Q}_B(\mathbf{M}+\mathbf{H})\mathbf{Q}_A^\top$. Using $\operatorname{vec}(\mathbf{Q}_B\mathbf{X}\mathbf{Q}_A^\top) = (\mathbf{Q}_A\otimes\mathbf{Q}_B)\operatorname{vec}(\mathbf{X})$ and the diagonalizing identities,

$$\|\mathbf{W}+\mathbf{\Delta}\|_{F_f}^2 = \operatorname{vec}(\mathbf{M}+\mathbf{H})^\top\big(\mathbf{\Sigma}_A^2\otimes\mathbf{\Sigma}_B^2\big)\operatorname{vec}(\mathbf{M}+\mathbf{H}) = \sum_{j,i}\sigma_{A,i}^2\sigma_{B,j}^2(M_{ji}+H_{ji})^2. \tag{18}$$

and analogously $\|\mathbf{\Delta}\|_{F_r}^2 = \|\mathbf{H}\|_F^2$.

For the second claim, $\mathbf{K}_B = \mathbf{L}_{B_r}^{-1}\mathbf{L}_{B_f}$ gives $\mathbf{L}_{B_r}\mathbf{U}_B = \mathbf{L}_{B_f}\mathbf{V}_B\mathbf{\Sigma}_B^{-1}$, hence $\mathbf{L}_{B_r} = \mathbf{L}_{B_f}\mathbf{V}_B\mathbf{\Sigma}_B^{-1}\mathbf{U}_B^\top$ and $\mathbf{U}_B^\top\mathbf{L}_{B_r}^\top = \mathbf{\Sigma}_B^{-1}\mathbf{V}_B^\top\mathbf{L}_{B_f}^\top$. By analogous algebra, $\mathbf{L}_{A_r}\mathbf{U}_A = \mathbf{L}_{A_f}\mathbf{V}_A\mathbf{\Sigma}_A^{-1}$. Therefore

$$\begin{aligned}\mathbf{M} &= \mathbf{Q}_B^{-1}\mathbf{W}\mathbf{Q}_A^{-\top} = \mathbf{U}_B^\top\mathbf{L}_{B_r}^\top\mathbf{W}\mathbf{L}_{A_r}\mathbf{U}_A\\ &= \mathbf{\Sigma}_B^{-1}\mathbf{V}_B^\top\mathbf{L}_{B_f}^\top\mathbf{W}\mathbf{L}_{A_f}\mathbf{V}_A\mathbf{\Sigma}_A^{-1}\\ &= \mathbf{\Sigma}_B^{-1}\mathbf{R}\mathbf{\Sigma}_A^{-1}.\end{aligned}$$

□

A.2 Proof of Theorem 1

Proof. By Lemma 1, J is separable across entries (j,i), so its Hessian in $\mathbf{H}$ is diagonal with entries $2(\sigma_{A,i}^2\sigma_{B,j}^2+\lambda_{\text{ret}})$. Under the assumption $\sigma_{A,i},\sigma_{B,j}>0$ (Section 3.1), each entry is strictly positive, so J is strictly convex and admits a unique minimizer. Setting the partial derivative to zero,

$$2\sigma_{A,i}^2\sigma_{B,j}^2(M_{ji}+H_{ji}) + 2\lambda_{\text{ret}}H_{ji} = 0,$$

yields

$$H_{ji}^\star = -\frac{\sigma_{A,i}^2\sigma_{B,j}^2}{\sigma_{A,i}^2\sigma_{B,j}^2+\lambda_{\text{ret}}}M_{ji}.$$

Substituting $M_{ji} = R_{ji}/(\sigma_{A,i}\sigma_{B,j})$ from Lemma 1,

$$H_{ji}^\star = -\frac{\sigma_{A,i}\sigma_{B,j}}{\sigma_{A,i}^2\sigma_{B,j}^2+\lambda_{\text{ret}}}R_{ji} = -G_{ji}R_{ji},$$

so $\mathbf{H}^\star = -(\mathbf{G}\odot\mathbf{R})$. Mapping back,

$$\mathbf{\Delta}^\star = \mathbf{Q}_B\mathbf{H}^\star\mathbf{Q}_A^\top = -\mathbf{L}_{B_r}^{-\top}\mathbf{U}_B(\mathbf{G}\odot\mathbf{R})\mathbf{U}_A^\top\mathbf{L}_{A_r}^{-1}.$$

For $r = \min(m,n)$ the rank constraint is inactive, so $\mathbf{\Delta}^\star$ is also the rank-r minimizer. □

A.3 Proof of Proposition 1

Proof. Set $\lambda_{\text{ret}} = 0$. By Lemma 1,

$$J(\mathbf{\Delta})\big|_{\lambda=0} = \sum_{j,i} \sigma_{A,i}^2 \sigma_{B,j}^2 (M_{ji} + H_{ji})^2 = \|\mathbf{\Sigma}_B(\mathbf{M} + \mathbf{H})\mathbf{\Sigma}_A\|_F^2.$$

Using $\mathbf{\Sigma}_B \mathbf{M} \mathbf{\Sigma}_A = \mathbf{R}$ and defining $\mathbf{Y} := \mathbf{\Sigma}_B \mathbf{H} \mathbf{\Sigma}_A$,

$$J|_{\lambda_{\text{ret}}=0}(\mathbf{H}) = \|\mathbf{Y} + \mathbf{R}\|_F^2.$$

Since $\mathbf{\Sigma}_A, \mathbf{\Sigma}_B$ are invertible diagonal and $\mathbf{Q}_A, \mathbf{Q}_B$ are invertible, $\operatorname{rank}(\mathbf{\Delta}) \le r \Leftrightarrow \operatorname{rank}(\mathbf{H}) \le r \Leftrightarrow \operatorname{rank}(\mathbf{Y}) \le r$. The constrained problem becomes

$$\min_{\operatorname{rank}(\mathbf{Y}) \le r} \|\mathbf{Y} - (-\mathbf{R})\|_F^2,$$

whose Eckart–Young (Frobenius) minimizer may be chosen as $\mathbf{Y}_r = -\operatorname{SVD}_r(\mathbf{R})$; equivalently, any rank-r SVD truncation of $-\mathbf{R}$ attains the optimum. For $1 \le r < \operatorname{rank}(\mathbf{R})$, the minimizer is unique when $\sigma_r(\mathbf{R}) > \sigma_{r+1}(\mathbf{R})$. If $r \ge \operatorname{rank}(\mathbf{R})$, the unique minimizer is $\mathbf{Y} = -\mathbf{R}$; for $r = 0$, it is $\mathbf{Y} = \mathbf{0}$. Unscaling this choice gives $\mathbf{H}_r = -\mathbf{\Sigma}_B^{-1} \operatorname{SVD}_r(\mathbf{R}) \mathbf{\Sigma}_A^{-1}$, and

$$\mathbf{\Delta}_{\lambda=0}^{(r)} = \mathbf{Q}_B \mathbf{H}_r \mathbf{Q}_A^\top = -\mathbf{L}_{B_r}^{-\top} \mathbf{U}_B \mathbf{\Sigma}_B^{-1} \operatorname{SVD}_r(\mathbf{R}) \mathbf{\Sigma}_A^{-1} \mathbf{U}_A^\top \mathbf{L}_{A_r}^{-1}. \qquad \square$$

A.4 Proposition 2: Limit consistency of the relaxation

Proposition 2. *The relaxed edit $\mathbf{\Delta}^{(r)}$ in Eq. 10 is exact in two limits: (i) at $r = \min(m, n)$ it equals the unique minimizer $\mathbf{\Delta}^\star$ from Theorem 1 for any $\lambda_{\text{ret}} \ge 0$; (ii) at $\lambda_{\text{ret}} = 0$ it equals an exact rank-r minimizer from Proposition 1 for any r.*

Proof. Case (i): $r = \min(m, n)$. The truncation $\operatorname{SVD}_r(\mathbf{Y}^\star) = \mathbf{Y}^\star$ is vacuous, so $\mathbf{H}_r = \mathbf{\Sigma}_B^{-1} \mathbf{Y}^\star \mathbf{\Sigma}_A^{-1} = -\mathbf{\Sigma}_B^{-1} (\widetilde{\mathbf{G}} \odot \mathbf{R}) \mathbf{\Sigma}_A^{-1}$. Entrywise,

$$(H_r)_{ji} = -\frac{\widetilde{G}_{ji}}{\sigma_{A,i}\sigma_{B,j}} R_{ji} = -\frac{\sigma_{A,i}\sigma_{B,j}}{\sigma_{A,i}^2 \sigma_{B,j}^2 + \lambda_{\text{ret}}} R_{ji} = -G_{ji} R_{ji},$$

which is $\mathbf{H}^\star$ from the proof of Theorem 1. Hence $\mathbf{\Delta}^{(r)} = \mathbf{Q}_B \mathbf{H}^\star \mathbf{Q}_A^\top = \mathbf{\Delta}^\star$.

Case (ii): $\lambda_{\text{ret}} = 0$. Then

$$\widetilde{G}_{ji} = \frac{\sigma_{A,i}^2 \sigma_{B,j}^2}{\sigma_{A,i}^2 \sigma_{B,j}^2} = 1.$$

Thus $\mathbf{Y}^\star = -\mathbf{R}$, and any $\mathbf{Y}_r = \operatorname{SVD}_r(-\mathbf{R})$ is an Eckart–Young minimizer of the transformed problem. Unscaling $\mathbf{H}_r = \mathbf{\Sigma}_B^{-1} \mathbf{Y}_r \mathbf{\Sigma}_A^{-1}$ and mapping back $\mathbf{\Delta}^{(r)} = \mathbf{Q}_B \mathbf{H}_r \mathbf{Q}_A^\top$ therefore yields an exact rank-r minimizer characterized by Proposition 1. $\square$

B Additional Experimental Results

Steps	Method	Initialization	Utility↑	Forget Probability↓	Extraction↓	ROUGE↓
50	NPO	scratch	0.589±0.021	0.086±0.005	**0.057**±0.006	**0.237**±0.010
		K-FORGE	**0.639**±0.015	**0.065**±0.004	0.070±0.001	0.293±0.005
	SimNPO	scratch	0.636±0.005	0.691±0.008	0.351±0.007	0.587±0.006
		K-FORGE	**0.656**±0.009	**0.583**±0.004	**0.264**±0.006	**0.519**±0.008
100	NPO	scratch	0.648±0.024	**0.034**±0.003	**0.055**±0.004	**0.260**±0.012
		K-FORGE	**0.670**±0.006	0.039±0.002	0.077±0.001	0.306±0.010
	SimNPO	scratch	0.647±0.009	0.517±0.006	0.223±0.005	0.486±0.004
		K-FORGE	**0.655**±0.003	**0.455**±0.006	**0.191**±0.005	**0.467**±0.004
250	NPO	scratch	**0.665**±0.008	**0.027**±0.003	**0.057**±0.003	**0.261**±0.007
		K-FORGE	0.662±0.008	0.029±0.001	0.079±0.002	0.302±0.008
	SimNPO	scratch	0.662±0.005	0.334±0.002	0.151±0.001	0.434±0.009
		K-FORGE	**0.666**±0.007	**0.314**±0.005	**0.142**±0.002	**0.429**±0.008

Table 5: Llama 3.2 3B Instruct results on TOFU `forget10` ($n = 3$). K-FORGE uses $\alpha = 0.450$; best displayed means are bolded within each method and budget, including ties.

Split	Method	Init	S50		S100	
			Utility↑	Forget Probability↓	Utility↑	Forget Probability↓
forget01	NPO	scratch	**0.578**±0.003	0.014±0.001	0.580±0.000	0.012±0.001
		random	0.574±0.003	0.013±0.002	0.579±0.001	0.012±0.002
		weight-SVD	0.577±0.002	0.013±0.002	**0.582**±0.001	0.012±0.001
		K-FORGE	0.573±0.001	**0.010**±0.001	0.579±0.004	**0.011**±0.001
	SimNPO	scratch	0.573±0.008	0.288±0.005	**0.580**±0.008	0.242±0.008
		random	0.573±0.007	0.286±0.006	**0.580**±0.006	0.241±0.010
		weight-SVD	**0.575**±0.006	0.304±0.007	0.578±0.007	0.256±0.014
		K-FORGE	0.570±0.008	**0.238**±0.008	0.577±0.007	**0.209**±0.005
forget05	NPO	scratch	0.542±0.007	0.050±0.005	0.578±0.009	0.028±0.001
		random	0.557±0.005	0.043±0.006	0.580±0.002	**0.025**±0.002
		weight-SVD	0.573±0.007	0.043±0.003	**0.592**±0.002	0.034±0.002
		K-FORGE	**0.575**±0.003	**0.032**±0.002	0.586±0.005	**0.025**±0.001
	SimNPO	scratch	0.573±0.010	0.549±0.029	0.586±0.001	0.417±0.016
		weight-SVD	**0.575**±0.007	0.476±0.010	**0.588**±0.003	0.375±0.005
		K-FORGE	0.571±0.004	**0.425**±0.012	0.586±0.005	**0.331**±0.010

Table 6: Transfer initialization controls on TOFU splits `forget05` and `forget01` ($n = 3$). The `forget05`/SimNPO block reports scratch, weight-SVD, and K-FORGE; a random-control result is not included. Best displayed means are bolded within each split, method, and step-budget block, including ties.

Domain	Steps	Init.	Extraction↓	KnowMem↓	VerbMem↓	Retain↑
News	50	scratch	**0.307**±0.002	0.630±0.006	0.574±0.002	**0.534**±0.007
	50	K-FORGE	0.315±0.004	**0.624**±0.010	**0.564**±0.012	0.532±0.006
	100	scratch	0.303±0.004	0.629±0.002	0.573±0.003	**0.532**±0.008
	100	K-FORGE (0.450)	0.310±0.001	**0.628**±0.006	0.562±0.012	0.522±0.006
	100	K-FORGE (1.000)	**0.301**±0.000	0.629±0.007	**0.551**±0.007	0.520±0.011
Books	50	scratch	**0.911**±0.000	**0.411**±0.013	**0.994**±0.005	**0.649**±0.001
	50	K-FORGE	**0.911**±0.000	0.426±0.005	0.996±0.002	0.645±0.006
	100	scratch	**0.852**±0.025	0.365±0.010	**0.955**±0.020	**0.610**±0.006
	100	K-FORGE	0.869±0.020	**0.357**±0.002	0.963±0.013	0.586±0.017

Table 7: MUSE transfer with SimNPO and Llama 2 7B ($n = 3$). K-FORGE uses $\alpha = 0.450$ unless shown otherwise. Best displayed means are bolded within each domain and step-budget block, including ties.

Method	Initialization	Utility↑	Forget Probability↓	Extraction↓	ROUGE↓
NPO	scratch	**0.401**±0.002	0.066±0.002	0.055±0.004	**0.298**±0.013
	K-FORGE	**0.401**±0.013	**0.053**±0.003	**0.033**±0.002	0.365±0.009
SimNPO	scratch	**0.410**±0.002	0.272±0.000	0.127±0.003	0.399±0.004
	K-FORGE	0.409±0.002	**0.269**±0.000	**0.122**±0.001	**0.395**±0.005

Table 8: Gemma 3 1B comparison at 100 downstream steps with $\alpha = 0.800$ ($n = 4$). Best displayed means are bolded within each optimizer, including ties.

Steps	Init.	n	Utility↑	Forget Probability↓	Extraction↓	ROUGE↓
50	scratch	3	**0.345**±0.002	**0.262**±0.000	**0.121**±0.001	0.359±0.004
50	K-FORGE	3	**0.345**±0.002	**0.262**±0.000	**0.121**±0.001	**0.356**±0.003
100	scratch	3	**0.345**±0.000	**0.262**±0.000	**0.120**±0.001	0.361±0.004
100	K-FORGE	3	**0.345**±0.000	**0.262**±0.000	0.121±0.001	**0.360**±0.004

Table 9: Qwen2.5 1.5B SimNPO pilot on TOFU `forget10`. K-FORGE edits layer 15 with rank 2 and $\alpha = 0.450$. Best displayed means are bolded within each budget, including ties.

Method	Initialization	Steps	FLOPs (10^{15}) Train	FLOPs (10^{15}) Setup	Total wall (s)	Forget Probability↓	Utility↑	Extraction↓	ROUGE↓
NPO	scratch	73	3.78	0.00	285.5	0.059	0.549	**0.067**	**0.263**
NPO	K-FORGE	50	2.59	0.36	**277.2**	**0.046**	**0.577**	0.072	0.272
SimNPO	scratch	86	3.80	0.00	221.6	0.553	**0.583**	0.196	0.487
SimNPO	K-FORGE	50	2.21	0.36	**210.5**	**0.537**	0.572	**0.188**	**0.454**

Table 10: Independent FP32 comparison on TOFU `forget10` using paired runs ($n = 3$), with evaluation excluded from timing. K-FORGE totals include the full 80.2-second setup and its FLOPs. K-FORGE uses less total wall time and fewer FLOPs than scratch in both optimizer comparisons.

Updates	Initialization	Utility↑	Forget Probability↓	Extraction↓	ROUGE↓
0	scratch	0.568	**0.047**	**0.067**	**0.268**
	K-FORGE	**0.575**	0.052	0.073	0.272
13	scratch	0.455	**0.376**	**0.112**	**0.376**
	K-FORGE	**0.493**	0.546	0.200	0.474
39	scratch	0.493	**0.730**	**0.407**	**0.622**
	K-FORGE	**0.510**	0.911	0.812	0.884

Table 11: NPO relearning results ($n = 3$) from a separate experiment series. Scratch uses an S100 checkpoint; K-FORGE uses S50 with $\alpha = 0.600$. Updates count relearning steps, with 0 denoting the pre-relearning checkpoint. Values at 0 and 13 updates are from one-epoch runs; values at 39 updates are from separate three-epoch runs. Best displayed means are bolded within each update block, including ties.

Pair	Load	Forget Probability↓	Utility↑	Extraction↓	ROUGE↓
NPO	8-bit	**0.049**/0.054	0.564/**0.569**	**0.066**/0.071	**0.270**/0.275
NPO	4-bit	**0.056**/0.064	**0.531**/0.526	**0.062**/0.065	**0.265**/0.268
SimNPO S50	8-bit	0.694/**0.561**	**0.575**/0.573	0.284/**0.192**	0.547/**0.465**
SimNPO S50	4-bit	0.563/**0.472**	**0.532**/0.528	0.178/**0.137**	0.468/**0.427**

Table 12: Post-quantization results from paired scratch/K-FORGE runs ($n = 3$); each cell lists scratch/K-FORGE. For NPO, the source checkpoints are S100 for scratch and S50 for K-FORGE; for SimNPO, both source checkpoints are S50. Eight-bit loading uses `load_in_8bit`; four-bit loading uses NF4 quantization with bfloat16 compute and double quantization disabled. The quantized checkpoints come from a series distinct from the relearning comparison.

Model	Method	Updates	Initialization	Forget Probability↓	Utility↑	Extraction↓	ROUGE↓
Llama 1B	SimNPO	0	scratch	**0.545**	**0.585**	**0.184**	0.480
			K-FORGE	0.569	0.575	0.202	**0.466**
		13	scratch	**0.803**	0.560	**0.539**	**0.708**
			K-FORGE	0.874	**0.564**	0.694	0.815
		39	scratch	**0.978**	**0.565**	**0.968**	**0.984**
			K-FORGE	0.982	0.553	0.977	0.987
Gemma 1B	NPO	0	scratch	**0.074**	0.350	0.031	**0.318**
			K-FORGE	**0.074**	**0.357**	**0.030**	0.383
		13	scratch	**0.261**	0.352	**0.119**	**0.414**
			K-FORGE	0.295	**0.374**	0.135	0.415
		39	scratch	**0.501**	**0.395**	**0.197**	**0.490**
			K-FORGE	0.552	0.394	0.225	0.520
Gemma 1B	SimNPO	0	scratch	0.273	**0.404**	0.127	0.406
			K-FORGE	**0.270**	0.403	**0.122**	**0.402**
		13	scratch	**0.353**	**0.385**	**0.145**	0.436
			K-FORGE	**0.353**	**0.385**	**0.145**	**0.433**
		39	scratch	**0.658**	**0.396**	**0.283**	**0.597**
			K-FORGE	0.665	0.393	0.296	0.598

Table 13: Additional relearning results ($n = 3$). For Llama, scratch uses an S100 checkpoint and K-FORGE uses an S50 checkpoint with $\alpha = 0.600$. For Gemma, both initializers use S50 checkpoints and K-FORGE uses $\alpha = 0.800$. Update 0 is the pre-relearning operating point. The Gemma relearning series is separate from the four-seed S100 initialization comparison. Best displayed means are bolded within each model, method, and update block, including ties.

Method	Load	Initialization	Forget Probability↓	Utility↑	Extraction↓	ROUGE↓
NPO	8-bit	scratch	**0.070**	0.388	0.053	**0.302**
	8-bit	K-FORGE	**0.070**	**0.390**	**0.034**	0.368
	4-bit	scratch	**0.058**	**0.327**	0.035	**0.330**
	4-bit	K-FORGE	0.061	0.295	**0.030**	0.399
SimNPO	8-bit	scratch	0.269	**0.403**	0.123	**0.402**
	8-bit	K-FORGE	**0.266**	**0.403**	**0.121**	**0.402**
	4-bit	scratch	0.219	0.388	0.102	0.419
	4-bit	K-FORGE	**0.214**	**0.390**	**0.101**	**0.417**

Table 14: Gemma 3 1B results after 8-bit and 4-bit loading ($n = 4$). Best displayed means are bolded within each method and precision, including ties.