WHEN FORGOTTEN FACTS RETURN: RELEARNING-AWARE UNLEARNING OF KNOWLEDGE GRAPH EMBEDDINGS VIA TANGENT KERNELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate unlearning answers a deletion request on a knowledge-graph embedding by editing entity embeddings instead of retraining from scratch. These embeddings keep being updated as new facts arrive. Under this ordinary continued training, forgotten facts recover more than in the exact-unlearning baseline, a model retrained without them and given the same continued training. We trace this relearning to entity embeddings shared by forgotten and new facts. To our knowledge, this is the first study of relearning in knowledge-graph embeddings. Relearning-robust methods developed for deep networks do not account for the graph structure underlying this recovery. We propose Kernel-Anticipated Unlearning (KAU), which uses the tangent kernel to model how continued training changes forgotten facts' scores through these shared parameters. KAU expresses this first-order response in closed form and differentiates through it inside the forgetting loss. The objective steers edits toward directions that continued training reverses less. We compare KAU with state-of-the-art knowledge-graph unlearners and relearning-robust methods under identical continued training, measuring leakage beyond the exact-unlearning baseline. Across three backbones, KAU reduces the strongest knowledge-graph unlearner's leakage by 46–63% on FB15k-237 and 50–82% on CoDEx-L, at higher retained utility.

1 INTRODUCTION

Once a fact has been deleted from a knowledge-graph embedding, does it stay deleted? Such embeddings rank facts, many of them about real people, for search, recommendation and question answering. When a person requests erasure or a curator retracts a false fact, the released model must stop ranking the deleted facts highly while still ranking the rest as a model retrained without them would. Retraining at every request is impractical, so knowledge-graph unlearners edit the released model instead (Liu et al., 2026; Xiao et al., 2025a;b; Xu et al., 2025). Evaluation usually stops at release, although the embeddings keep changing as the graph grows.

As new facts arrive, users keep training their embeddings on old and new true edges (Liu et al., 2024). We call this continued training *the update*. The update is ordinary use: it contains no forgotten fact and requires no adversary, yet it can undo unlearning. This failure, known as *benign relearning*, has been observed in language and vision models (Hu et al., 2025; Łucki et al., 2025; Yoon et al., 2026; Yang et al., 2026; Siddiqui et al., 2025). In a knowledge graph, the same entity embeddings must support new learning and preserve the deletion.

Measuring this failure in a knowledge graph requires a reference that itself moves. A language model retrained without facts about fictitious authors regains none under benign fine-tuning (Yoon et al., 2026). Facts in a graph, however, predict one another (Wei et al., 2025), so a model retrained without the forgotten facts partly recovers them by re-inferring them from the remaining graph. Only recovery beyond that can count as a failure to forget, so, following Siddiqui et al. (2025), we give this *retrained oracle* the identical update and measure the released model's gain in forget-set mean reciprocal rank (MRR) beyond the oracle's own (Dehghani et al., 2026). We call this *leakage*.

Measured this way, two recent knowledge-graph unlearners, GraphDPO (Liu et al., 2026) and ZeroFisher (Xiao et al., 2025b), leak on every tested graph and backbone. At release, GraphDPO on

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.

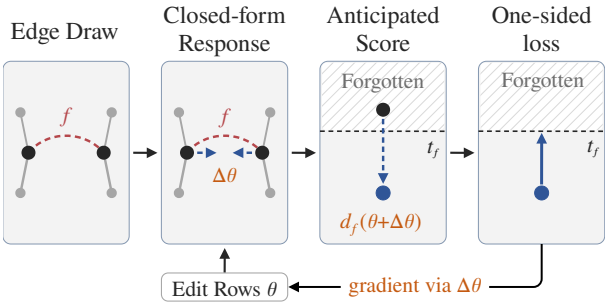


Figure 1: KAU's objective (Section 4). The loss acts on each forgotten fact f's score after a closed-form response $\Delta\theta$ to a drawn edge set, pushing it to a target t_f. Its gradient flows through $\Delta\theta$ to the edited rows θ.

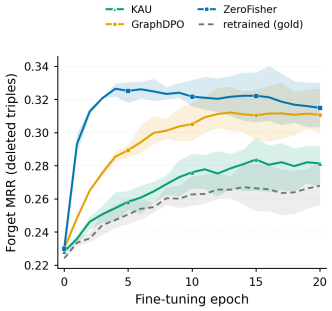


Figure 2: Forget-set MRR along one update on FB15k-237 with TransE. GraphDPO and ZeroFisher relearn far beyond the oracle; KAU stays closest.

FB15k-237 with TransE ranks the forgotten facts nearly as low as the retrained oracle. Yet under the update it regains, beyond the oracle, about as much as it removed (Figure 2).

What carries the return? Not the relation rows, which every edge of a relation shares, but the entity rows that the forgotten facts share with the update's edges (Appendix D.4). To forget a fact, an unlearner moves its endpoints' rows, which also raises the loss of true edges at those entities. The update repairs those edges by moving the rows back, and the forgotten fact returns with them. The retrained oracle re-infers through the same rows but has no edit to undo. If this repair undoes forgetting, each fact must be forgotten where the update will leave its rows, not only where they sit at release. Existing relearning-robust methods simulate the update, penalize a proxy, or confine the edit to a subspace (Tamirisa et al., 2025; Gao et al., 2025; Li et al., 2025; Fan et al., 2025b; Siddiqui et al., 2025; Golatkar et al., 2020b; Chen et al., 2024; Xiao et al., 2026), but none writes in closed form how the update's edges move a forgotten fact.

A knowledge-graph embedding makes this tractable. A fact's score gradient touches only its three rows, so the tangent kernel (Jacot et al., 2018), which couples two facts' scores to first order, is explicit and sparse. The update's linearized response has a closed form, known from continual-learning theory (Doan et al., 2021; Bennani et al., 2020) and used by Golatkar et al. (2020b) for a forgetting edit. We propose Kernel-Anticipated Unlearning (KAU), which minimizes an *anticipated loss*, the forgetting loss evaluated after this response (Figure 1). A score-preservation term keeps the retained facts near their pre-edit scores, leaving the update little to repair. In a linearized model where the update repairs every edge the edit changed, the update multiplies the edit by a survival filter (Section 4.6). The filter keeps little of each direction that strongly moves the scores of the update's edges, more of weaker ones, and all others. KAU's anticipated loss is then the plain forgetting loss on the surviving edit, so its gradient favors directions the update restores least. On both FB15k-237 (Toutanova & Chen, 2015) and CoDEx-L (Safavi & Koutra, 2020), with each of TransE, TransH and RotatE (Bordes et al., 2013; Wang et al., 2014; Sun et al., 2019), KAU leaks less than either unlearner and ranks the retained facts higher (Table 1). Our main contributions are:

1. **Forgotten facts return under the update.** We provide, to our knowledge, the first evidence that deletions by knowledge-graph unlearners do not last under ordinary continued training. We find that the forgotten facts return beyond what a retrained model recovers, through the representations of the entities they share with the training edges.
2. **A kernel-based framework for persistent unlearning.** We introduce KAU, which anticipates how continued training restores forgotten facts through shared entity embeddings. By differentiating through a closed-form tangent-kernel response inside the forgetting objective, KAU optimizes edits for persistence after the update. Our linearized theory shows that the update acts on the edit as a classical spectral filter weighting directions by resistance to reversal, with retained-gradient projection as a limiting case.
3. **Relearning defences on knowledge-graph embeddings.** We adapt relearning defences built for deep networks to knowledge-graph embeddings and find that none closes the gap to the retrained oracle, while those added to an existing unlearner leave most of its leakage (Sections 5.4 and 5.5).

2 RELATED WORK

Knowledge-graph unlearning. GraphDPO (Liu et al., 2026) updates the entity and relation tables by preferring reconstructed alternatives over forgotten facts and replaying adjacent retained edges. ZeroFisher (Xiao et al., 2025b) applies a Fisher-based influence update to the rows of the forgotten facts' local neighbours. Schema (Xiao et al., 2025a) re-embeds the deleted entities' neighbourhood, and MetaEU (Xu et al., 2025) regenerates the forgotten entities' rows from their relations and neighbours. Like graph neural network unlearners (Cheng et al., 2023; Wu et al., 2023; Cong & Mahdavi, 2023), all four use graph structure as edit scope or a term preserving neighbours. None computes how training on those neighbours changes a forgotten fact's score or has been evaluated for recovery under continued training. KAU edits only the entity rows at the forgotten facts' endpoints, and its loss also acts on each fact's score after a closed-form response to the update.

Relearning-robust methods. These methods were built for deep networks, where linking a training example to the parameters it moves is neither explicit nor cheap. Unrolled methods evaluate the forgetting loss after a simulated update (Gao et al., 2025; Tamirisa et al., 2025; Sondej et al., 2025). Proxy methods replace the update by a cheaper quantity: sharpness (Fan et al., 2025b), weight distance (Siddiqui et al., 2025) or the weight spectrum (Rosati et al., 2026). Subspace methods place the edit off the retained gradients' span to protect utility (Golatkar et al., 2020b; Chen et al., 2024; Zhou et al., 2025), in slowly restored directions (Xiao et al., 2026; Sondej & Yang, 2025; Sondej et al., 2026), or on weights a retain-only update leaves inert (Chen et al., 2026). In a knowledge graph no row is inert, because forgotten facts share rows with retained neighbours. Subspace methods also estimate their subspaces from retained or forget data, so they miss the update's never-trained edges. None of these methods writes in closed form how the update's edges move each forgotten fact. KAU does so to first order, with draws that include stand-ins for never-trained edges (Section 4.1). In the linearized model of Section 4.6, retained-span projection and slow-direction methods take the form of hard thresholds of the survival filter, which KAU's gradient applies softly.

Tangent-kernel response. The tangent kernel (Jacot et al., 2018) describes pretrained models' fine-tuning (Malladi et al., 2023), expresses interference between consecutive tasks in closed form (Bennani et al., 2020; Doan et al., 2021; Hidekel & Raviv, 2026) and underlies a scrubbing update (Golatkar et al., 2020b). Unlike these works, KAU places the response to the post-edit update inside its objective and differentiates through it. Appendix E extends this comparison (Table 19).

3 PROBLEM FORMULATION

Knowledge-graph embeddings. A knowledge graph is a set of triples $x = (h, r, t)$ over entities $\mathcal{V}$ and relations $\mathcal{R}$. An embedding model stores an entity row $\boldsymbol{e}_v \in \mathbb{R}^m$ for every $v \in \mathcal{V}$ and a relation row $\boldsymbol{w}_r$ for every $r \in \mathcal{R}$, and scores a triple from its three rows by a function $d_x(\theta)$ that is low when the triple is believed. Link prediction ranks candidate triples by that score. TransE scores $d_x(\theta) = \|\bar{\boldsymbol{e}}_h + \bar{\boldsymbol{w}}_r - \bar{\boldsymbol{e}}_t\|_1$ with $\bar{\boldsymbol{e}} = \boldsymbol{e}/\|\boldsymbol{e}\|_2$ (Bordes et al., 2013). RotatE (Sun et al., 2019) and TransH (Wang et al., 2014) are given in Appendix B.8. The score function and its row parametrization define a backbone. Each backbone is trained with a margin-ranking loss over random corruptions of each triple (Table 3), with margin γ, and we write $\mathrm{relu}(a) = [a]_+ = \max\{a, 0\}$.

Unlearning. The unlearner receives an original model θ° trained on $\mathcal{D}_{\mathrm{train}} \setminus B$, a forget set $\mathcal{F} \subset \mathcal{D}_{\mathrm{train}} \setminus B$ and the retained set $\mathcal{D}_R = \mathcal{D}_{\mathrm{train}} \setminus (B \cup \mathcal{F})$, and returns a model θ_u that no longer believes the forgotten facts but still believes the remaining graph. The never-trained set B contains true edges unseen in training but drawn by the update. The ideal release is the retrained oracle θ_g, trained from scratch on $\mathcal{D}_R$ using θ°'s recipe (subscript gold in Appendix C). The forgotten facts' endpoints are the involved entities $\mathcal{V}_\mathcal{F}$, and their rows the involved rows. Every relation row is shared by all triples of its relation, and an entity row outside $\mathcal{V}_\mathcal{F}$ enters the score of no forgotten fact, so the edit $\theta_u - \theta^\circ$ is confined to the involved rows, and Θ_{inv} is the set of parameters that agree with θ° elsewhere.

The update. After release, the user continues training on true edges. The update uses the training loss under Adam (Kingma & Ba, 2015) on a set N of retained edges and never-trained edges from B, with $N \cap \mathcal{F} = \emptyset$. It yields the model $\mathrm{upd}_N(\theta)$ from θ, read at the point specified in Section 5. The update class $\mathcal{N}$ is a distribution over such edge sets and $N \sim \mathcal{N}$ one draw. The unlearner knows how the class is built, but neither B nor the draw. We assume the update's edges and schedules

are chosen independently of the released model, so every release, the oracle included, receives the identical update.

Leakage and the problem. Two mean reciprocal ranks measure a model (Appendix C): the forget-MRR $\mathrm{MRR}_{\mathcal{F}}(\theta)$ over the forgotten facts, lower meaning more forgotten, and the utility $\mathrm{MRR}_{R}(\theta)$ over retained edges. An audit at release sees the gap $\mathrm{MRR}_{\mathcal{F}}(\theta_u) - \mathrm{MRR}_{\mathcal{F}}(\theta_g)$. The leakage is the rise of the release's forget-MRR under the update beyond the oracle's own rise,

$$\mathrm{Leak}(\theta_u; N) := \big[\mathrm{MRR}_{\mathcal{F}}(\mathrm{upd}_N(\theta_u)) - \mathrm{MRR}_{\mathcal{F}}(\theta_u)\big] - \big[\mathrm{MRR}_{\mathcal{F}}(\mathrm{upd}_N(\theta_g)) - \mathrm{MRR}_{\mathcal{F}}(\theta_g)\big], \quad (1)$$

in MRR points, read beside the utility $\mathrm{MRR}_{R}(\theta_u)$. The oracle has zero leakage, and negative leakage means slower relearning. Section 5 reads equation 1 along the update and adds two normalized readings. The unlearner sees neither the oracle nor the draw and cannot evaluate equation 1, but can act on each forgotten fact's post-update score. Its problem is to choose $\theta_u \in \Theta_{\mathrm{inv}}$ so that every forgotten fact stays forgotten after the update, in expectation over the class, at preserved retained scores:

$$\min_{\theta \in \Theta_{\mathrm{inv}}} \mathbb{E}_{N \sim \mathcal{N}}\Big[\frac{1}{|\mathcal{F}|}\sum_{f \in \mathcal{F}} \ell_f\big(d_f(\mathrm{upd}_N(\theta))\big)\Big] + \beta\, \ell_R(\theta). \quad (2)$$

Here ℓ_f is a forgetting loss on f's post-update score, zero above a target t_f fixed from θ° alone. The term ℓ_R preserves retained edges' scores under θ°, and β weighs the two. The obstacle is upd_N, a multi-step run on unknown edges. Section 4 replaces it by its closed-form linearized response.

4 KERNEL-ANTICIPATED UNLEARNING

KAU replaces $\mathrm{upd}_N(\theta)$ in the forgetting term of equation 2 by $\theta + \Delta\theta(\theta; N)$, where the linearized response $\Delta\theta(\theta; N)$ is the first-order parameter change from one update run on N. KAU's forgetting loss acts on the smaller of each forgotten fact's current and predicted post-update scores,

$$\mathcal{L}(\theta) = \mathbb{E}_{N \sim \mathcal{N}} \frac{1}{|\mathcal{F}|}\sum_{f \in \mathcal{F}} \mathrm{relu}\,\big(t_f - \tilde{d}_f\big)^2, \qquad \tilde{d}_f = \min\big\{d_f(\theta),\, d_f\big(\theta + \Delta\theta(\theta; N)\big)\big\}. \quad (3)$$

A lower score means stronger belief, so a fact counts as forgotten only if forgotten both now and after the predicted update. We call $d_f(\theta + \Delta\theta)$ the anticipated score, $\tilde{d}_f$ the composed score and t_f (equation 7) the target. Because $\Delta\theta$ depends on θ, the anticipated score's gradient shows which edits keep the fact forgotten after the update. KAU follows that gradient even when the minimum selects the current score (Appendix B.3).

4.1 THE UPDATE MODEL AND ITS RESIDUAL

The response needs a model of which edges the update trains on and how strongly it pulls each. KAU takes the edges from a random draw and measures each pull by a residual. Since the unlearner lacks the never-trained edges the class draws from B, KAU replaces $\mathcal{N}$ in equation 3 with a two-channel distribution $\hat{\mathcal{N}}$ over edge sets. The known channel includes each retained edge at an involved entity independently with a fixed probability. The unseen channel draws stand-ins for the never-trained edges. A stand-in pairs an involved entity with a relation from its retained edges and a partner entity the original model scores as plausible for that relation. No stand-in is a forgotten fact or a retained edge. Each draw keeps at most $k_{\max}$ edges per involved entity (Section 4.2). Each forgetting step, an optimizer step on equation 3, takes a fresh draw, so its loss stochastically estimates that objective. Averaging the loss over draws, rather than applying it to the mean response, avoids underestimating it under a first-order score expansion (Lemma A.3). We model the update, which minimizes the margin-ranking loss, as gradient descent on the drawn edges' squared residual $\frac{1}{2}\sum_{i \in N} r_i^2$, because that model has a closed-form linearized response (Section 4.2). Each residual is one-sided,

$$r_i(\theta) = \mathrm{relu}\,\big(d_i(\theta) - d_i(\theta^\circ)\big). \quad (4)$$

An edge the edit raised above its original score $d_i(\theta^\circ)$ is restored toward it, and an edge the edit lowered has zero residual. The one-sided form blocks a shortcut. Under the signed residual $d_i(\theta) - d_i(\theta^\circ)$, the modelled update would also restore edges the edit lowered. An edit could then lower the scores of edges at a fact's endpoints and let their modelled repair move the shared endpoint

rows, raising the fact's anticipated score. A loss on the anticipated score alone would be met while the released model still believes the fact. The minimum in equation 3 also blocks this shortcut, since the loss vanishes only when both scores reach the target. Section 4.6 still analyzes the signed residual, since it is linear in the edit under linearized scores. The modelled update also differs from the real one in which edges it pulls. It restores every raised edge, but the margin-ranking loss barely pulls an edge scored well over γ below its corruptions' mean (Lemma A.2). Such an edge still follows margin-violating edges, whose correction moves the whole endpoint row back along the edit (Appendix A.9). The residual can thus ignore how far below the margin an edge lies. Scaling it by the edge's margin-violation probability, the per-relation gate of Table 17, leaves leakage unchanged.

4.2 THE RESPONSE IN CLOSED FORM

The linearized response has a sparse closed form because a triple's score depends on only three rows. The slot feature of a triple x at its endpoint v is $\phi_x^{(v)} = \partial d_x / \partial \boldsymbol{e}_v \in \mathbb{R}^m$, taken through any normalization inside the score. The update trains relation rows too, but the response fixes them, as they contribute little to recovery (Appendix D.4). The full gradient $\phi_x = \nabla_\theta d_x$ is then nonzero only on x's two entity rows, so the kernel $\mathbf{K}_{NN} = \boldsymbol{\Phi}_N \boldsymbol{\Phi}_N^\top$ of the drawn edges' Jacobian $\boldsymbol{\Phi}_N$ couples only edges sharing an entity. Lemma 1 writes equation 4 as $r_i = d_i(\theta) - y_i$ with the level $y_i = \min\{d_i(\theta), d_i(\theta^\circ)\}$, the original score of a raised edge and the current score of a lowered one.

Lemma 1 (Linearized response, Doan et al., 2021; Bennani et al., 2020)**.** *Freeze the features ϕ_i of the update edges at θ and hold the levels y_i fixed, so that the residual $r_i = d_i - y_i$ of edge i moves linearly, $r_i(\theta + \Delta\theta) = r_i(\theta) + \phi_i^\top \Delta\theta$. For a ridge $\sigma > 0$, the minimizer over $\Delta\theta$ of the ridge-damped objective $\frac{1}{2}\sum_{i \in N} r_i(\theta + \Delta\theta)^2 + \frac{\sigma}{2}\|\Delta\theta\|_2^2$ is one damped Gauss–Newton step,*

$$\Delta\theta(\theta; N) = -\boldsymbol{\Phi}_N^\top \big(\mathbf{K}_{NN} + \sigma \mathbf{I}\big)^{-1} r_N(\theta). \tag{5}$$

The ridge σ models the update's finite duration and partial fit. For T steps at rate $\eta \leq 1/\lambda_{\max}(\mathbf{K}_{NN})$ and $\sigma = 1/(\eta T)$, ridge and gradient descent fit each eigen-direction of the kernel within a factor of two of each other (Remark A.2). The single step of equation 5 then approximates the whole frozen-feature run of gradient descent on the squared residual. The response leaves out Adam's per-coordinate scaling (Appendix A.12) and the features' motion. For Lipschitz features, re-linearizing once at the shifted point changes the response only at second order in the residual (Lemma A.1). KAU recomputes the drawn features at each forgetting step and freezes them during differentiation, so the response's gradient flows through the residual alone, without second derivatives.

KAU solves the lemma's problem over the involved rows only, in parameter space rather than through the kernel form of equation 5. Holding every other entity row fixed assigns the whole residual of a drawn edge to its involved endpoints. Let $\boldsymbol{\Phi}$ denote the drawn edges' Jacobian over the involved rows, Δ the stacked response, with block $\Delta\boldsymbol{e}_v$ at row v, and r the residuals of equation 4. KAU replaces the uniform ridge with the damping $\frac{1}{2}\Delta^\top \boldsymbol{\Sigma} \Delta$, where $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_v \mathbf{I})$ and σ_v is a fixed multiple of the mean squared feature norm at v. This absorbs the scale by which features differ across rows (Appendix A.12). The minimizer solves the joint system,

$$\big(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Sigma}\big)\Delta = -\boldsymbol{\Phi}^\top r. \tag{6}$$

Zero-residual edges still enter $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$, so the solve keeps lowered edges near their current scores.

The analysis uses a simpler per-entity form of equation 6, $\Delta\boldsymbol{e}_v = -\boldsymbol{A}_v r_v$, which drops the off-diagonal blocks of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ coupling entities that share a drawn edge. Here $\boldsymbol{A}_v = \boldsymbol{\Phi}_v^\top (\boldsymbol{\Phi}_v \boldsymbol{\Phi}_v^\top + \sigma_v \mathbf{I})^{-1}$, $\boldsymbol{\Phi}_v$ stacks the slot features at v of the drawn edges at v, and r_v collects their residuals. This form counts the residual of an edge between two involved entities twice, once at each endpoint (Lemma A.4), so KAU solves the joint system itself. The joint response also predicts per-fact realized recovery more closely than the per-entity form (Appendix D.5). The per-entity form explains the cap $k_{\max}$. At maximal rank (Appendix B.1), the drawn features at v span every forgotten fact's feature $\phi_f^{(v)}$. Let $\lambda_{\min}^+$ be the smallest nonzero eigenvalue of $\boldsymbol{\Phi}_v \boldsymbol{\Phi}_v^\top$. Under the signed residual and a first-order score expansion, the anticipated score's gradient through v's own response keeps at most $\sigma_v/(\lambda_{\min}^+ + \sigma_v)$ of $\|\phi_f^{(v)}\|_2$ (Theorem A.1), nearly nothing when $\sigma_v \ll \lambda_{\min}^+$. The ridge models update length and cannot be raised freely to offset this shrinkage. The cap keeps each draw's feature rank at v below that maximum, leaving some directions off the draw's span, where the edit survives the response. Fresh draws still cover the class over a run (Corollary A.3).

4.3 THE ANTICIPATED SCORE AND ITS TARGET

KAU evaluates the anticipated score exactly, adding $\Delta \boldsymbol{e}_v$ to the rows and calling the backbone's score function, not the first-order evaluation $d_f(\theta) + \sum_v \phi_f^{(v)\top} \Delta \boldsymbol{e}_v$. The score is nonlinear in the rows and the shift is not small relative to the score's per-coordinate terms, so a first-order expansion is not a controlled approximation (Appendix A.13). The shifted rows still come from the linearized response, so we judge the response by how well it ranks facts by recovery, not by the size it predicts. Exact evaluation ranks facts better than first-order evaluation (Appendix D.5) and lowers leakage (Table 16). Because the loss is one-sided, the target, not the response, sets how far each fact is pushed. It comes from the original model alone,

$$t_f = d_f(\theta^\circ) + s \ \mathrm{relu}\left(q_f - d_f(\theta^\circ)\right). \tag{7}$$

Here q_f is the 5% quantile of the original model's scores on corrupted copies of f, the level of a plausible false triple. A fact scored below that level gets a target a fraction s of the way toward it.

4.4 TWO CONTROLS ON THE RESPONSE

The edit can control both factors of a forgotten fact's first-order score change under the update. Write $\boldsymbol{A} = \boldsymbol{\Phi}_N^\top (\mathbf{K}_{NN} + \sigma \mathbf{I})^{-1}$ for the operator of equation 5, with its uniform ridge, and $r = r_N(\theta)$ for the residual on the drawn edges. In this expansion, which KAU replaces by exact evaluation (Section 4.3), the score of f changes by

$$\phi_f^\top \Delta\theta(\theta; N) = -\phi_f^\top \boldsymbol{A}\, r. \tag{8}$$

The change multiplies the residual r by the alignment $\phi_f^\top \boldsymbol{A}$ of the fact's feature with the range of $\boldsymbol{A}$. An edge with zero residual transmits nothing, so the first control is the residual on the known edges, which the unlearner sees and can limit. KAU thus adds the score-preservation term of equation 2,

$$\mathcal{L}_{\mathrm{tot}}(\theta) = \mathcal{L}(\theta) + \beta \ \mathrm{mean}_{i \in S} \left(d_i(\theta) - d_i(\theta^\circ)\right)^2. \tag{9}$$

Here S is a fresh sample of retained edges at $\mathcal{V}_\mathcal{F}$, drawn separately from N at each step. The two-sided term over all retained edges at $\mathcal{V}_\mathcal{F}$, estimated by S, bounds the one-sided residual of equation 4 and hence the known-channel response (Proposition A.2). The second control, alignment, sets how much of the residual left on the stand-ins and known edges reaches the fact. The response $-\boldsymbol{A}r$ lies in the range of $\boldsymbol{A}$, so it leaves the part of the edit off that range in place. Under the signed residual, it restores the edit most along directions that move the drawn edges' scores most (Proposition A.1). KAU exercises this control through $\mathcal{L}$, whose gradient in the linearized model favors the least-restored directions (Section 4.6).

4.5 THE FULL PROCEDURE

Each epoch runs a retained pass, then forgetting steps. The retained pass keeps utility by minimizing the margin-ranking loss on retained edges at $\mathcal{V}_\mathcal{F}$ plus the second term of equation 9. Each forgetting step draws $N \sim \hat{\mathcal{N}}$, forms the residual of equation 4, solves equation 6, scores the forgotten facts at the shifted rows and takes an Adam step on $\mathcal{L}$. After every optimizer step, a norm retraction resets each edited row to its norm under θ° (Appendix B.4). KAU releases the checkpoint maximizing a weighted sum of the forget-MRR drop and retained MRR on their development subsets (Appendix B.6). Algorithm 1 (Appendix B.7) lists one run. A release takes 13.4 seconds on CoDEx-L with TransE, against 72.8 for GraphDPO and 240.4 for retraining (Appendix B.10).

4.6 WHAT SURVIVES A LINEARIZED UPDATE

In a linearized model with the signed residual and a fixed draw's features frozen at θ°, an edit $\delta = \theta - \theta^\circ$ survives the update as $\boldsymbol{S}\delta$. Here $\boldsymbol{S} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1} \boldsymbol{\Sigma}$ is the survival filter, classical in ridge and early-stopped least squares (Yao et al., 2007; Ali et al., 2019) and linearized continual learning (Doan et al., 2021; Bennani et al., 2020). Under a uniform ridge $\boldsymbol{\Sigma} = \sigma \mathbf{I}$, $\boldsymbol{S}$ keeps $\sigma/(\lambda_k + \sigma)$ of each eigen-direction of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ with eigenvalue $\lambda_k > 0$ and everything off the features' span. Directions that strongly move drawn edges' scores keep little of the edit, weaker ones more. For every draw and per-entity ridge, the loss on the anticipated score alone is the plain, pre-update loss

on the surviving edit, $\mathcal{L}_{\text{ant}}(\delta) = \mathcal{L}_{\text{plain}}(\boldsymbol{S}\delta)$ (Corollary A.1). Under a uniform ridge, KAU's per-draw gradient for each fact lies along $\boldsymbol{S}\phi_f$, favoring directions restored least. Confining the edit to directions with $\lambda_k \leq \bar{\lambda}$ or off the features' span hard-thresholds the filter at $\bar{\lambda}$ (Corollary A.2), the form of several subspace methods. At $\bar{\lambda} = 0$ it is null-space projection onto edits with $\boldsymbol{\Phi}\delta = 0$, which the update never touches. Drawing exactly the involved entities' retained edges gives the retained-span projector on their rows (Golatkar et al., 2020b; Chen et al., 2024; Zhou et al., 2025). The real update's unseen edges can have features outside that span and move edits the projector keeps. Methods confining edits to slowly restored directions (Xiao et al., 2026; Sondej et al., 2026; Sondej & Yang, 2025) take the form of positive thresholds on spectra from forget or retained data. KAU instead weights directions softly, by their surviving fraction under draws whose stand-ins model unseen edges.

The filter explains why KAU composes the response into the loss rather than penalizing the predicted recovery $d_f(\theta) - d_f(\theta + \Delta\theta)$ beside a plain loss. That penalty is minimal at the unedited model, where all residuals vanish. Undoing a row's edit along its drawn features, which the update would do anyway, lowers the current score while the anticipated one barely moves (Proposition A.1). The penalty thus falls, rewarding the optimizer for undoing its forgetting. KAU's composed score caps that gain at a term vanishing with the ridge.

KAU freezes features at the current θ, where the anticipated score's first-order gradient through v's own response is $(\mathbf{I} - \boldsymbol{A}_v\boldsymbol{\Phi}_v)\phi_f^{(v)}$. Averaged over draws, these per-entity filters form a soft projector tending to v's retained-span projector with vanishing ridge and exactly the retained edges drawn. These results assume the signed residual and a ridge or gradient-descent fit, not Adam on the margin-ranking loss. On FB15k-237 with TransE, the plain loss and its retained-span projection leak more than KAU at matched forget-MRR drops (Section 5.5).

5 Experiments

We evaluate whether anticipating continued training improves the persistence of unlearning while preserving retained utility, against existing unlearners, relearning defences and ablations.

5.1 Experimental settings

Datasets. We evaluate graphs differing in source, size and relation sets: FB15k-237 (Toutanova & Chen, 2015), from Freebase with 14,541 entities and 237 relations, and CoDEx-L (Safavi & Koutra, 2020), from Wikidata with 77,951 entities and 69 relations. Let $\mathcal{D}_{\text{train}}$ denote the dataset's training triples. For each dataset, we sample a forget set $\mathcal{F}$ containing 10% of $\mathcal{D}_{\text{train}}$. We also reserve a disjoint set $B \subset \mathcal{D}_{\text{train}}$, containing approximately 2.2% of the training triples, to provide previously unseen facts for the relearning evaluation. The original model's training set is $\mathcal{D}_{\text{train}} \setminus B$, and the retained training set is $\mathcal{D}_R = \mathcal{D}_{\text{train}} \setminus (B \cup \mathcal{F})$.

Backbones and baselines. We use TransE (Bordes et al., 2013) as the primary backbone and evaluate TransH (Wang et al., 2014) and RotatE (Sun et al., 2019) on both datasets to test whether results extend across scoring functions. Baselines are GraphDPO (Liu et al., 2026), which uses preference optimization, and ZeroFisher (Xiao et al., 2025b), which uses zeroth-order curvature information. All methods use the same deletion request and evaluation procedure. Appendix C gives full experimental details and each method's forgetting and utility at release.

5.2 Evaluation metrics

All methods and the oracle receive the same update data, excluding the deleted facts (Section 3). We measure three quantities relative to the oracle, which itself regains some deleted facts from the remaining graph. **Leakage** (Leak) is the rise of a method's forget-set MRR beyond the oracle's rise, equation 1, in MRR points. Zero means the method relearns no faster than a model retrained without the deleted facts. **Relative excess recovery** (RER) is the leakage as a percentage of the erased knowledge, the original model's forget-set MRR minus the oracle's. **Excess recovery** (ExRec) compares the share of its own unlearning-induced drop that each model regains. It also reflects differences in forgetting depth, so Leak and RER are the primary measures. Each is read at the epoch of the method's peak leakage. We select each method's highest-leakage update from 48

Table 1: Relearning on FB15k-237 and CoDEx-L, with each method at its highest-leakage update configuration and peak-leakage epoch (Section 5.2). Means over seven unlearning seeds. Bold marks the best result, and Reduction is KAU's relative reduction from the better baseline.

		TransE				TransH				RotatE			
	Method	ExRec↓	Leak↓	RER↓	M_R↑	ExRec↓	Leak↓	RER↓	M_R↑	ExRec↓	Leak↓	RER↓	M_R↑
FB15k-237	GraphDPO	101.4	0.044	76.1	0.26	73.7	0.039	134.2	0.24	78.3	0.064	62.7	0.25
	ZeroFisher	140.0	0.074	129.0	0.23	124.3	0.069	234.0	0.22	124.3	0.086	83.8	0.23
	KAU	**48.5**	**0.022**	**37.5**	**0.28**	**6.6**	**0.021**	**72.0**	**0.26**	**23.8**	**0.024**	**23.4**	**0.33**
	Reduction (%)	52.1	50.8	50.8	–	91.1	46.4	46.4	–	69.6	62.7	62.7	–
CoDEx-L	GraphDPO	77.3	0.052	31.3	0.22	75.5	0.062	39.4	0.21	108.7	0.100	53.7	0.20
	ZeroFisher	132.0	0.108	64.5	0.17	135.0	0.144	91.3	0.15	147.6	0.133	71.3	0.17
	KAU	**26.9**	**0.021**	**12.8**	**0.25**	**27.1**	**0.031**	**19.7**	**0.25**	**24.9**	**0.018**	**9.6**	**0.31**
	Reduction (%)	65.2	59.1	59.1	–	64.1	50.1	50.1	–	77.1	82.1	82.1	–

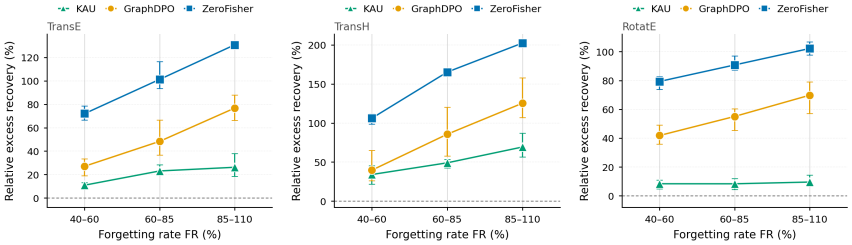


Figure 3: Relative excess recovery versus forgetting depth on FB15k-237, one update configuration per backbone. Points average three unlearning seeds. Each method is released at three depths by varying only its budget, and the release nearest each band is shown.

shared configurations using the selection half of a 1,000-triple deleted-fact probe and report on the other half. M_R is the retain-set MRR at release. Appendix C gives the formulas and the grid.

5.3 KAU IMPROVES ROBUSTNESS TO RELEARNING

Table 1 reports robustness on both datasets and three backbones under each method's highest-leakage configuration chosen on the selection half. KAU attains the lowest ExRec, Leak and RER on every dataset and backbone. On FB15k-237, it cuts the strongest baseline's leakage by about half on TransE and TransH and nearly two thirds on RotatE. On CoDEx-L it reduces the leakage of GraphDPO by 59% on TransE, 50% on TransH and 82% on RotatE. It keeps the retained MRR above both baselines on every backbone. The M_R column shows that both baselines disturb retained facts around deleted entities to reach their forgetting level, whereas KAU keeps them near their trained scores. By fitting retained or never-trained true edges at shared entity rows, the update restores deleted facts sharing those rows (Appendices D.2 and D.4). KAU leaves the update little to restore and places its edit where the update reaches least, so less of the deleted knowledge returns.

KAU stays closest to the oracle throughout the TransE update (Figure 2). Its advantage widens as forgetting deepens (Figure 3). Depth is the forgetting rate (FR), the forget-MRR drop at release as a percentage of the oracle's. KAU achieves the lowest mean RER at every depth and backbone. From shallowest to deepest release, its RER rises by less than half as much as either baseline's.

5.4 RELEARNING-RESISTANT ADD-ONS DO NOT CLOSE THE GAP

We test whether established relearning defences close the leakage gap when added to our strongest baseline, GraphDPO, on FB15k-237. We use NTK projection (Golatkar et al., 2020b), sharpness-aware minimization (SAM) (Fan et al., 2025b) and tamper-resistance training (TAR) (Tamirisa et al., 2025), each added to GraphDPO at its selected setting (Table 2, Appendix C.5). NTK projection and SAM constrain edit directions using retained gradients and flatten the loss, respectively. Neither directly optimizes forgotten-fact scores after continued training. TAR simulates updates but leaks

Table 2: Relearning-resistant add-ons on GraphDPO, means over three unlearning seeds. Each method uses its strongest update configuration at peak leakage, which may differ from Table 1. Bold marks the best result, and the last row gives KAU's reduction over the best add-on.

	TransE				TransH				RotatE			
Method	ExRec↓	Leak↓	RER↓	M_R↑	ExRec↓	Leak↓	RER↓	M_R↑	ExRec↓	Leak↓	RER↓	M_R↑
GraphDPO	98.6	0.047	81.8	0.26	78.3	0.044	149.4	0.24	76.8	0.065	63.1	0.25
GraphDPO+NTK	95.7	0.046	79.7	0.26	74.8	0.042	144.7	0.24	69.7	0.067	65.2	0.25
GraphDPO+SAM	101.5	0.042	72.8	0.26	80.1	0.040	136.3	0.24	76.6	0.067	65.7	0.25
GraphDPO+TAR	91.0	0.043	74.6	0.26	79.1	0.040	136.9	0.24	75.4	0.066	64.6	0.25
KAU	**46.3**	**0.022**	**37.7**	**0.28**	**6.5**	**0.023**	**77.0**	**0.26**	**26.2**	**0.026**	**25.3**	**0.32**
Reduction (%)	49.1	48.3	48.3	–	91.3	43.5	43.5	–	62.5	60.9	60.9	–

no less than SAM on TransE and TransH. No add-on reduces leakage by more than 11%, and none lowers GraphDPO's leakage on RotatE. KAU leaks 43–61% less than the best add-on and retains more on every backbone. These results indicate that hardening GraphDPO's objective does not reproduce KAU's gains in forgetting persistence and retained utility.

5.5 The anticipated objective outperforms the tested alternatives

Ablation of KAU. We isolate how KAU's components and approximations affect leakage and retained utility within a shared forgetting-rate band and update configuration (Table 16 in Appendix D.7). Removing anticipation more than doubles leakage and causes the largest increase in excess recovery. Per-entity block solves and first-order score evaluation cause the next largest leakage increases, supporting the joint solve and exact evaluation. Removing either update channel increases leakage by roughly one third, and removing the known channel lowers retained MRR from 0.2860 to 0.2834. Using a signed residual also raises leakage. Without score preservation, leakage rises and retained MRR falls furthest, to 0.2788. Adding a retained-span projector increases leakage, consistent with the distinction between hard projection and the graded response (Section 4.6).

Alternative forgetting objectives. We test whether existing relearning-robust objectives can reproduce KAU's robustness when substituted for its anticipated forgetting loss. On FB15k-237 with TransE, only the forgetting step changes, with releases in a shared forgetting-rate band (Table 18 in Appendix D.7). Alternatives include retained-span projection (Chen et al., 2024; Hoang et al., 2024; Zhou et al., 2025), SAM variants (Fan et al., 2025b), weight-distance regularization, connectivity-based fine-tuning and weight distortion (Lubana et al., 2023; Siddiqui et al., 2025), TAR-style unrolling (Tamirisa et al., 2025), Booster (Huang et al., 2025) and NTK scrubbing (Golatkar et al., 2020b). Plain forgetting sets $\Delta = 0$ (Table 16). KAU achieves the lowest ExRec, Leak and RER. The plain forgetting loss leaks 0.0250 forget-MRR beyond the retrained oracle, and the retained-span projection leaks 0.0158. All alternatives except weight distortion reduce leakage relative to plain forgetting, yet none matches KAU: the best alternative, SAM-GP, reaches 0.0150, against KAU's 0.0111. TAR's ten-step unrolling and Booster's one-step retained-loss proxy also leave a gap. KAU evaluates forgotten-fact scores after a coupled response to sampled retained and stand-in edges. NTK scrubbing and weight distortion have the highest ExRec among the alternatives. With the surrounding procedure controlled and forgetting depth comparable, these results support the anticipated objective as a source of robustness beyond the tested regularizers and update proxies.

6 Conclusion

We showed that existing knowledge-graph unlearners forget each fact where its entity embeddings sit at release, not where continued training will leave them. Under ordinary continued training, the deleted facts therefore return beyond what a retrained model recovers, which an audit at release cannot detect, undermining deletion requests in practice. To overcome this, we propose KAU, which predicts in closed form, through the tangent kernel, how continued training moves each forgotten fact and forgets it after that move. Our results confirm that KAU substantially reduces the relearning of forgotten facts while preserving the retained knowledge. By forgetting each fact where continued training will leave it, our approach is a promising step toward deletions that last in knowledge-graph embeddings.

REPRODUCIBILITY STATEMENT

All experiments use the public benchmarks FB15k-237 and CoDEx-L with TransE, TransH and RotatE, each with a fixed-seed 10% forget split. Appendix C gives full experimental details, including the split, update fine-tuning data and 48 configurations, shared selection rule, CoDEx-L settings, and training recipes for the original and retrained models. Appendices B and C.4 list all FB15k-237 unlearner hyperparameters, and Appendix B.10 gives the GPU and each method's time per release. All original, retrained, released and updated checkpoints are published with per-edge evaluation files, from which every reported statistic can be recomputed without a GPU. Proofs appear in Appendix A, with each assumption in its corresponding result statement. The code is available at `https://anonymous.4open.science/r/Relearning-Aware-Unlearning-of-Knowledge-Graph-Embeddings-via-Tangent-Kernels-1F64/`.

ETHICS STATEMENT

We ask whether a deletion request honored by an approximate unlearner survives ordinary continued training. This bears on the right to be forgotten for knowledge in embeddings when released models are edited rather than retrained. We study a standard training run that downstream users already perform on true edges with no deletion request, giving them no additional capability. We release the method to make deletions more durable. Both datasets are public benchmarks: FB15k-237 derives from Freebase and CoDEx-L from Wikidata, and both contain facts about public entities, including public figures. No personal data were collected and no human subjects were involved.

AI USE STATEMENT

We used AI tools to assist with writing and presentation, including rewriting text, correcting grammar, and improving clarity, readability, and logical flow. We also used AI tools to assist in auditing mathematical proofs and exploring theoretical arguments. All AI-assisted outputs were reviewed, verified, and revised by the authors, who remain responsible for the correctness and content of the paper.

REFERENCES

Alnur Ali, J. Zico Kolter, and Ryan J. Tibshirani. A Continuous-Time View of Early Stopping for Least Squares Regression. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1370–1378, 2019.

Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical Analysis of Auto Rate-Tuning by Batch Normalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. URL `https://arxiv.org/abs/1812.03981`.

Mehdi Abbana Bennani, Thang Doan, and Masashi Sugiyama. Generalisation Guarantees for Continual Learning with Orthogonal Gradient Descent. *arXiv preprint arXiv:2006.11942*, 2020. URL `https://arxiv.org/abs/2006.11942`.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013. URL `https://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data`.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine Unlearning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2021. URL `https://arxiv.org/abs/1912.03817`.

Yinzhi Cao and Junfeng Yang. Towards Making Systems Forget with Machine Unlearning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 463–480, 2015. URL `https://doi.org/10.1109/SP.2015.35`.

Wei-Kai Chang and Rajiv Khanna. Why Some Models Resist Unlearning: A Linear Stability Perspective. *arXiv preprint arXiv:2602.02986*, 2026. URL https://arxiv.org/abs/2602.02986.

Zora Che, Stephen Casper, Robert Kirk, Anirudh Satheesh, Stewart Slocum, Lev E. McKinney, Rohit Gandikota, Aidan Ewart, Domenic Rosati, Zichu Wu, Zikui Cai, Bilal Chughtai, Yarin Gal, Furong Huang, and Dylan Hadfield-Menell. Model Tampering Attacks Enable More Rigorous Evaluations of LLM Capabilities. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2502.05209.

Huiqiang Chen, Tianqing Zhu, Xin Yu, and Wanlei Zhou. Machine Unlearning via Null Space Calibration. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2024. URL https://arxiv.org/abs/2404.13588.

Yi Chen, Hanna Hsieh, Shuhong Liu, Chuanbo Hua, Zihan Ma, Kun Wang, and Joo-Young Kim. Distance Is Not Enough: Forget-Retain Alignment Gap Predicts LLM Relearning Robustness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2026. URL https://arxiv.org/abs/2608.25429.

Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. GNNDelete: A General Strategy for Unlearning in Graph Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2302.13406.

Weilin Cong and Mehrdad Mahdavi. Efficiently Forgetting What You Have Learned in Graph Representation Learning via Projection. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 206, pp. 6674–6703, 2023. URL https://proceedings.mlr.press/v206/cong23a.html.

Aghyad Deeb and Fabien Roger. Do Unlearning Methods Remove Information from Language Model Weights? *arXiv preprint arXiv:2410.08827*, 2024. URL https://arxiv.org/abs/2410.08827.

Zahra Dehghani, Pablo Piantanida, and Mohammadhadi Shateri. Source-Free Class Relearning: Diagnosing Forgetting in Class Unlearning. *arXiv preprint arXiv:2609.02018*, 2026. URL https://arxiv.org/abs/2609.02018.

Thang Doan, Mehdi Bennani, Bogdan Mazoure, Guillaume Rabusseau, and Pierre Alquier. A Theoretical Analysis of Catastrophic Forgetting through the NTK Overlap Matrix. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. URL https://arxiv.org/abs/2010.04003.

Bowen Fan, Yuming Ai, Xunkai Li, Zhilin Guo, Zhu Lei, Guang Zeng, Rong-Hua Li, and Guoren Wang. OpenGU: A Comprehensive Benchmark for Graph Unlearning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025a. URL https://arxiv.org/abs/2501.02728.

Chongyu Fan, Jinghan Jia, Yihua Zhang, Anil Ramakrishna, Mingyi Hong, and Sijia Liu. Towards LLM Unlearning Resilient to Relearning Attacks: A Sharpness-Aware Minimization Perspective and Beyond. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025b. URL https://arxiv.org/abs/2502.05374.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.02355.

Hongcheng Gao, Tianyu Pang, Chao Du, Taihang Hu, Zhijie Deng, and Min Lin. Meta-Unlearning on Diffusion Models: Preventing Relearning Unlearned Concepts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. URL https://arxiv.org/abs/2410.12777.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a. URL https://arxiv.org/abs/1911.04933.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020b. URL https://arxiv.org/abs/2003.02960.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified Data Removal from Machine Learning Models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/1911.03030.

SeungBum Ha, Saerom Park, and Sung Whan Yoon. Unlearning's Blind Spots: Over-Unlearning and Prototypical Relearning Attack. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2506.01318.

Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. AdamP: Slowing Down the Slowdown for Momentum Optimizers on Scale-invariant Weights. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2006.08217.

Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952. URL https://nvlpubs.nist.gov/nistpubs/jres/049/6/V49.N06.A08.pdf.

Ido Nitzan Hidekel and Dan Raviv. Catastrophic Forgetting is Low-Rank: A Function-Space Theory for Continual Adaptation. In *ICML 2026 Workshop on Continual Adaptation at Scale: Towards Sustainable AI*, 2026. URL https://arxiv.org/abs/2606.18024.

Tuan Hoang, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Learn to Unlearn for Deep Neural Networks: Minimizing Unlearning Interference with Gradient Projection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024. URL https://arxiv.org/abs/2312.04095.

Shengyuan Hu, Yiwei Fu, Zhiwei Steven Wu, and Virginia Smith. Unlearning or Obfuscating? Jogging the Memory of Unlearned LLMs via Benign Relearning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2406.13356.

Tiansheng Huang, Sihao Hu, Fatih Ilhan, Selim Furkan Tekin, and Ling Liu. Booster: Tackling Harmful Fine-tuning for Large Language Models via Attenuating Harmful Perturbation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2409.01586.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate Data Deletion from Machine Learning Models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. URL https://arxiv.org/abs/2002.10077.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1806.07572.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. Knowledge Unlearning for Mitigating Privacy Risks in Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://arxiv.org/abs/2210.01504.

Jiaming Ji, Kaile Wang, Tianyi Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Juntao Dai, Yunhuai Liu, and Yaodong Yang. Language Models Resist Alignment: Evidence From Data Compression. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. URL https://arxiv.org/abs/2406.06144.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017. URL https://arxiv.org/abs/1703.04730.

Bruce W. Lee, Addie Foote, Alex Infanger, Leni Shor, Harish Kamath, Jacob Goldman-Wetzler, Bryce Woodworth, Alex Cloud, and Alexander Matt Turner. Distillation Robustifies Unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.06278.

Boheng Li, Renjie Gu, Junjie Wang, Leyi Qi, Yiming Li, Run Wang, Zhan Qin, and Tianwei Zhang. Towards Resilient Safety-driven Unlearning for Diffusion Models against Downstream Fine-tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2507.16302.

Jiajun Liu, Wenjun Ke, Peng Wang, Ziyu Shang, Jinhua Gao, Guozheng Li, Ke Ji, and Yanhe Liu. Towards Continual Knowledge Graph Embedding via Incremental Distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2405.04453.

Jiajun Liu, Wenjun Ke, Peng Wang, Yao He, Ziyu Shang, Guozheng Li, Zijie Xu, and Ke Ji. Unlearning of Knowledge Graph Embedding via Preference Optimization. In *Proceedings of the ACM Web Conference (WWW)*, 2026. URL https://arxiv.org/abs/2507.20566.

Ekdeep Singh Lubana, Eric J. Bigelow, Robert P. Dick, David Krueger, and Hidenori Tanaka. Mechanistic Mode Connectivity. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22965–23004, 2023.

Jakub Łucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An Adversarial Perspective on Machine Unlearning for AI Safety. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2409.18025.

Sadhika Malladi, Alexander Wettig, Dingli Yu, Danqi Chen, and Sanjeev Arora. A Kernel-Based View of Language Model Fine-Tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2210.05643.

Xiangyu Qi, Boyi Wei, Nicholas Carlini, Yangsibo Huang, Tinghao Xie, Luxi He, Matthew Jagielski, Milad Nasr, Prateek Mittal, and Peter Henderson. On Evaluating the Durability of Safeguards for Open-Weight LLMs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2412.07097.

Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1909.04630.

Domenic Rosati, Xijie Zeng, Hong Huang, Sebastian Dionicio, Subhabrata Majumdar, Frank Rudzicz, and Hassan Sajjad. Limits of Convergence-Rate Control for Open-Weight Safety. *arXiv preprint arXiv:2602.18868*, 2026. URL https://arxiv.org/abs/2602.18868.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8328–8350. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.669. URL https://aclanthology.org/2020.emnlp-main.669/.

Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient Projection Memory for Continual Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2103.09762.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember What You Want to Forget: Algorithms for Machine Unlearning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2103.03279.

Shoaib Ahmed Siddiqui, Adrian Weller, David Krueger, Gintare Karolina Dziugaite, Michael Curtis Mozer, and Eleni Triantafillou. From Dormant to Deleted: Tamper-Resistant Unlearning Through Weight-Space Regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.22310.

Filip Sondej and Yushi Yang. Collapse of Irrelevant Representations (CIR) Ensures Robust and Non-Disruptive LLM Unlearning. *arXiv preprint arXiv:2509.11816*, 2025. URL https://arxiv.org/abs/2509.11816.

Filip Sondej, Yushi Yang, Mikołaj Kniejski, and Marcel Windys. Robust LLM Unlearning with MUDMAN: Meta-Unlearning with Disruption Masking And Normalization. *arXiv preprint arXiv:2506.12484*, 2025. URL https://arxiv.org/abs/2506.12484.

Filip Sondej, Yushi Yang, and Adam Mahdi. RepSelect: Robust LLM Unlearning via Representation Selectivity. *arXiv preprint arXiv:2606.17168*, 2026. URL https://arxiv.org/abs/2606.17168.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1902.10197.

Vinith M. Suriyakumar, Rohan Alur, Ayush Sekhari, Manish Raghavan, and Ashia C. Wilson. Unstable Unlearning: The Hidden Risk of Concept Resurgence in Diffusion Models. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2410.08074.

Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks, and Mantas Mazeika. Tamper-Resistant Safeguards for Open-Weight LLMs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2408.00761.

Kristina Toutanova and Danqi Chen. Observed versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pp. 57–66, 2015. URL https://aclanthology.org/W15-4007/.

Khoa Tran and Simon S. Woo. Efficient Unlearning through Maximizing Relearning Convergence Delay. *arXiv preprint arXiv:2604.09391*, 2026. URL https://arxiv.org/abs/2604.09391.

Changsheng Wang, Yihua Zhang, Jinghan Jia, Parikshit Ram, Dennis Wei, Yuguang Yao, Soumyadeep Pal, Nathalie Baracaldo, and Sijia Liu. Invariance Makes LLM Unlearning Resilient Even to Unanticipated Downstream Fine-Tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2506.01339.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training Networks in Null Space of Feature Covariance for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Wang_Training_Networks_in_Null_Space_of_Feature_Covariance_for_Continual_CVPR_2021_paper.html.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1112–1119, 2014.

Rongzhe Wei, Peizhi Niu, Hans Hao-Hsun Hsu, Ruihan Wu, Haoteng Yin, Mohsen Ghassemi, Yifan Li, Vamsi K. Potluru, Eli Chien, Kamalika Chaudhuri, Olgica Milenkovic, and Pan Li. Do LLMs Really Forget? Evaluating Unlearning with Knowledge Correlation and Confidence Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2506.05735.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. GIF: A General Graph Unlearning Strategy via Influence Function. In *Proceedings of the ACM Web Conference (WWW)*, 2023. URL https://arxiv.org/abs/2304.02835.

Yang Xiao, Ruimeng Ye, and Bo Hui. Knowledge Graph Unlearning with Schema. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025a. URL https://aclanthology.org/2025.coling-main.238/.

Yang Xiao, Ruimeng Ye, Bohan Liu, Xiaolong Ma, and Bo Hui. Efficient Knowledge Graph Unlearning with Zeroth-order Information. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2025b. URL https://arxiv.org/abs/2508.14013.

Zeguan Xiao, Xuanzhe Xu, Yun Chen, Yong Wang, Jian Yang, Yanqing Hu, and Guanhua Chen. Robust LLM Unlearning Against Relearning Attacks: The Minor Components in Representations Matter. *arXiv preprint arXiv:2605.11685*, 2026. URL https://arxiv.org/abs/2605.11685.

Naixing Xu, Qian Li, Zhaochuan Li, Xu Wang, Bingchen Liu, Jabulani Brown Mpofu, Jingchen Li, and Xin Li. Learn to Unlearn: Meta-Learning-Based Knowledge Graph Embedding Unlearning. *Knowledge and Information Systems*, 67(8):7101–7122, 2025. URL https://doi.org/10.1007/s10115-025-02407-4.

Xiaoyu Xu, Xiang Yue, Yang Liu, Qingqing Ye, Huadi Zheng, Peizhao Hu, Minxin Du, and Haibo Hu. Unlearning Isn't Deletion: Investigating Reversibility of Machine Unlearning in LLMs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2505.16831.

Nakyeong Yang, Dong-Kyum Kim, Jea Kwon, Minsung Kim, Kyomin Jung, and Meeyoung Cha. Erase or Hide? Suppressing Spurious Unlearning Neurons for Robust Unlearning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2509.22263.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On Early Stopping in Gradient Descent Learning. *Constructive Approximation*, 26(2):289–315, 2007.

Hang Yin, Zipeng Liu, Xiaoyong Peng, and Liyao Xiang. Graph Unlearning via Embedding Reconstruction – A Range-Null Space Decomposition Approach. *arXiv preprint arXiv:2508.02044*, 2025. URL https://arxiv.org/abs/2508.02044.

Xiangyu Yin, Jiaxu Liu, Zhen Chen, and Chih-Hong Cheng. Crossing the Margin Cliff: Toward Relearn-Robust LLM Unlearning via Margin Calibration. *arXiv preprint arXiv:2607.27836*, 2026. URL https://arxiv.org/abs/2607.27836.

Sangyeon Yoon, Hyesoo Hong, Wonje Jeung, and Albert No. Rethinking Benign Relearning: Syntax as the Hidden Driver of Unlearning Failures. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.03379.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative Preference Optimization: From Catastrophic Collapse to Effective Unlearning. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2404.05868.

Duo Zhou, Yuji Zhang, Tianxin Wei, Ruizhong Qiu, Ke Yang, Xiao Lin, Cheng Qian, Jingrui He, Hanghang Tong, Chengxiang Zhai, Heng Ji, and Huan Zhang. Geometric-Disentanglement Unlearning. *arXiv preprint arXiv:2511.17100*, 2025. URL https://arxiv.org/abs/2511.17100.

A Analysis and proofs

Section 4.6 summarizes what the update does to an edit. Appendix A.1 states that analysis in full, in Proposition A.1, Corollaries A.1, A.2 and A.3, Theorem A.1 and Remark A.1. The remaining subsections prove these results and Lemma 1, and they state the supporting results that Section 4 cites.

Conventions. Throughout, v is an involved entity, N a fixed draw, $P_v(N) = \{i \in N : v \in i\}$ the set of drawn edges incident to v, and $n_v = |P_v(N)|$. A drawn edge is a triple of the draw N, and it contributes one row to every Jacobian below. The slot feature of edge i at v is $\phi_i^{(v)} = \partial d_i / \partial \boldsymbol{e}_v \in \mathbb{R}^m$, and $\boldsymbol{\Phi}_v \in \mathbb{R}^{n_v \times m}$ stacks $\phi_i^{(v)\top}$ over $i \in P_v(N)$. For the two backbones with normalized rows the slot feature has the form

$$\phi_i^{(v)} = \frac{(\mathbf{I} - \bar{\boldsymbol{e}}_v \bar{\boldsymbol{e}}_v^\top)\, \boldsymbol{g}_i^{(v)}}{\|\boldsymbol{e}_v\|_2}, \tag{10}$$

where $\boldsymbol{g}_i^{(v)}$ is the gradient of the score with respect to the normalized row $\bar{\boldsymbol{e}}_v$. For TransE, $\boldsymbol{g}_i^{(v)} = \pm \operatorname{sign}(\bar{\boldsymbol{e}}_h + \bar{\boldsymbol{w}}_r - \bar{\boldsymbol{e}}_t)$, with the plus sign at the head and the minus sign at the tail, defined wherever no coordinate of the translation error $\bar{\boldsymbol{e}}_h + \bar{\boldsymbol{w}}_r - \bar{\boldsymbol{e}}_t$ vanishes. For TransH, $\boldsymbol{g}_i^{(v)} = \pm \boldsymbol{P}_r \operatorname{sign}(\boldsymbol{P}_r \bar{\boldsymbol{e}}_h + \boldsymbol{d}_r - \boldsymbol{P}_r \bar{\boldsymbol{e}}_t)$, the sign vector of the TransH translation error multiplied by the symmetric projector $\boldsymbol{P}_r$. In both cases every slot feature at v lies in the tangent space $\bar{\boldsymbol{e}}_v^\perp$ of the unit sphere at $\bar{\boldsymbol{e}}_v$, a subspace of dimension $m - 1$. RotatE does not normalize its rows and its features carry no such constraint.

Let $\boldsymbol{\Phi}_v = \sum_{k \le \varrho} \sqrt{\lambda_k}\, \boldsymbol{q}_k \boldsymbol{u}_k^\top$ be a singular value decomposition, with $\varrho = \operatorname{rank} \boldsymbol{\Phi}_v$, $\lambda_1 \ge \cdots \ge \lambda_\varrho > 0$ the nonzero eigenvalues of $\mathbf{K}_v = \boldsymbol{\Phi}_v \boldsymbol{\Phi}_v^\top$, $\boldsymbol{q}_k \in \mathbb{R}^{n_v}$ the corresponding orthonormal eigenvectors, and $\boldsymbol{u}_k = \boldsymbol{\Phi}_v^\top \boldsymbol{q}_k / \sqrt{\lambda_k} \in \mathbb{R}^m$ an orthonormal basis of $\operatorname{range}(\boldsymbol{\Phi}_v^\top)$. With $\sigma_v > 0$,

$$\begin{aligned} \boldsymbol{A}_v &= \boldsymbol{\Phi}_v^\top (\mathbf{K}_v + \sigma_v \mathbf{I})^{-1} = \sum_{k \le \varrho} \frac{\sqrt{\lambda_k}}{\lambda_k + \sigma_v}\, \boldsymbol{u}_k \boldsymbol{q}_k^\top, \\ \boldsymbol{P}_{N,v}^\perp &= \mathbf{I} - \boldsymbol{A}_v \boldsymbol{\Phi}_v = \sum_{k \le \varrho} \frac{\sigma_v}{\lambda_k + \sigma_v}\, \boldsymbol{u}_k \boldsymbol{u}_k^\top + \Pi_{N,v}^\perp, \end{aligned} \tag{11}$$

where $\Pi_{N,v} = \sum_{k \le \varrho} \boldsymbol{u}_k \boldsymbol{u}_k^\top$ is the orthogonal projector onto the span of the drawn features at v and $\Pi_{N,v}^\perp = \mathbf{I} - \Pi_{N,v}$. The first identity holds because $(\mathbf{K}_v + \sigma_v \mathbf{I})^{-1}$ has the eigenvectors $\boldsymbol{q}_k$ with eigenvalues $1/(\lambda_k + \sigma_v)$ on $\operatorname{range}(\boldsymbol{\Phi}_v)$ and $\boldsymbol{\Phi}_v^\top$ annihilates its orthogonal complement. The second follows by multiplying out. Three facts are read off equation 11 and used below. The operator $\boldsymbol{P}_{N,v}^\perp$ is symmetric with spectrum $\{\sigma_v/(\lambda_k + \sigma_v)\}_{k \le \varrho}$, together with 1 when $\varrho < m$. It is therefore positive definite, and it is idempotent only when $\varrho = 0$, where it is the identity. It converges to $\Pi_{N,v}^\perp$ as $\sigma_v \to 0$. Its companion satisfies $\|\boldsymbol{A}_v\|_{\mathrm{op}} = \max_k \sqrt{\lambda_k}/(\lambda_k + \sigma_v) \le 1/(2\sqrt{\sigma_v})$, because $\sqrt{\lambda}/(\lambda + \sigma)$ is maximized over $\lambda \ge 0$ at $\lambda = \sigma$. When no edge is drawn at v, $\boldsymbol{\Phi}_v$ is empty, $\Delta \boldsymbol{e}_v = 0$ and $\boldsymbol{P}_{N,v}^\perp = \mathbf{I}$, and statements that involve $\lambda_{\min}^+$ are made for $\varrho \ge 1$.

A.1 What survives a linearized update

This subsection gives the full statements behind Section 4.6. It shows what the update does to an edit, and what that implies for the objective of equation 3 and for subspace methods. In a linearized model the update multiplies the edit by a fixed matrix, the survival filter, and the anticipated loss is the plain forgetting loss on the filtered edit. Subspace methods confine the edit to a hard threshold of the same filter. Theorem A.1 gives the local version for the nonlinear objective, and Corollary A.3 averages it over draws.

The linearized model. Fix a draw N and write $\delta = \theta - \theta^\circ$ for the edit of the involved rows. The linearized model freezes the stacked features $\boldsymbol{\Phi}$ of the drawn edges and the feature ϕ_f of every forgotten fact at the original model. Every score is then linear in the edit, $d_f(\theta^\circ + \delta) = d_f(\theta^\circ) + \phi_f^\top \delta$. The residual is the signed residual $r(\theta) = \boldsymbol{\Phi}(\theta - \theta^\circ)$, and the response is the joint solve of equation 6, $\Delta\theta(\theta) = -(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1} \boldsymbol{\Phi}^\top r(\theta)$. The proposition below shows that the update returns the edit multiplied by a fixed matrix, and it gives the fraction of each direction that survives.

Proposition A.1 (The update filters the edit)**.** *In the linearized model, every edit δ satisfies*

$$\theta^\circ + \delta + \Delta\theta(\theta^\circ + \delta) = \theta^\circ + \boldsymbol{S}\delta, \qquad \boldsymbol{S} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}, \tag{12}$$

and $\theta + \delta + \Delta\theta(\theta + \delta) = \theta + \Delta\theta(\theta) + \boldsymbol{S}\delta$ at every θ. Moreover, $\boldsymbol{S}\delta = \delta$ if and only if $\boldsymbol{\Phi}\delta = 0$. For a uniform ridge $\boldsymbol{\Sigma} = \sigma\mathbf{I}$, with $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$,

$$\boldsymbol{S} = \mathbf{I} - \boldsymbol{\Phi}^\top(\mathbf{K} + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi} = \sum_k \frac{\sigma}{\lambda_k + \sigma}\,\boldsymbol{u}_k\boldsymbol{u}_k^\top + \Pi^\perp,$$

where $(\lambda_k, \boldsymbol{u}_k)$ are the nonzero eigenpairs of $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ and $\Pi^\perp$ is the orthogonal projector off the span of the update's features. The component of the edit along $\boldsymbol{u}_k$ survives with the fraction $\sigma/(\lambda_k + \sigma)$, and the component off the span survives entirely. Let the update be T steps of gradient descent at rate $\eta \le 1/\lambda_{\max}(\mathbf{K})$ on the linearized squared residual $\frac{1}{2}\|r\|_2^2$, started at the edited model. The surviving edit is then exactly $(\mathbf{I} - \eta\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^T\delta$, and its fraction along $\boldsymbol{u}_k$ satisfies $(1 - \eta\lambda_k)^T \le \sigma/(\lambda_k + \sigma)$ at $\sigma = 1/(\eta T)$. For an edit that moves the row $\boldsymbol{e}_v$ alone, by δ_v, the per-entity response $\Delta\boldsymbol{e}_v = -\boldsymbol{A}_v r_v$ of Section 4.2 obeys the same identities with the per-entity filter $\boldsymbol{P}_{N,v}^\perp = \mathbf{I} - \boldsymbol{\Phi}_v^\top(\boldsymbol{\Phi}_v\boldsymbol{\Phi}_v^\top + \sigma_v\mathbf{I})^{-1}\boldsymbol{\Phi}_v$ in place of $\boldsymbol{S}$ and δ_v in place of δ. For $\boldsymbol{\Phi}_v \neq 0$, every displacement $\delta_v \in \mathrm{range}(\boldsymbol{\Phi}_v^\top)$ of the row $\boldsymbol{e}_v$ satisfies $\|\boldsymbol{P}_{N,v}^\perp\delta_v\|_2 \le \sigma_v/(\lambda_{\min}^+ + \sigma_v)\,\|\delta_v\|_2$, with $\lambda_{\min}^+$ the smallest nonzero eigenvalue of $\boldsymbol{\Phi}_v\boldsymbol{\Phi}_v^\top$.

An update of finite length restores the directions in which its features are large and leaves the rest of the edit in place. The spectral form is the classical spectral filter of ridge and early-stopped least squares (Yao et al., 2007; Ali et al., 2019) (Remark A.2). It is also the operator that governs interference between tasks in linearized continual learning (Doan et al., 2021; Bennani et al., 2020). With the Gauss–Newton Hessian $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ and the uniform ridge as proximal weight, it is also the Jacobian of a proximal model of adaptation with respect to its centre (Rajeswaran et al., 2019). Li et al. (2025) use its full-Hessian form at a simulated fine-tune's end point to differentiate a forgetting loss, whereas KAU's response is the closed-form proximal point of the linearized squared residual. The proposition records that filter for the edit. What is new is its use in Corollaries A.1 and A.2. Because the signed residual vanishes at the original model, the anticipated loss is the plain loss on the filtered edit, and confinement to a subspace is its hard threshold. The identity of equation 12 and the identity of Corollary A.1 hold for every positive diagonal $\boldsymbol{\Sigma}$, which covers the per-entity ridge of the experiments. The spectral form and Corollary A.2 take the uniform ridge, under which $\boldsymbol{S}$ is symmetric. Appendix A.7 proves the proposition, the two corollaries that rest on it, and Remark A.1.

The objective acts on the surviving edit. The next corollary shows that anticipating the update amounts to applying the plain loss to the surviving edit. Write $\mathcal{L}_{\mathrm{ant}}(\delta)$ for the loss of equation 3 at the draw N, with the anticipated score in place of the composed score.

Corollary A.1 (The objective acts on the surviving edit)**.** *In the linearized model the anticipated score of a forgotten fact is $d_f(\theta^\circ) + \phi_f^\top\boldsymbol{S}\delta$, so*

$$\mathcal{L}_{\mathrm{ant}}(\delta) = \mathcal{L}_{\mathrm{plain}}(\boldsymbol{S}\delta), \qquad \mathcal{L}_{\mathrm{plain}}(\delta) = \frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}} \mathrm{relu}\left(t_f - d_f(\theta^\circ) - \phi_f^\top\delta\right)^2,$$

where $\mathcal{L}_{\mathrm{plain}}$ is the same loss on the pre-update score. With the composed score of equation 3, the loss of fact f is $\max\{\ell_f(\delta), \ell_f(\boldsymbol{S}\delta)\}$ with $\ell_f(\delta) = \mathrm{relu}(t_f - d_f(\theta^\circ) - \phi_f^\top\delta)^2$, since $\mathrm{relu}(t - \min\{a,b\})^2 = \max\{\mathrm{relu}(t-a)^2, \mathrm{relu}(t-b)^2\}$. It equals $\ell_f(\boldsymbol{S}\delta)$ whenever the update lowers the fact's score. The minimum of equation 3 then selects the anticipated score, which is the case whenever $\phi_f^\top(\mathbf{I} - \boldsymbol{S})\delta \ge 0$. The condition holds under a uniform ridge for every edit $\delta = c\,\phi_f$ and every edit $\delta = c\,\boldsymbol{S}\phi_f$ with $c \ge 0$, the second being the direction the anticipated loss itself produces. On both branches the gradient is taken through the anticipated score (Section 4), so the minimum changes a scalar weight and leaves the direction $\boldsymbol{S}^\top\phi_f$ unchanged.

In the linearized model the anticipated loss is therefore the plain forgetting loss evaluated on the part of the edit that survives the update. It is the objective of KAU under the signed residual, wherever the minimum selects the anticipated score. For an edit that combines several facts, the condition of the corollary can fail for one of them. The minimum then changes the weight of that fact and leaves its direction unchanged. The identity relates the two objectives, and it holds for the joint response at

every draw. That the anticipated loss is the plain loss at the post-update point holds by construction. The content of the identity is that, in the linearized model, the post-update edit is a fixed linear image $\boldsymbol{S}\delta$ of the edit, with the explicit spectrum of Proposition A.1.

The identity alone leaves the location of a minimizer open, because the filter is invertible for every positive ridge. Under a uniform ridge, the edit $\boldsymbol{S}^{-1}\delta'$ reaches a plain optimum δ' by overshooting each restored direction $\boldsymbol{u}_k$ by the factor $(\lambda_k+\sigma)/\sigma$. Two parts of the method rule that edit out. Every gradient of the anticipated loss lies in the span of the vectors $\boldsymbol{S}^\top\phi_f$, in which a restored direction carries the weight $\sigma/(\lambda_k+\sigma)$ (Proposition A.1 and Corollary A.1). The score-preservation term of equation 9 charges the squared residual on the retained edges, which grows with the square of an overshoot along their features. Together they select a small edit whose surviving part forgets.

Subspace methods as thresholds of the filter. A further corollary shows that confining the edit to a subspace is a hard threshold of the filter, at level zero or at a positive level.

Corollary A.2 (Two thresholds of the filter)**.** *Take a uniform ridge in the linearized model. (a) As $\sigma \to 0$, $\boldsymbol{S} \to \Pi^{\perp}$, and for the gradient-descent update of Proposition A.1, $(\mathbf{I} - \eta\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^T \to \Pi^{\perp}$ as $T \to \infty$. For every ridge, the update leaves an edit with $\boldsymbol{\Phi}\delta = 0$ untouched. (b) For a threshold $\bar{\lambda} > 0$, let $U_{\bar{\lambda}}$ be the span of the eigenvectors $\boldsymbol{u}_k$ with $\lambda_k \le \bar{\lambda}$ together with the complement of the span of the update's features. Every $\delta \in U_{\bar{\lambda}}$ satisfies $\|\boldsymbol{S}\delta\|_2 \ge \frac{\sigma}{\bar{\lambda}+\sigma}\|\delta\|_2$, and every δ in the span of the eigenvectors with $\lambda_k > \bar{\lambda}$ satisfies $\|\boldsymbol{S}\delta\|_2 \le \frac{\sigma}{\bar{\lambda}+\sigma}\|\delta\|_2$.*

Confining the edit to the complement of the update's span is null-space projection, the threshold at level zero. When the update class is the retained edges at the involved entities, it is, restricted to the involved rows, the retained-span projector that Golatkar et al. (2020b), Chen et al. (2024) and Zhou et al. (2025) use to protect utility. For an edit that moves one row alone, the constraint $\boldsymbol{\Phi}\delta = 0$ reduces to the constraint on that row, and the projector is the second limit of Corollary A.3. Read through the filter, the retained-span projector also marks the edits that continued training on retained data cannot undo to first order. The update of Section 3 also trains on edges the unlearner never saw, whose features lie outside the retained span. That update can therefore move an edit that lies in the retained-span null space. The stand-ins of the unseen channel put modelled directions of this kind into $\boldsymbol{\Phi}$ (the first limit of Corollary A.3). Confining the edit to $U_{\bar{\lambda}}$ is a hard threshold at a positive level, the form of the methods that place the edit in directions continued training restores slowly (Xiao et al., 2026; Sondej et al., 2026; Sondej & Yang, 2025). FRAG (Chen et al., 2026) applies the same idea per coordinate, on weights the retained data barely activate. Those methods read the spectrum from a proxy, the activations or gradients of the forget set or of the retained data, because they do not hold the update's data. For Xiao et al. (2026) the shared-eigenstructure assumption is what licenses the proxy. The correspondence concerns the form of the constraint. Those methods compute their spectra from the activations or gradients of deep networks and use their own losses. The subspace $U_{\bar{\lambda}}$ coincides with theirs when the proxy shares its leading eigenvectors with $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ (Appendix A.7). KAU applies the soft weights, $\sigma/(\lambda_k+\sigma)$ under a uniform ridge, with the spectrum of its draws from the modelled update class. That class holds the retained edges themselves on the known channel and stand-ins on the unseen one.

The remark below shows what the one-sided residual of equation 4 adds to the subspace of Corollary A.2.

Remark A.1 (The one-sided residual relaxes the subspace to a cone). With the one-sided residual of equation 4 in the linearized model, the response at $\theta^\circ + \delta$ is $-(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^\top \mathrm{relu}(\boldsymbol{\Phi}\delta)$, with relu applied entrywise. It vanishes exactly when $\boldsymbol{\Phi}\delta \le 0$ entrywise, so every such edit survives entirely. The set of edits the update leaves untouched grows from the subspace $\{\boldsymbol{\Phi}\delta = 0\}$ to the cone $\{\boldsymbol{\Phi}\delta \le 0\}$, which contains it.

Null-space projection forbids every change to the score of an update edge, and the response of KAU restores only the raised edges.

Composition versus a penalty on predicted recovery. The filter also says why the response is composed into the loss and is not penalized beside it. The alternative, and the form a proxy method would take, is a penalty on the predicted score recovery $\hat{\rho}_f = d_f(\theta) - d_f(\theta + \Delta\theta)$ added beside a plain forgetting loss. At $\theta = \theta^\circ$ every residual of equation 4 vanishes, so $\Delta\theta = 0$ and $\hat{\rho}_f = 0$ for every fact and every draw, and the penalty attains its minimum at the unedited model. The last

bound of Proposition A.1 gives the local form of this fact. A displacement inside the drawn span that undoes part of the forgetting, $\phi_f^{(v)\top}\delta_v < 0$, lowers $d_f(\theta)$. It moves the anticipated point by at most the fraction $\sigma_v/(\lambda^+_{\min} + \sigma_v)$ of its length. To first order it therefore lowers $\hat{\rho}_f$ by $|\phi_f^{(v)\top}\delta_v|$, up to a term of at most $\|\phi_f^{(v)}\|_2$ times that fraction of $\|\delta_v\|_2$. Such an objective can therefore reduce its penalty by undoing its own forgetting. The composed objective of equation 3 sees the fact only through the smaller of its current and anticipated scores, so it earns at most that term from the displacement.

Beyond the linearized model. The scores of the three backbones are nonlinear in the rows. Outside the linearized model, Corollary A.1 becomes a statement about the gradient at the current point θ, under three approximations. The features are frozen at θ, which KAU does too (Section 4.2). For the TransE score the slot feature is the sign pattern of the translation error, projected on the tangent of the normalized row equation 10. The sign pattern is constant while no coordinate of the translation error changes sign, so inside such a cell the feature changes only through the normalization of the row. The post-update score is evaluated to first order,

$$d_f(\theta + \Delta\theta) \;\approx\; d_f^{\mathrm{lin}}(\theta; N) := d_f(\theta) + \sum_{v\in\{h,t\}} \phi_f^{(v)\top}\Delta\boldsymbol{e}_v.$$

The one-sided residual of equation 4 is replaced by the signed residual $r_i = d_i(\theta) - d_i(\theta^\circ)$ on every drawn edge, which credits the update with restoring lowered edges as well as raised ones. The next theorem takes the per-entity response and the gradient of the anticipated score with respect to $\boldsymbol{e}_v$, through v's own response with the other endpoint's response held fixed. It shows that this gradient is the per-entity filter applied to $\phi_f^{(v)}$, the direction in which a plain forgetting loss on the pre-update score would move $\boldsymbol{e}_v$.

Theorem A.1 (Operator identity)**.** *Fix a draw N, a ridge $\sigma_v > 0$ at each involved entity v, and a point θ at which the scores of f and of the drawn edges are differentiable, which holds for almost every θ. Let $\boldsymbol{\Phi}_v$ stack the slot features at v of the drawn edges incident to v, and let $\Delta\boldsymbol{e}_v(\theta) = -\boldsymbol{A}_v r_v(\theta)$ with $\boldsymbol{A}_v = \boldsymbol{\Phi}_v^\top(\boldsymbol{\Phi}_v\boldsymbol{\Phi}_v^\top + \sigma_v\mathbf{I})^{-1}$ be the per-entity form of the response of Lemma 1. Assume that the features $\boldsymbol{\Phi}_v$ and $\phi_f^{(v)}$ are held fixed, that the post-update score is evaluated to first order, and that the residual is the signed residual $r_i = d_i(\theta) - d_i(\theta^\circ)$ on every drawn edge. Then for each endpoint v of a forgotten fact f,*

$$\frac{\partial}{\partial \boldsymbol{e}_v}\Big[d_f(\theta) + \phi_f^{(v)\top}\Delta\boldsymbol{e}_v(\theta)\Big] = \big(\mathbf{I} - \boldsymbol{\Phi}_v^\top(\boldsymbol{\Phi}_v\boldsymbol{\Phi}_v^\top + \sigma_v\mathbf{I})^{-1}\boldsymbol{\Phi}_v\big)\,\phi_f^{(v)} = \boldsymbol{P}^{\perp}_{N,v}\,\phi_f^{(v)}. \tag{13}$$

The operator $\boldsymbol{P}^{\perp}_{N,v}$ is the per-entity filter of Proposition A.1. It is symmetric. It scales the k-th right singular direction of $\boldsymbol{\Phi}_v$ by $\sigma_v/(\lambda_k + \sigma_v)$, with λ_k the nonzero eigenvalues of $\boldsymbol{\Phi}_v\boldsymbol{\Phi}_v^\top$, and it leaves the orthogonal complement of the span of the drawn features unchanged. It is therefore a shrinkage operator, and it converges to the orthogonal projector off that span as $\sigma_v \to 0$.

Proof sketch. Hold $\boldsymbol{A}_v$ fixed and let r_v vary with θ. Then $\partial\Delta\boldsymbol{e}_v/\partial\boldsymbol{e}_v = -\boldsymbol{A}_v\,\partial r_v/\partial\boldsymbol{e}_v$. Under the signed residual, $d_i(\theta^\circ)$ is a constant and $\partial d_i/\partial\boldsymbol{e}_v = \phi_i^{(v)\top}$ by definition of the slot feature, so $\partial r_v/\partial\boldsymbol{e}_v = \boldsymbol{\Phi}_v$. The chain rule gives $\phi_f^{(v)} - (\boldsymbol{A}_v\boldsymbol{\Phi}_v)^\top\phi_f^{(v)}$, which is equation 13 because the inverse is symmetric. The spectral claims follow from a singular value decomposition of $\boldsymbol{\Phi}_v$. Appendix A.5 gives the full proof, the operator under the one-sided residual, the cross-entity terms of the joint response, and the same operator at every further Gauss–Newton round (Remark A.3). □

Theorem A.1 is the local form of Corollary A.1 for the per-entity response. At any point, the anticipated loss moves each row along the plain forgetting direction passed through the per-entity filter built at that point. The draw changes at every step, so the direction the optimizer follows over a run is an average of per-draw filters. The next corollary gives that average and two of its limits. The first limit is the projector off the span of every retained edge and stand-in at the entity. The second is the projector off the span of the retained gradients alone, which we call the retained-span projector and which Section 5 runs as the retained-span projection of Table 18.

Corollary A.3 (Soft projector and its two limits)**.** *Under the hypotheses of Theorem A.1 at every draw, replace $\tilde{d}_f$ in equation 3 by the first-order score d_f^{lin}, which also drops the minimum with the current score. The expected gradient of the loss term of fact f with respect to $\boldsymbol{e}_v$ through v's own response is then*

$$-\mathbb{E}_N\big[w_N\, \boldsymbol{P}_{N,v}^{\perp}\big]\phi_f^{(v)}, \qquad w_N = 2\ \mathrm{relu}\left(t_f - d_f^{\mathrm{lin}}(\theta; N)\right) \geq 0,$$

with w_N the weight of draw N. Assume $\mathbb{E}_N[w_N] > 0$ and write $\bar{\boldsymbol{P}}_v^{\perp} = \mathbb{E}_N[w_N \boldsymbol{P}_{N,v}^{\perp}]/\mathbb{E}_N[w_N]$ for the normalized operator. As $\sigma_v \to 0$, $\bar{\boldsymbol{P}}_v^{\perp}$ removes a unit direction $\boldsymbol{u}$ in the proportion $\mathbb{E}_N[w_N \|\Pi_{N,v}\boldsymbol{u}\|_2^2]/\mathbb{E}_N[w_N]$, where $\Pi_{N,v}$ is the orthogonal projector onto the span of the drawn features at v. Two limits follow. When every candidate edge enters with probability one, the cap $k_{\max}$ is lifted and $\sigma_v \to 0$, $\bar{\boldsymbol{P}}_v^{\perp}$ becomes the orthogonal projector off the span of all retained edges and stand-ins at v. That is the step direction of a projected variant that keeps the stand-ins. When the unseen channel is removed as well, $\bar{\boldsymbol{P}}_v^{\perp}$ becomes the orthogonal projector off the span of the retained edges incident to v, which, for an edit of the row $\boldsymbol{e}_v$ alone, is the retained-span projector of Golatkar et al. (2020b), Chen et al. (2024), Saha et al. (2021), Wang et al. (2021) and Zhou et al. (2025) restricted to that row.

Averaged over draws, the filter removes each direction in proportion to how often, and how strongly, the modelled class draws it. Beyond the two limits, KAU adds the unseen channel, which puts edges the unlearner has never seen into the span. It also adds the sampled draw and the exact evaluation of the score at the shifted rows.

Scope and empirical counterparts. Proposition A.1, its two corollaries and Remark A.1 are exact in the linearized model, and Theorem A.1 is the local statement outside it. The theorem takes the per-entity response, and Appendix A.5 gives the cross-entity terms of the joint solve. Table 16 measures the joint solve against the per-entity block solve, first-order against exact evaluation, and the signed residual against KAU's, one row each. Table 17 measures the sampled draw against the expected draw and the plain composition against KAU's, one row each. Table 18 lists the retained-span projection, which approaches the second limit of Corollary A.3 as its ridge vanishes, released at KAU's forgetting depth.

The filter describes the modelled update, a ridge or gradient-descent fit of a squared residual on the drawn edges. The update of Section 3 runs Adam on the ranking loss, its sampled corruptions add features outside $\boldsymbol{\Phi}$, and its features move. Measured survival is therefore lower than the filter's. At matched forgetting depth on TransE, the update restores 0.0111 forget-MRR of KAU's edit beyond the oracle's own rise (Table 18). The same update restores 0.0158 for the retained-span projection and 0.0250 for the plain forgetting loss. The response predicts the order of recovery across facts (Appendix D.5). Corollary A.1 predicts where the gain appears. The loss stays active until the surviving score reaches t_f, so the forgetting KAU keeps is the part that the update does not restore. Section 5 controls for depth by releasing every variant at KAU's forgetting depth and by reading the relative excess recovery against depth. In Figure 3, KAU stays below GraphDPO and ZeroFisher at every depth, and the gap widens as forgetting deepens.

A.2 Proof of Lemma 1: the linearized response

Proof of Lemma 1. Write Δ for the parameter change $\Delta\theta$ of the lemma, $r = r_N(\theta) \in \mathbb{R}^{|N|}$ and $\boldsymbol{\Phi} = \boldsymbol{\Phi}_N$, whose row i is the transpose of the feature $\phi_i = \nabla_\theta d_i$ of update edge i with respect to all trainable parameters. Linearizing the scores at θ gives $r_i(\theta + \Delta) \approx r_i + \phi_i^\top \Delta$, and ridge damping adds $\frac{\sigma}{2}\|\Delta\|_2^2$, so the linearized response of the update is the minimizer of $\frac{1}{2}\|r + \boldsymbol{\Phi}\Delta\|_2^2 + \frac{\sigma}{2}\|\Delta\|_2^2$. For $\sigma > 0$ this objective is strictly convex, and its unique minimizer solves the stationarity condition $(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma\mathbf{I})\Delta = -\boldsymbol{\Phi}^\top r$. Expanding both sides shows $\boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma\mathbf{I}) = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma\mathbf{I})\boldsymbol{\Phi}^\top$, and multiplying this identity by $(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma\mathbf{I})^{-1}$ on the left and by $(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma\mathbf{I})^{-1}$ on the right gives the push-through identity $(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma\mathbf{I})^{-1}$, which is equation 5. A Gauss–Newton step on $\frac{1}{2}\sum_i r_i(\theta+\Delta)^2$ from $\Delta = 0$ minimizes the same linearized least-squares objective without damping, and Levenberg–Marquardt damping adds exactly the ridge term, so equation 5 is one damped Gauss–Newton step. The per-entity form of Section 4.2 restricts the same computation to the edges incident to v and to the block $\boldsymbol{e}_v$, which gives $\Delta\boldsymbol{e}_v = -(\boldsymbol{\Phi}_v^\top\boldsymbol{\Phi}_v + \sigma_v\mathbf{I})^{-1}\boldsymbol{\Phi}_v^\top r_v = -\boldsymbol{A}_v r_v$ by the same identity. The relation between the ridge and a finite number of descent steps is the content of the remark below. □

Remark A.2 (Ridge damping and early stopping). Consider the linearized least-squares problem $\frac{1}{2}\|r+\mathbf{\Phi}\Delta\|_2^2$ in Δ, started at $\Delta=0$. Gradient flow for time τ, gradient descent with step η for T steps under $\eta\lambda_{\max}(\mathbf{K})\le 1$, and ridge damping with parameter σ all produce a solution of the form $\Delta=-\mathbf{\Phi}^\top\sum_k g(\lambda_k)\lambda_k^{-1}\boldsymbol{q}_k\boldsymbol{q}_k^\top r$ over the nonzero eigenpairs $(\lambda_k,\boldsymbol{q}_k)$ of $\mathbf{K}=\mathbf{\Phi}\mathbf{\Phi}^\top$, with the filters

$$g_\tau(\lambda)=1-e^{-\tau\lambda},\qquad g_{\eta,T}(\lambda)=1-(1-\eta\lambda)^T,\qquad g_\sigma(\lambda)=\frac{\lambda}{\lambda+\sigma}.$$

Gradient flow solves $\dot\Delta=-\mathbf{\Phi}^\top(r+\mathbf{\Phi}\Delta)$, whose solution is $\Delta(\tau)=-(\mathbf{I}-e^{-\tau\mathbf{\Phi}^\top\mathbf{\Phi}})(\mathbf{\Phi}^\top\mathbf{\Phi})^+\mathbf{\Phi}^\top r$, and this has the stated form because $(\mathbf{\Phi}^\top\mathbf{\Phi})^+\mathbf{\Phi}^\top=\mathbf{\Phi}^\top\mathbf{K}^+$ and $\mathbf{\Phi}^\top\mathbf{\Phi}$ has the eigenvectors $\mathbf{\Phi}^\top\boldsymbol{q}_k/\sqrt{\lambda_k}$ with eigenvalues λ_k. Gradient descent replaces $e^{-\tau\lambda}$ by $(1-\eta\lambda)^T$, and ridge damping gives the filter $\lambda/(\lambda+\sigma)$ by the push-through identity of the proof above. With $\sigma=1/\tau$ and $\tau=\eta T$, and for every $\lambda\in[0,1/\eta]$, an interval that contains every eigenvalue of $\mathbf{K}$ under $\eta\lambda_{\max}(\mathbf{K})\le 1$,

$$g_\sigma(\lambda)\le g_\tau(\lambda)\le g_{\eta,T}(\lambda)\le\min\{1,\eta T\lambda\}\le 2\,g_\sigma(\lambda).$$

The first inequality is $e^{-u}\le 1/(1+u)$ at $u=\tau\lambda$. The second is $(1-\eta\lambda)^T\le e^{-\eta\lambda T}$. The third is Bernoulli's inequality $(1-\eta\lambda)^T\ge 1-T\eta\lambda$ together with $g_{\eta,T}\le 1$. The fourth is $\min\{1,u\}\le 2u/(1+u)$, which holds for $u\le 1$ because $1+u\le 2$ and for $u\ge 1$ because $1+u\le 2u$. An early-stopped linearized update is therefore bracketed, on every eigen-direction, by the ridge filter with $\sigma=1/(\eta T)$ from below and by twice that filter from above. Sharper constants, for the risk rather than for the filter, are known (Ali et al., 2019). The ridge σ_v of Section 4.2 is set relative to the feature norms at v (Appendix B.9), and every experiment runs the update as a training run rather than as a filter.

A.3 Score preservation is the known-channel residual

This subsection proves the claim of Section 4.4 that the score-preservation term of equation 9 bounds the known-channel part of the response.

Proposition A.2 (Score preservation is the known-channel residual)**.** *Let S be the sample of retained edges at the involved entities in equation 9. The score-preservation term of equation 9, $\beta\,\mathrm{mean}_{i\in S}\big(d_i(\theta)-d_i(\theta^\circ)\big)^2$, is β times the mean squared signed residual of the known channel on S, and it bounds β times the mean squared one-sided residual of equation 4 on S from above. Let r_v^{kn} be r_v with the entries of the stand-ins, the unseen-channel edges of Section 4.1, set to zero. The part of the per-entity response built from the known edges satisfies $\|\boldsymbol{A}_v r_v^{\mathrm{kn}}\|_2\le\|\boldsymbol{A}_v\|_{\mathrm{op}}\|r_v^{\mathrm{kn}}\|_2$ with $\|\boldsymbol{A}_v\|_{\mathrm{op}}\le 1/(2\sqrt{\sigma_v})$, and whenever S contains the drawn retained edges at v,*

$$\|\boldsymbol{A}_v r_v^{\mathrm{kn}}\|_2\le\sqrt{\frac{|S|}{4\sigma_v}}\,\Big(\mathrm{mean}_{i\in S}\big(d_i(\theta)-d_i(\theta^\circ)\big)^2\Big)^{1/2},$$

so minimizing the term on such a sample bounds that displacement. The joint solve of Section 4.2 obeys the same bound for the stacked known-channel displacement, with $\min_v\sigma_v$ in place of σ_v, whenever S contains every drawn retained edge.

Proof of Proposition A.2. For a retained edge $i\in S$ at an involved entity, the signed residual of the known channel is $d_i(\theta)-d_i(\theta^\circ)$ by definition, so $\beta\,\mathrm{mean}_{i\in S}(d_i(\theta)-d_i(\theta^\circ))^2$ is β times the mean squared signed residual on S, which is the first claim. Since $\mathrm{relu}(x)^2\le x^2$ for every real x, the same quantity bounds β times the mean squared one-sided residual of equation 4 on S from above, which is the second. For the third, the per-entity response decomposes by linearity as $-\boldsymbol{A}_v r_v=-\boldsymbol{A}_v r_v^{\mathrm{kn}}-\boldsymbol{A}_v r_v^{\mathrm{un}}$, where r_v^{un} keeps the stand-in entries, and the part built from the known edges satisfies $\|\boldsymbol{A}_v r_v^{\mathrm{kn}}\|_2\le\|\boldsymbol{A}_v\|_{\mathrm{op}}\|r_v^{\mathrm{kn}}\|_2$ with $\|\boldsymbol{A}_v\|_{\mathrm{op}}\le 1/(2\sqrt{\sigma_v})$ by the third fact following equation 11. By $\mathrm{relu}(x)^2\le x^2$ once more, $\|r_v^{\mathrm{kn}}\|_2^2\le\sum_i(d_i(\theta)-d_i(\theta^\circ))^2$ with the sum over the drawn retained edges at v, and whenever S contains those edges this sum is at most $\sum_{i\in S}(d_i(\theta)-d_i(\theta^\circ))^2=|S|\,\mathrm{mean}_{i\in S}(d_i(\theta)-d_i(\theta^\circ))^2$, which gives the displayed bound. The unseen part $\boldsymbol{A}_v r_v^{\mathrm{un}}$ is not controlled by the term. For the joint solve, write $\mathbf{\Sigma}=\mathrm{diag}(\sigma_v\mathbf{I})$ and $\tilde{\mathbf{\Phi}}=\mathbf{\Phi}\mathbf{\Sigma}^{-1/2}$. Then $\mathbf{\Phi}^\top\mathbf{\Phi}+\mathbf{\Sigma}=\mathbf{\Sigma}^{1/2}(\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}+\mathbf{I})\mathbf{\Sigma}^{1/2}$, so $(\mathbf{\Phi}^\top\mathbf{\Phi}+\mathbf{\Sigma})^{-1}\mathbf{\Phi}^\top=\mathbf{\Sigma}^{-1/2}(\tilde{\mathbf{\Phi}}^\top\tilde{\mathbf{\Phi}}+\mathbf{I})^{-1}\tilde{\mathbf{\Phi}}^\top$, whose operator norm is at most $\|\mathbf{\Sigma}^{-1/2}\|_{\mathrm{op}}\cdot\frac{1}{2}=1/(2\sqrt{\min_v\sigma_v})$ by the same fact with ridge one. The bound on the known-channel part of the joint displacement follows with $\min_v\sigma_v$ in place of σ_v. □

A.4 BEYOND THE LINEARIZED RESPONSE

Lemma 1 gives the linearized response of the whole run of the update, modelled as a ridge-damped least-squares fit under frozen features. This subsection bounds what one re-linearization round adds and shows that the operator identity survives re-linearization. Fix a draw N with $n = |N|$ edges, the features $\mathbf{\Phi}$ with rows $\phi_i^\top$ at θ, the kernel $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top$, a ridge $\sigma > 0$, and the residual $r = r(\theta)$ with $r_i(\theta + \Delta) = d_i(\theta + \Delta) - y_i$ as in the lemma. The ridge-damped update minimizes

$$F(\Delta) = \tfrac{1}{2} \sum_{i \in N} r_i(\theta + \Delta)^2 + \tfrac{\sigma}{2} \|\Delta\|_2^2.$$

Its Gauss–Newton iteration starts at $\Delta_0 = 0$ and sets

$$\Delta_{k+1} = \Delta_k - (\mathbf{\Phi}_k^\top \mathbf{\Phi}_k + \sigma \mathbf{I})^{-1} \big(\mathbf{\Phi}_k^\top r(\theta + \Delta_k) + \sigma \Delta_k \big),$$

with $\mathbf{\Phi}_k$ the features at $\theta + \Delta_k$. The first iterate is $\Delta_1 = -(\mathbf{\Phi}^\top \mathbf{\Phi} + \sigma \mathbf{I})^{-1} \mathbf{\Phi}^\top r = -\mathbf{\Phi}^\top (\mathbf{K} + \sigma \mathbf{I})^{-1} r$, which is the response of Lemma 1. Write $\alpha = (\mathbf{K} + \sigma \mathbf{I})^{-1} r$ for the dual coefficients, so that $\Delta_1 = -\mathbf{\Phi}^\top \alpha$, and write $r^{\mathrm{lin}}(\Delta) = r + \mathbf{\Phi}\Delta$ for the linearized residual, so that $r^{\mathrm{lin}}(\Delta_1) = r - \mathbf{K}\alpha = \sigma\alpha$.

Lemma A.1 (One-round error)**.** *Assume that along the segment $\{\theta + t\Delta_1 : t \in [0, 1]\}$ every score d_i, $i \in N$, is differentiable, with features $\phi_i = \nabla_\theta d_i$ satisfying $\|\phi_i(\theta + t\Delta_1) - \phi_i(\theta + s\Delta_1)\|_2 \le L\,|t - s|\,\|\Delta_1\|_2$ for all $s, t \in [0, 1]$. Then (i) $|r_i(\theta + \Delta_1) - r_i^{\mathrm{lin}}(\Delta_1)| \le \frac{L}{2} \|\Delta_1\|_2^2$ for every $i \in N$, (ii) $\|\mathbf{\Phi}_1 - \mathbf{\Phi}\|_{\mathrm{op}} \le \sqrt{n}\, L\, \|\Delta_1\|_2$, and (iii)*

$$\|\Delta_2 - \Delta_1\|_2 \le \frac{\sqrt{n}\, L}{4\sqrt{\sigma}} \|\Delta_1\|_2^2 + \sqrt{n}\, L\, \|\Delta_1\|_2 \|\alpha\|_2 \le \sqrt{n}\, L\, \|\mathbf{\Phi}\|_{\mathrm{op}} \Big(\frac{\|\mathbf{\Phi}\|_{\mathrm{op}}}{4\sqrt{\sigma}} + 1 \Big) \|\alpha\|_2^2.$$

Proof. (i) By the fundamental theorem of calculus along the segment,

$$r_i(\theta + \Delta_1) - r_i(\theta) - \phi_i^\top \Delta_1 = \int_0^1 \big(\phi_i(\theta + t\Delta_1) - \phi_i(\theta) \big)^\top \Delta_1 \, dt,$$

and the Lipschitz hypothesis bounds the integrand by $L\,t\,\|\Delta_1\|_2^2$, whose integral is $\frac{L}{2}\|\Delta_1\|_2^2$. (ii) Each row of $\mathbf{\Phi}_1 - \mathbf{\Phi}$ is $\phi_i(\theta + \Delta_1) - \phi_i(\theta)$, of norm at most $L\|\Delta_1\|_2$, so the Frobenius norm, which dominates the operator norm, is at most $\sqrt{n}\, L \|\Delta_1\|_2$. (iii) The gradient of F at Δ_1 is $\nabla F(\Delta_1) = \mathbf{\Phi}_1^\top r(\theta + \Delta_1) + \sigma\Delta_1$. Since Δ_1 minimizes the linearized objective $\frac{1}{2}\|r^{\mathrm{lin}}(\Delta)\|_2^2 + \frac{\sigma}{2}\|\Delta\|_2^2$, it satisfies $\mathbf{\Phi}^\top r^{\mathrm{lin}}(\Delta_1) + \sigma\Delta_1 = 0$. Subtracting,

$$\nabla F(\Delta_1) = \mathbf{\Phi}_1^\top e + (\mathbf{\Phi}_1 - \mathbf{\Phi})^\top r^{\mathrm{lin}}(\Delta_1) = \mathbf{\Phi}_1^\top e + \sigma(\mathbf{\Phi}_1 - \mathbf{\Phi})^\top \alpha,$$

where $e = r(\theta + \Delta_1) - r^{\mathrm{lin}}(\Delta_1)$ has $\|e\|_2 \le \sqrt{n}\, \frac{L}{2} \|\Delta_1\|_2^2$ by (i). By definition $\Delta_2 - \Delta_1 = -(\mathbf{\Phi}_1^\top \mathbf{\Phi}_1 + \sigma \mathbf{I})^{-1} \nabla F(\Delta_1)$. The derivation of equation 11 holds for any matrix and any $\sigma > 0$. Applied to $\mathbf{\Phi}_1$, the push-through identity of the proof of Lemma 1 and the third fact following equation 11 give $\|(\mathbf{\Phi}_1^\top \mathbf{\Phi}_1 + \sigma \mathbf{I})^{-1} \mathbf{\Phi}_1^\top\|_{\mathrm{op}} \le 1/(2\sqrt{\sigma})$, and $\|(\mathbf{\Phi}_1^\top \mathbf{\Phi}_1 + \sigma \mathbf{I})^{-1}\|_{\mathrm{op}} \le 1/\sigma$. Hence $\|\Delta_2 - \Delta_1\|_2 \le \|e\|_2/(2\sqrt{\sigma}) + \|\mathbf{\Phi}_1 - \mathbf{\Phi}\|_{\mathrm{op}} \|\alpha\|_2$, which is the first inequality of (iii) after (ii). The second follows from $\|\Delta_1\|_2 = \|\mathbf{\Phi}^\top \alpha\|_2 \le \|\mathbf{\Phi}\|_{\mathrm{op}} \|\alpha\|_2$. □

Both terms of (iii) are quadratic in the dual coefficients α, hence of second order in the residual the edit left on the drawn edges. The first term is the curvature of the scores along the displacement. The second is the drift of the features weighted by $\sigma\alpha$, the residual the linearized fit leaves. Each further round repeats the step from the new point and costs one more solve on the same draw. Appendix B.11 records that a second round on the same draw improved no reported quantity, which is why the objective uses one round. Appendix D.5 reports that the one-round response tracks the realized per-fact score recovery with a Pearson correlation of 0.53 and a Spearman correlation of 0.21, and it predicts the ranking of the facts and not the level of the change. The lemma bounds the correction of one further round. It does not bound the level gap of an update of many epochs, and the objective relies on the ranking while the target sets the level (Section 4.3).

For TransE and TransH the score is piecewise linear in the translation error of the triple and its slot features jump where a coordinate of that vector changes sign. Along a segment that crosses no sign change the sign vector $\boldsymbol{s}$ is constant and the feature $(\mathbf{I} - \bar{\boldsymbol{e}}_v \bar{\boldsymbol{e}}_v^\top)\boldsymbol{s}/\|\boldsymbol{e}_v\|_2$ is smooth in the row. The Lipschitz hypothesis then holds with a constant set by the derivative of the normalization

Jacobian, of order $\sqrt{m}/\|\boldsymbol{e}_v\|_2^2$ on a region where the row norms are bounded below. Across a sign change the hypothesis fails. By Lemma A.5 the gap of item (i) is then the normalization's second-order term plus the crossed-coordinate excess $\varkappa(\boldsymbol{D})$ of Appendix A.13, which is non-negative and, in a coordinate with $|x_j|$ small against $|D_j|$, of first order in the displacement rather than second. Appendix D.5 measures its effect on the ranking.

Remark A.3 (The operator identity round by round). Hold the features fixed at every round and use the signed residual, as in Theorem A.1, so that $\boldsymbol{\Phi}_k$ enters through the Jacobian of the residual at $\theta + \Delta_k$ but is not itself differentiated. Write $\boldsymbol{G}_k = (\boldsymbol{\Phi}_k^\top \boldsymbol{\Phi}_k + \sigma \mathbf{I})^{-1}$. Differentiating the iteration with respect to θ gives

$$\frac{\partial \Delta_{k+1}}{\partial \theta} = \frac{\partial \Delta_k}{\partial \theta} - \boldsymbol{G}_k \Big(\boldsymbol{\Phi}_k^\top \boldsymbol{\Phi}_k \Big(\mathbf{I} + \frac{\partial \Delta_k}{\partial \theta} \Big) + \sigma \, \frac{\partial \Delta_k}{\partial \theta} \Big) = -\boldsymbol{G}_k \boldsymbol{\Phi}_k^\top \boldsymbol{\Phi}_k,$$

because $\boldsymbol{G}_k(\boldsymbol{\Phi}_k^\top \boldsymbol{\Phi}_k + \sigma \mathbf{I}) = \mathbf{I}$. The derivative of the anticipated point $\theta + \Delta_{k+1}$ is therefore $\mathbf{I} - \boldsymbol{G}_k \boldsymbol{\Phi}_k^\top \boldsymbol{\Phi}_k = \mathbf{I} - \boldsymbol{\Phi}_k^\top (\boldsymbol{\Phi}_k \boldsymbol{\Phi}_k^\top + \sigma \mathbf{I})^{-1} \boldsymbol{\Phi}_k$ by the push-through identity of the proof of Lemma 1, a symmetric matrix, hence the shrinkage operator of equation 13 built from the features at the k-th anticipated point. Earlier rounds enter only through the point at which those features are taken. This operator is the implicit Jacobian of a proximal model of adaptation with respect to its centre (Rajeswaran et al., 2019). Theorem A.1 therefore holds for every round with $\boldsymbol{\Phi}_k$ in place of $\boldsymbol{\Phi}$, and so does Proposition A.1 when the residual at the k-th anticipated point changes by $\boldsymbol{\Phi}_k$ times the motion of that point, which holds to first order in the displacement. The per-entity form of the theorem is the block-diagonal case, and the computation is unchanged with $\boldsymbol{\Sigma}$ in place of $\sigma \mathbf{I}$.

A.5 Proof of Theorem A.1: the operator identity

Proof of Theorem A.1. Let u be the other endpoint of f and write

$$d_f^{\mathrm{lin}}(\theta) = d_f(\theta) + \phi_f^{(v)\top} \Delta \boldsymbol{e}_v(\theta) + \phi_f^{(u)\top} \Delta \boldsymbol{e}_u(\theta)$$

for the anticipated score evaluated to first order, with $\phi_f^{(v)}$, $\phi_f^{(u)}$, $\boldsymbol{A}_v$ and $\boldsymbol{A}_u$ held fixed. Its derivative with respect to $\boldsymbol{e}_v$ has three terms. The first is $\partial d_f / \partial \boldsymbol{e}_v = \phi_f^{(v)}$, by definition of the slot feature. The second is $(\partial \Delta \boldsymbol{e}_v / \partial \boldsymbol{e}_v)^\top \phi_f^{(v)}$. The third is the cross-entity term $C_v = (\partial \Delta \boldsymbol{e}_u / \partial \boldsymbol{e}_v)^\top \phi_f^{(u)}$, treated in part (c) below. The theorem concerns the first two, which are the derivative of $d_f + \phi_f^{(v)\top} \Delta \boldsymbol{e}_v$.

(a) The signed residual. Since $\Delta \boldsymbol{e}_v = -\boldsymbol{A}_v r_v(\theta)$ with $\boldsymbol{A}_v$ fixed and r_v varying with θ, $\partial \Delta \boldsymbol{e}_v / \partial \boldsymbol{e}_v = -\boldsymbol{A}_v \, \partial r_v / \partial \boldsymbol{e}_v$. Under the signed residual $r_i = d_i(\theta) - d_i(\theta^\circ)$ the subtracted term is a constant, so $\partial r_i / \partial \boldsymbol{e}_v = \phi_i^{(v)\top}$ for every $i \in P_v(N)$, that is, $\partial r_v / \partial \boldsymbol{e}_v = \boldsymbol{\Phi}_v$. The matrix that builds the response operator therefore enters a second time through the residual. Hence $\partial \Delta \boldsymbol{e}_v / \partial \boldsymbol{e}_v = -\boldsymbol{A}_v \boldsymbol{\Phi}_v$ and

$$\phi_f^{(v)} + (-\boldsymbol{A}_v \boldsymbol{\Phi}_v)^\top \phi_f^{(v)} = \phi_f^{(v)} - \boldsymbol{\Phi}_v^\top (\mathbf{K}_v + \sigma_v \mathbf{I})^{-1} \boldsymbol{\Phi}_v \phi_f^{(v)} = \boldsymbol{P}_{N,v}^{\perp} \phi_f^{(v)},$$

where the transpose was taken using the symmetry of $(\mathbf{K}_v + \sigma_v \mathbf{I})^{-1}$. This is equation 13. The spectral claims of the statement are the first two facts read off equation 11. For the loss of draw N, $\mathrm{relu}(t_f - d_f^{\mathrm{lin}})^2$, the chain rule gives $\partial \, \mathrm{relu}(t_f - d_f^{\mathrm{lin}})^2 / \partial \boldsymbol{e}_v = -2 \, \mathrm{relu}(t_f - d_f^{\mathrm{lin}}) \, (\boldsymbol{P}_{N,v}^{\perp} \phi_f^{(v)} + C_v)$.

(b) The one-sided residual. Under $r_i = \mathrm{relu}(d_i(\theta) - d_i(\theta^\circ))$, almost everywhere $\partial r_i / \partial \boldsymbol{e}_v = \mathbf{1}[r_i > 0] \, \phi_i^{(v)\top}$. Let $\mathcal{A} \subseteq P_v(N)$ be the edges with positive residual and $\boldsymbol{D}_{\mathcal{A}}$ the diagonal matrix with entry one on $\mathcal{A}$ and zero elsewhere. Then $\partial \Delta \boldsymbol{e}_v / \partial \boldsymbol{e}_v = -\boldsymbol{A}_v \boldsymbol{D}_{\mathcal{A}} \boldsymbol{\Phi}_v$, whose transpose is $-\boldsymbol{\Phi}_v^\top \boldsymbol{D}_{\mathcal{A}} (\mathbf{K}_v + \sigma_v \mathbf{I})^{-1} \boldsymbol{\Phi}_v$, and the operator of equation 13 becomes

$$\boldsymbol{M}_{N,v} = \mathbf{I} - \boldsymbol{\Phi}_v^\top \boldsymbol{D}_{\mathcal{A}} (\mathbf{K}_v + \sigma_v \mathbf{I})^{-1} \boldsymbol{\Phi}_v.$$

Only the edges with positive residual enter through the residual, whereas every drawn edge still enters through the kernel. For $\sigma_v > 0$ and $\mathcal{A} \neq P_v(N)$ the operator $\boldsymbol{M}_{N,v}$ is in general not symmetric and its norm can exceed one, so the shrinkage reading of Theorem A.1 is specific to the signed residual. When every drawn edge has positive residual, $\boldsymbol{M}_{N,v} = \boldsymbol{P}_{N,v}^{\perp}$. In the limit $\sigma_v \to 0$ with $\mathbf{K}_v$ invertible, write $\boldsymbol{\Phi}_{\mathcal{A}}$ for the rows of $\boldsymbol{\Phi}_v$ in $\mathcal{A}$ and $\mathbf{K}_{:,\mathcal{A}} = \boldsymbol{\Phi}_v \boldsymbol{\Phi}_{\mathcal{A}}^\top$ for the corresponding

columns of $\mathbf{K}_v$. Since $\mathbf{K}_v^{-1}\mathbf{K}_{:,\mathcal{A}}$ is the matrix that selects the columns in $\mathcal{A}$ and $\boldsymbol{D}_\mathcal{A}$ leaves those columns unchanged, $\boldsymbol{M}_{N,v}\boldsymbol{\Phi}_\mathcal{A}^\top = \boldsymbol{\Phi}_\mathcal{A}^\top - \boldsymbol{\Phi}_v^\top \boldsymbol{D}_\mathcal{A}\mathbf{K}_v^{-1}\mathbf{K}_{:,\mathcal{A}} = \boldsymbol{\Phi}_\mathcal{A}^\top - \boldsymbol{\Phi}_\mathcal{A}^\top = 0$, and

$$\boldsymbol{M}_{N,v}^2 = \mathbf{I} - 2\boldsymbol{\Phi}_v^\top \boldsymbol{D}_\mathcal{A}\mathbf{K}_v^{-1}\boldsymbol{\Phi}_v + \boldsymbol{\Phi}_v^\top \boldsymbol{D}_\mathcal{A}\mathbf{K}_v^{-1}\mathbf{K}_v\boldsymbol{D}_\mathcal{A}\mathbf{K}_v^{-1}\boldsymbol{\Phi}_v = \mathbf{I} - \boldsymbol{\Phi}_v^\top \boldsymbol{D}_\mathcal{A}\mathbf{K}_v^{-1}\boldsymbol{\Phi}_v = \boldsymbol{M}_{N,v},$$

using $\boldsymbol{D}_\mathcal{A}^2 = \boldsymbol{D}_\mathcal{A}$. The limiting operator is thus an oblique projector that annihilates the active features. The set $\mathcal{A}$ depends on θ, so the identity holds on each region of parameter space on which the active set is constant.

(c) Cross-entity terms and the joint response. Under the per-entity response, $\partial\Delta\boldsymbol{e}_u/\partial\boldsymbol{e}_v = -\boldsymbol{A}_u\,\partial r_u/\partial\boldsymbol{e}_v$, and for $i \in P_u(N)$ the derivative $\partial r_i/\partial\boldsymbol{e}_v$ is nonzero only if v is the other endpoint of i. Hence $C_v = 0$ unless some drawn edge at u has v as its other endpoint, in which case $C_v = -\boldsymbol{\Phi}_{u|v}^\top \boldsymbol{A}_u^\top \phi_f^{(u)}$, where $\boldsymbol{\Phi}_{u|v} \in \mathbb{R}^{n_u \times m}$ has the row $\phi_i^{(v)\top}$ for $i \in P_u(N) \cap P_v(N)$ and zero rows elsewhere. Under the joint response of Section 4.2, stack the involved rows as $\Delta \in \mathbb{R}^{m|\mathcal{V}_\mathcal{F}|}$, let $\boldsymbol{\Phi} \in \mathbb{R}^{|N|\times m|\mathcal{V}_\mathcal{F}|}$ be the full Jacobian of the drawn edges, whose row i carries $\phi_i^{(h)\top}$ in the block of its head and $\phi_i^{(t)\top}$ in the block of its tail, and let $\phi_f \in \mathbb{R}^{m|\mathcal{V}_\mathcal{F}|}$ be the stacked feature of f, nonzero in the blocks of h and t. Then $\Delta = -(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^\top r$, the signed residual has Jacobian $\boldsymbol{\Phi}$ with respect to the stacked involved rows, and the gradient of $d_f + \phi_f^\top\Delta$ with respect to the stacked rows is

$$\phi_f - \boldsymbol{\Phi}^\top\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\phi_f = \boldsymbol{\Sigma}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma})^{-1}\phi_f.$$

For a uniform ridge $\boldsymbol{\Sigma} = \sigma\mathbf{I}$ the push-through identity turns this into $\big(\mathbf{I} - \boldsymbol{\Phi}^\top(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi}\big)\phi_f$, the operator of equation 13 with the full Jacobian in place of the per-entity one, and for a general diagonal $\boldsymbol{\Sigma}$ it is the similarity transform $\boldsymbol{\Sigma}^{1/2}\big(\mathbf{I} - \tilde{\boldsymbol{\Phi}}^\top(\tilde{\boldsymbol{\Phi}}\tilde{\boldsymbol{\Phi}}^\top + \mathbf{I})^{-1}\tilde{\boldsymbol{\Phi}}\big)\boldsymbol{\Sigma}^{-1/2}\phi_f$ with $\tilde{\boldsymbol{\Phi}} = \boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1/2}$. The block of this operator at (v, v), applied to $\phi_f^{(v)}$, is the part of the derivative with respect to $\boldsymbol{e}_v$ that passes through v's own displacement, and its block at (v, u), applied to $\phi_f^{(u)}$, is the cross-entity term. By Lemma A.4, $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma}$ is block diagonal when no drawn edge has two involved endpoints, and in that case the diagonal block reduces to $\boldsymbol{P}_{N,v}^\perp$ and every cross-entity term vanishes. Otherwise the inverse of the block-sparse positive definite matrix $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \boldsymbol{\Sigma}$ is generically dense on each connected component of the graph formed by the drawn edges, so the cross-entity terms generically couple every pair of involved entities joined by a path of drawn edges. No per-entity projector contains these terms. □

A.6 Proof of Corollary A.3: the soft projector and its two limits

Proof of Corollary A.3. The draw. At entity v the candidate pool $\mathcal{C}_v$ consists of the retained edges incident to v, each included independently with probability ρ_{kn}, and the stand-ins of v, each included independently with probability $p_{\mathrm{un}}(v) = \min\{1, \rho_{\mathrm{un}}\deg(v)/32\}$. When more than $k_{\max}$ edges are included, a uniformly random subset of size $k_{\max}$ is kept. The result is a probability distribution over subsets $P_v(N) \subseteq \mathcal{C}_v$, and every expectation below is a finite sum.

The soft projector. By Theorem A.1, the gradient of the loss of draw N through v's own response is $-w_N\boldsymbol{P}_{N,v}^\perp\phi_f^{(v)}$ with $w_N = 2\,\mathrm{relu}(t_f - d_f^{\mathrm{lin}}(\theta; N)) \geq 0$, and its expectation over the draw is $-\mathbb{E}_N[w_N\boldsymbol{P}_{N,v}^\perp]\phi_f^{(v)}$, a non-negative combination of symmetric matrices with spectra in $(0, 1]$. When $\mathbb{E}_N[w_N] > 0$, the normalized operator $\bar{\boldsymbol{P}}_v^\perp = \mathbb{E}_N[w_N\boldsymbol{P}_{N,v}^\perp]/\mathbb{E}_N[w_N]$ is symmetric with spectrum in $(0, 1]$. For a unit vector $\boldsymbol{u}$, by equation 11,

$$\boldsymbol{u}^\top(\mathbf{I} - \boldsymbol{P}_{N,v}^\perp)\boldsymbol{u} = \sum_{k\leq\varrho}\frac{\lambda_k}{\lambda_k + \sigma_v}\langle\boldsymbol{u}, \boldsymbol{u}_k\rangle^2 \in \Big[\frac{\lambda_{\min}^+(\mathbf{K}_v)}{\lambda_{\min}^+(\mathbf{K}_v) + \sigma_v}\,\|\Pi_{N,v}\boldsymbol{u}\|_2^2,\ \|\Pi_{N,v}\boldsymbol{u}\|_2^2\Big],$$

so the fraction of $\boldsymbol{u}$ that $\bar{\boldsymbol{P}}_v^\perp$ removes, $\boldsymbol{u}^\top(\mathbf{I} - \bar{\boldsymbol{P}}_v^\perp)\boldsymbol{u}$, lies between the w-weighted expectations of the two ends and equals $\mathbb{E}_N[w_N\|\Pi_{N,v}\boldsymbol{u}\|_2^2]/\mathbb{E}_N[w_N]$ as $\sigma_v \to 0$. This is the weighted expected squared fraction of $\boldsymbol{u}$ that falls inside the drawn span. With equal weights the operator is $\mathbb{E}_N[\boldsymbol{P}_{N,v}^\perp]$.

Limit 1, the update span. If $\rho_{\mathrm{kn}} \to 1$, $p_{\mathrm{un}}(v) \to 1$ and the cap is lifted, then $P_v(N) = \mathcal{C}_v$ with probability one, the expectation over draws reduces to the single term $\boldsymbol{P}_{\mathcal{C}_v,v}^\perp$, and $\sigma_v \to 0$ gives $\Pi_{\mathcal{C}_v,v}^\perp$ by equation 11, the orthogonal projector off the span of all retained edges and stand-ins at

v. A first-order projected variant that keeps the stand-ins applies this projector to the gradient of a plain forgetting loss on the pre-update score, $-2\,\mathrm{relu}(t_f - d_f(\theta))\,\Pi^{\perp}_{\mathcal{C}_v,v}\phi_f^{(v)}$, and the limit of the corollary is $-w\,\Pi^{\perp}_{\mathcal{C}_v,v}\phi_f^{(v)}$ with w evaluated on the linearized anticipated score. The two steps share their direction and differ in the scalar that weights it.

Limit 2, the retained span. If $\rho_{\mathrm{kn}} \to 1$, $\rho_{\mathrm{un}} \to 0$ and the cap is lifted, every stand-in has inclusion probability tending to zero and every retained edge has inclusion probability tending to one, so $P_v(N)$ converges in distribution to the set of retained edges incident to v, and $\sigma_v \to 0$ gives the orthogonal projector off their span. Retained-span methods project the unlearner's edit off the span of the per-example gradients of a retained loss. For a loss that depends on θ through the retained scores, the gradient of the term of retained edge i with respect to $\boldsymbol{e}_v$ is $c_i\phi_i^{(v)}$ with a scalar c_i, so the span of the per-edge retained gradients at v is contained in the span of the retained slot features at v, with equality when every $c_i \neq 0$. A ranking loss with sampled negatives adds the features of corrupted triples at v, which this identification leaves out. For an edit confined to the row $\boldsymbol{e}_v$, the constraint of those methods reads $\phi_i^{(v)\top}\delta_v = 0$ over the retained edges at v, which is this projector, and that is the sense of the restriction to the row $\boldsymbol{e}_v$ in the statement. Their projector over several rows at once is the joint one and is not the stack of per-entity projectors (Lemma A.4). Those methods apply the projector to the unlearner's edit to protect the retained set. Here the same operator arises as the direction of the anticipated loss.

The $m-1$ condition. By equation 10, on the two normalized backbones every slot feature at v, including $\phi_f^{(v)}$, lies in $\bar{\boldsymbol{e}}_v^{\perp}$, of dimension $m-1$. If the drawn edges at v satisfy $\mathrm{rank}\,\boldsymbol{\Phi}_v = m-1$, then $\mathrm{range}(\boldsymbol{\Phi}_v^{\top}) = \bar{\boldsymbol{e}}_v^{\perp} \ni \phi_f^{(v)}$ and $\Pi^{\perp}_{N,v}\phi_f^{(v)} = 0$, while for $\sigma_v > 0$ the bound $\|\boldsymbol{P}^{\perp}_{N,v}\phi_f^{(v)}\|_2 \leq \sigma_v/(\lambda^{+}_{\min}(\mathbf{K}_v)+\sigma_v)\,\|\phi_f^{(v)}\|_2$ holds by the computation in the proof of Proposition A.1. Rank $m-1$ requires at least $m-1$ edges, so an entity whose degree in the update class is below $m-1$ never reaches this regime. Under the cap, $\mathrm{rank}\,\boldsymbol{\Phi}_v \leq k_{\max} = 96 < m-1 = 199$ for the 200-wide rows of TransE and TransH, and $k_{\max} < m = 400$ for the unnormalized 400-wide rows of RotatE, where the regime requires rank m. Each draw therefore leaves at least $m-1-k_{\max}$ tangent directions at v untouched ($m-k_{\max}$ directions on RotatE), and no draw of the sampled objective enters the regime. Both limits, which lift the cap, and the update itself on an entity of high degree, whose edges are not capped, can enter it. □

A.7 Proofs for the survival filter

This subsection proves the statements of Appendix A.1 that are made in the linearized model, namely the joint form of Proposition A.1, Corollaries A.1 and A.2, and Remark A.1. Throughout, $\boldsymbol{\Phi} \in \mathbb{R}^{|N|\times m|\mathcal{V}_{\mathcal{F}}|}$ is the full Jacobian of the drawn edges over the stacked involved rows, as in part (c) of the proof of Theorem A.1, frozen at θ°. We write $\boldsymbol{C} = \boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}$, $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\top}$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_v\mathbf{I})$ with every $\sigma_v > 0$, so that $\boldsymbol{C}+\boldsymbol{\Sigma}$ is positive definite. The edit is $\delta = \theta - \theta^{\circ} \in \mathbb{R}^{m|\mathcal{V}_{\mathcal{F}}|}$, the stacked feature of a forgotten fact is $\phi_f \in \mathbb{R}^{m|\mathcal{V}_{\mathcal{F}}|}$, the residual is $r(\theta) = \boldsymbol{\Phi}(\theta-\theta^{\circ})$, and the response is $\Delta\theta(\theta) = -(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^{\top}r(\theta)$, the solution of equation 6.

Proof of Proposition A.1, joint form. The identity. The residual is affine, $r(\theta+\delta) = r(\theta) + \boldsymbol{\Phi}\delta$, so

$$\Delta\theta(\theta+\delta) = -(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^{\top}r(\theta) - (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\delta = \Delta\theta(\theta) - (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{C}\,\delta.$$

Adding $\theta+\delta$ to both sides gives $\theta+\delta+\Delta\theta(\theta+\delta) = \theta+\Delta\theta(\theta) + \big(\mathbf{I} - (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{C}\big)\delta$. Since $(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}(\boldsymbol{C}+\boldsymbol{\Sigma}) = \mathbf{I}$,

$$\mathbf{I} - (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{C} = (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\big((\boldsymbol{C}+\boldsymbol{\Sigma}) - \boldsymbol{C}\big) = (\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma} = \boldsymbol{S},$$

which is the identity at every θ. At $\theta = \theta^{\circ}$ the residual is $r(\theta^{\circ}) = 0$, hence $\Delta\theta(\theta^{\circ}) = 0$, and the identity reads $\theta^{\circ}+\delta+\Delta\theta(\theta^{\circ}+\delta) = \theta^{\circ} + \boldsymbol{S}\delta$, which is equation 12.

The edits left untouched. By the last display, $\boldsymbol{S}\delta = \delta$ holds if and only if $(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{C}\delta = 0$, that is, if and only if $\boldsymbol{C}\delta = 0$, because $(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}$ is invertible. If $\boldsymbol{\Phi}\delta = 0$ then $\boldsymbol{C}\delta = \boldsymbol{\Phi}^{\top}\boldsymbol{\Phi}\delta = 0$. Conversely, $\boldsymbol{C}\delta = 0$ gives $\|\boldsymbol{\Phi}\delta\|_2^2 = \delta^{\top}\boldsymbol{C}\delta = 0$. Hence $\boldsymbol{S}\delta = \delta$ if and only if $\boldsymbol{\Phi}\delta = 0$.

The uniform ridge. Let $\boldsymbol{\Sigma} = \sigma\mathbf{I}$. The push-through identity of the proof of Lemma 1 gives $(\boldsymbol{C} + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}^\top(\mathbf{K} + \sigma\mathbf{I})^{-1}$, so

$$\boldsymbol{S} = \mathbf{I} - (\boldsymbol{C} + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = \mathbf{I} - \boldsymbol{\Phi}^\top(\mathbf{K} + \sigma\mathbf{I})^{-1}\boldsymbol{\Phi}.$$

The derivation of equation 11 uses only a singular value decomposition of the matrix and $\sigma > 0$, so it applies to the stacked $\boldsymbol{\Phi}$. With $\boldsymbol{\Phi} = \sum_{k\le\varrho}\sqrt{\lambda_k}\,\boldsymbol{q}_k\boldsymbol{u}_k^\top$, it gives

$$\boldsymbol{S} = \sum_{k\le\varrho}\frac{\sigma}{\lambda_k + \sigma}\,\boldsymbol{u}_k\boldsymbol{u}_k^\top + \Pi^\perp, \qquad \Pi^\perp = \mathbf{I} - \sum_{k\le\varrho}\boldsymbol{u}_k\boldsymbol{u}_k^\top. \tag{14}$$

The vectors $\boldsymbol{u}_k$ are orthonormal eigenvectors of $\boldsymbol{C} = \sum_{k\le\varrho}\lambda_k\boldsymbol{u}_k\boldsymbol{u}_k^\top$ with the eigenvalues $\lambda_k > 0$, which are also the nonzero eigenvalues of $\mathbf{K}$. They span $\mathrm{range}(\boldsymbol{\Phi}^\top)$, the span of the update's features, and $\Pi^\perp$ is the orthogonal projector off that span. Expanding $\delta = \sum_{k\le\varrho}c_k\boldsymbol{u}_k + \Pi^\perp\delta$ with $c_k = \langle\boldsymbol{u}_k, \delta\rangle$ gives $\boldsymbol{S}\delta = \sum_{k\le\varrho}\frac{\sigma}{\lambda_k+\sigma}c_k\boldsymbol{u}_k + \Pi^\perp\delta$. The component along $\boldsymbol{u}_k$ is multiplied by $\sigma/(\lambda_k + \sigma)$ and the component off the span is unchanged.

Gradient descent. Let the update start at the edited model $\theta^\circ + \delta$ and take T gradient steps of size η on the linearized squared residual. Write ϵ_t for the change after t steps, with $\epsilon_0 = 0$. The loss is $\frac{1}{2}\|r(\theta^\circ + \delta + \epsilon)\|_2^2 = \frac{1}{2}\|\boldsymbol{\Phi}(\delta + \epsilon)\|_2^2$, and its gradient in ϵ is $\boldsymbol{C}(\delta + \epsilon)$. The recursion is therefore $\epsilon_{t+1} = \epsilon_t - \eta\,\boldsymbol{C}(\delta + \epsilon_t)$, which reads $\delta + \epsilon_{t+1} = (\mathbf{I} - \eta\boldsymbol{C})(\delta + \epsilon_t)$. Induction on t from $\delta + \epsilon_0 = \delta$ gives $\delta + \epsilon_T = (\mathbf{I} - \eta\boldsymbol{C})^T\delta$, the surviving edit. In the eigenbasis of $\boldsymbol{C}$,

$$(\mathbf{I} - \eta\boldsymbol{C})^T = \sum_{k\le\varrho}(1 - \eta\lambda_k)^T\boldsymbol{u}_k\boldsymbol{u}_k^\top + \Pi^\perp,$$

so the surviving fraction along $\boldsymbol{u}_k$ is $(1 - \eta\lambda_k)^T$, and it lies in $[0, 1)$ because $0 < \eta\lambda_k \le 1$ under $\eta \le 1/\lambda_{\max}(\mathbf{K})$. The first two inequalities of the chain in Remark A.2 state $g_\sigma(\lambda) \le g_{\eta,T}(\lambda)$ on $[0, 1/\eta]$ at $\sigma = 1/(\eta T)$, that is, $\lambda/(\lambda + \sigma) \le 1 - (1 - \eta\lambda)^T$. Rearranged at $\lambda = \lambda_k$ this is $(1 - \eta\lambda_k)^T \le \sigma/(\lambda_k + \sigma)$. The last inequality of the same chain, $g_{\eta,T} \le 2g_\sigma$, gives the lower bound $(1 - \eta\lambda_k)^T \ge (\sigma - \lambda_k)/(\lambda_k + \sigma)$.

The per-entity form. Restrict the computation of the first step to an edit that moves the row $\boldsymbol{e}_v$ alone, with $\boldsymbol{\Phi}_v$, σ_v and r_v in place of $\boldsymbol{\Phi}$, $\boldsymbol{\Sigma}$ and r. It gives the same identities with $\mathbf{I} - \boldsymbol{A}_v\boldsymbol{\Phi}_v = \boldsymbol{P}_{N,v}^\perp$ of equation 11 in place of $\boldsymbol{S}$. Appendix A.8 carries out this case and proves the bound for a displacement inside the drawn span. □

Proof of Corollary A.1. The identity. In the linearized model the score of f is $d_f(\theta^\circ + \delta) = d_f(\theta^\circ) + \phi_f^\top\delta$ for every δ. The anticipated score is the score at the anticipated point, and by equation 12 that point is $\theta^\circ + \boldsymbol{S}\delta$. Hence

$$d_f\big(\theta^\circ + \delta + \Delta\theta(\theta^\circ + \delta)\big) = d_f(\theta^\circ) + \phi_f^\top\boldsymbol{S}\delta,$$

which is the pre-update score $d_f(\theta^\circ) + \phi_f^\top\delta'$ evaluated at $\delta' = \boldsymbol{S}\delta$. Substituting into $\frac{1}{|\mathcal{F}|}\sum_f \mathrm{relu}(t_f - \cdot)^2$ gives $\mathcal{L}_{\mathrm{ant}}(\delta) = \mathcal{L}_{\mathrm{plain}}(\boldsymbol{S}\delta)$ for the draw N. The filter depends on the draw through $\boldsymbol{\Phi}$, and the expectation over draws is $\mathbb{E}_N[\mathcal{L}_{\mathrm{plain}}(\boldsymbol{S}_N\delta)]$.

The minimum. The map $x \mapsto \mathrm{relu}(t - x)^2$ is non-increasing, so its value at the smaller of two scores is the larger of its two values, $\mathrm{relu}(t - \min\{a, b\})^2 = \max\{\mathrm{relu}(t - a)^2, \mathrm{relu}(t - b)^2\}$. With a the current score and b the anticipated score, the composed loss of fact f is $\max\{\ell_f(\delta), \ell_f(\boldsymbol{S}\delta)\}$, and it equals $\ell_f(\boldsymbol{S}\delta)$ when the anticipated score is the smaller of the two. The current score is $d_f(\theta^\circ) + \phi_f^\top\delta$, so the anticipated score is the smaller of the two if and only if $\phi_f^\top\boldsymbol{S}\delta \le \phi_f^\top\delta$, that is, $\phi_f^\top(\mathbf{I} - \boldsymbol{S})\delta \ge 0$. For a uniform ridge, equation 14 gives $\mathbf{I} - \boldsymbol{S} = \sum_{k\le\varrho}\frac{\lambda_k}{\lambda_k+\sigma}\boldsymbol{u}_k\boldsymbol{u}_k^\top$, which is symmetric positive semidefinite. For $\delta = c\,\phi_f$ with $c \ge 0$ the condition reads $c\,\phi_f^\top(\mathbf{I} - \boldsymbol{S})\phi_f \ge 0$, which holds. For $\delta = c\,\boldsymbol{S}\phi_f$ the condition reads $c\,\phi_f^\top(\mathbf{I} - \boldsymbol{S})\boldsymbol{S}\phi_f \ge 0$, and $(\mathbf{I} - \boldsymbol{S})\boldsymbol{S} = \sum_{k\le\varrho}\frac{\sigma\lambda_k}{(\lambda_k+\sigma)^2}\boldsymbol{u}_k\boldsymbol{u}_k^\top$ is symmetric positive semidefinite. For a general diagonal $\boldsymbol{\Sigma}$ write $\tilde{\boldsymbol{C}} = \boldsymbol{\Sigma}^{-1/2}\boldsymbol{C}\boldsymbol{\Sigma}^{-1/2}$. Then $\mathbf{I} - \boldsymbol{S} = (\boldsymbol{C} + \boldsymbol{\Sigma})^{-1}\boldsymbol{C} = \boldsymbol{\Sigma}^{-1/2}(\tilde{\boldsymbol{C}} + \mathbf{I})^{-1}\tilde{\boldsymbol{C}}\,\boldsymbol{\Sigma}^{1/2}$. For $\delta = c\,\boldsymbol{\Sigma}^{-1}\phi_f$ with $c \ge 0$ and $\boldsymbol{z} = \boldsymbol{\Sigma}^{-1/2}\phi_f$, the condition reads $c\,\boldsymbol{z}^\top(\tilde{\boldsymbol{C}} + \mathbf{I})^{-1}\tilde{\boldsymbol{C}}\,\boldsymbol{z} \ge 0$. It holds because

$(\tilde{\boldsymbol{C}}+\mathbf{I})^{-1}\tilde{\boldsymbol{C}}$ is symmetric positive semidefinite. The matrix $\mathbf{I}-\boldsymbol{S}$ itself is symmetric exactly when $\boldsymbol{\Sigma}$ commutes with $\boldsymbol{C}$. That holds when $\sigma_u=\sigma_v$ for every pair of entities whose off-diagonal block of $\boldsymbol{C}$ is nonzero, so the statement takes the uniform ridge.

The gradient. The composed loss of fact f is $\mathrm{relu}(t_f-\tilde{d}_f)^2$, and the stop-gradient composition of Appendix B.3 takes its gradient through the anticipated score on both branches of the minimum. In the linearized model that gradient with respect to δ is $-2\,\mathrm{relu}(t_f-\tilde{d}_f)\,\boldsymbol{S}^\top\phi_f$, with $\boldsymbol{S}^\top=\boldsymbol{\Sigma}(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}$, the operator of part (c) of the proof of Theorem A.1. The branch enters through the scalar $\mathrm{relu}(t_f-\tilde{d}_f)$ alone, so the minimum changes the weight of $\boldsymbol{S}^\top\phi_f$ and leaves its direction unchanged.

Nonlinear scores. For a score that is not linear in the rows, freeze the features at the current point θ, evaluate the post-update score to first order and use the signed residual. Part (c) of the proof of Theorem A.1 shows that the gradient of the anticipated score with respect to the stacked rows is then $\boldsymbol{S}^\top\phi_f$ with $\boldsymbol{\Phi}$ taken at θ. With the per-entity response in place of the joint one, the part of this gradient that passes through v's own response is Theorem A.1. The two agree when no drawn edge joins two involved entities (Lemma A.4). The identity between the two objectives therefore holds to first order at each step, with the filter rebuilt from the features at that step. □

Proof of Corollary A.2. Part (a). In equation 14 every λ_k is positive, so $\sigma/(\lambda_k+\sigma)\to 0$ as $\sigma\to 0$ and $\boldsymbol{S}\to\Pi^\perp$. In the gradient-descent form, $0\le 1-\eta\lambda_k<1$, so $(1-\eta\lambda_k)^T\to 0$ as $T\to\infty$ and $(\mathbf{I}-\eta\boldsymbol{C})^T\to\Pi^\perp$. With $\sigma=1/(\eta T)$ the two limits are the same limit. If $\boldsymbol{\Phi}\delta=0$, then $\boldsymbol{S}\delta=\delta$ for every $\boldsymbol{\Sigma}$ by the second step of the proof of Proposition A.1, and $(\mathbf{I}-\eta\boldsymbol{C})^T\delta=\delta$ because $\boldsymbol{C}\delta=0$. Confining the edit to $\{\delta:\boldsymbol{\Phi}\delta=0\}=\mathrm{range}(\Pi^\perp)$ is projection onto the null space of the update's features. Let the update class consist of the retained edges at the involved entities, with every one of them drawn. The rows of $\boldsymbol{\Phi}$ are then the features of those edges, and the constraint reads $\phi_i^\top\delta=0$ over them. Limit 2 of the proof of Corollary A.3 identifies this constraint with that of the retained-span methods, for a retained loss that depends on θ through the retained scores. It also records what a ranking loss with sampled negatives adds. The projector here is the joint one over the involved rows, and Limit 2 is its restriction to one row.

Part (b). Let $\delta\in U_{\bar\lambda}$ and expand $\delta=\sum_{k:\lambda_k\le\bar\lambda}c_k\boldsymbol{u}_k+\delta_\perp$ with $\delta_\perp=\Pi^\perp\delta$. By equation 14 and orthonormality,

$$\|\boldsymbol{S}\delta\|_2^2=\sum_{k:\lambda_k\le\bar\lambda}\Big(\frac{\sigma}{\lambda_k+\sigma}\Big)^2c_k^2+\|\delta_\perp\|_2^2\ge\Big(\frac{\sigma}{\bar\lambda+\sigma}\Big)^2\Big(\sum_{k:\lambda_k\le\bar\lambda}c_k^2+\|\delta_\perp\|_2^2\Big)=\Big(\frac{\sigma}{\bar\lambda+\sigma}\Big)^2\|\delta\|_2^2.$$

The inequality holds because $\lambda\mapsto\sigma/(\lambda+\sigma)$ is decreasing, so each factor with $\lambda_k\le\bar\lambda$ is at least $\sigma/(\bar\lambda+\sigma)$, and because $1\ge\sigma/(\bar\lambda+\sigma)$. For $\delta=\sum_{k:\lambda_k>\bar\lambda}c_k\boldsymbol{u}_k$ each factor is at most $\sigma/(\bar\lambda+\sigma)$, which gives $\|\boldsymbol{S}\delta\|_2\le\frac{\sigma}{\bar\lambda+\sigma}\|\delta\|_2$. The same two bounds hold for the gradient-descent filter with $(1-\eta\bar\lambda)^T$ in place of $\sigma/(\bar\lambda+\sigma)$ when $\bar\lambda\le 1/\eta$, because $\lambda\mapsto(1-\eta\lambda)^T$ is decreasing on $[0,1/\eta]$.

The reading of the slowly restored directions. A method of this kind confines the edit to the orthogonal complement of the top j eigenvectors of a second-moment matrix that it computes from a proxy. If that matrix shares its top j eigenvectors with $\boldsymbol{C}$, and $\lambda_j>\lambda_{j+1}$ when $j<\varrho$, this complement is $U_{\bar\lambda}$ for every $\bar\lambda\in[\lambda_{j+1},\lambda_j)$, and for every $\bar\lambda\in(0,\lambda_\varrho)$ when $j=\varrho$, because $U_{\bar\lambda}$ is then the orthogonal complement of $\mathrm{span}\{\boldsymbol{u}_1,\dots,\boldsymbol{u}_j\}$. The shared-eigenstructure assumption of Xiao et al. (2026) is a hypothesis of this form. The corollary is a statement about the filter of the linearized model of this paper. The published methods compute their spectra from the activations or gradients of deep networks and use their own losses, so the correspondence concerns the form of the constraint. □

Proof of Remark A.1. In the linearized model the score of drawn edge i is $d_i(\theta^\circ+\delta)=d_i(\theta^\circ)+\phi_i^\top\delta$, so the one-sided residual of equation 4 at $\theta^\circ+\delta$ is $r=\mathrm{relu}(\boldsymbol{\Phi}\delta)$ with relu applied entrywise, and the solve of equation 6 returns $-(\boldsymbol{C}+\boldsymbol{\Sigma})^{-1}\boldsymbol{\Phi}^\top\mathrm{relu}(\boldsymbol{\Phi}\delta)$. If $\boldsymbol{\Phi}\delta\le 0$ entrywise, then $r=0$, the response vanishes and the anticipated point is $\theta^\circ+\delta$. Conversely, let the response vanish and write $\boldsymbol{a}=\mathrm{relu}(\boldsymbol{\Phi}\delta)\ge 0$. Then $\boldsymbol{\Phi}^\top\boldsymbol{a}=0$, hence $\boldsymbol{a}^\top\boldsymbol{\Phi}\delta=0$. Entry by entry, $a_i(\boldsymbol{\Phi}\delta)_i=\mathrm{relu}((\boldsymbol{\Phi}\delta)_i)^2=a_i^2$, so $\boldsymbol{a}^\top\boldsymbol{\Phi}\delta=\|\boldsymbol{a}\|_2^2$, and therefore $\boldsymbol{a}=0$, that is, $\boldsymbol{\Phi}\delta\le 0$. The edits the update leaves untouched are exactly the closed convex cone $\{\delta:\boldsymbol{\Phi}\delta\le 0\}$. It contains the subspace $\{\delta:\boldsymbol{\Phi}\delta=0\}$, which is the corresponding set under the signed residual by Proposition A.1. □

A.8 Proof of Proposition A.1, per-entity form, and the separate penalty

Proof of Proposition A.1, per-entity form. Fix a draw and a point θ, and let δ move the row $\boldsymbol{e}_v$ alone, by δ_v. The residuals of the drawn edges at v then change by $\boldsymbol{\Phi}_v\delta_v$, and nothing else changes. With $\boldsymbol{A}_v$ frozen and $r_v(\theta+\delta) = r_v(\theta) + \boldsymbol{\Phi}_v\delta_v$,

$$\Delta\boldsymbol{e}_v(\theta+\delta) = -\boldsymbol{A}_v\big(r_v(\theta) + \boldsymbol{\Phi}_v\delta_v\big) = \Delta\boldsymbol{e}_v(\theta) - \boldsymbol{A}_v\boldsymbol{\Phi}_v\delta_v,$$

so $\boldsymbol{e}_v + \delta_v + \Delta\boldsymbol{e}_v(\theta+\delta) = \boldsymbol{e}_v + \Delta\boldsymbol{e}_v(\theta) + (\mathbf{I} - \boldsymbol{A}_v\boldsymbol{\Phi}_v)\delta_v = \boldsymbol{e}_v + \Delta\boldsymbol{e}_v(\theta) + \boldsymbol{P}^{\perp}_{N,v}\delta_v$, which is the first claim. For the constant, expand $\delta_v \in \mathrm{range}(\boldsymbol{\Phi}_v^{\top})$ as $\delta_v = \sum_{k\le\varrho} c_k\boldsymbol{u}_k$. By equation 11, $\boldsymbol{P}^{\perp}_{N,v}\delta_v = \sum_{k\le\varrho}\frac{\sigma_v}{\lambda_k+\sigma_v}c_k\boldsymbol{u}_k$, and therefore

$$\|\boldsymbol{P}^{\perp}_{N,v}\delta_v\|_2^2 = \sum_{k\le\varrho}\Big(\frac{\sigma_v}{\lambda_k+\sigma_v}\Big)^2 c_k^2 \le \Big(\frac{\sigma_v}{\lambda^{+}_{\min}(\mathbf{K}_v)+\sigma_v}\Big)^2\|\delta_v\|_2^2,$$

the factor being largest on the direction of the smallest nonzero eigenvalue. Under first-order evaluation the anticipated score moves by $\phi_f^{(v)\top}\boldsymbol{P}^{\perp}_{N,v}\delta_v$, of magnitude at most $\|\phi_f^{(v)}\|_2$ times the bound. Under exact evaluation the anticipated point moves by the same bounded vector, and the score at the shifted rows is Lipschitz in each row on any region where the row norms are bounded below, so the anticipated score again moves by at most a constant times the bound.

Consequence. Let $\hat\rho_f(\theta) = d_f(\theta) - d_f^{\mathrm{lin}}(\theta)$ be the predicted score recovery under first-order evaluation. Under the displacement δ_v, the pre-update score d_f changes by $\phi_f^{(v)\top}\delta_v$ to first order, and the anticipated score by $\phi_f^{(v)\top}\boldsymbol{P}^{\perp}_{N,v}\delta_v$, so $\hat\rho_f$ changes by $\phi_f^{(v)\top}(\mathbf{I} - \boldsymbol{P}^{\perp}_{N,v})\delta_v = \phi_f^{(v)\top}\boldsymbol{A}_v\boldsymbol{\Phi}_v\delta_v$, which differs from $\phi_f^{(v)\top}\delta_v$ by $\phi_f^{(v)\top}\boldsymbol{P}^{\perp}_{N,v}\delta_v$, of magnitude at most $\|\phi_f^{(v)}\|_2$ times the bound above. A displacement that undoes part of the forgetting of f, that is, one with $\phi_f^{(v)\top}\delta_v < 0$, therefore lowers $\hat\rho_f$ by the same amount up to that term. An objective of the form $\mathrm{relu}(t_f - d_f(\theta))^2 + \zeta\,\mathrm{relu}(\hat\rho_f)^2$ pays in its first term and is rewarded in its second. The two quantities it targets, the pre-update score and the predicted score recovery, fall by the same amount up to that term. The predicted post-update score moves by at most that term. The anticipated forgetting loss acts on $\min\{d_f(\theta), d_f(\theta+\Delta\theta)\}$. The displacement lowers the first argument and moves the second by at most that term, so the loss falls by at most what that term allows. □

A.9 Expected force of the margin-loss update

This subsection supports the choice of the one-sided residual equation 4 in Section 4.1. The margin loss of the update on edge i is $\ell_i = \max\{0, d_i - \bar n_i + \gamma\}$, with γ the margin and $\bar n_i$ the mean score of K fresh random corruptions of i, redrawn at every step ($K = 10$ and $\gamma = 8$, Appendix B.9). Let μ_i be the mean corrupted score of edge i under θ°, which the unlearner can compute, and let τ_{n} be the standard deviation of the K-negative mean $\bar n_i$ about μ_i, the per-negative spread divided by $\sqrt{K}$. Write Φ_{G} and φ_{G} for the standard normal distribution and density functions.

Lemma A.2 (Expected margin force)**.** *Model $\bar n_i \sim \mathrm{Normal}(\mu_i, \tau_{\mathrm{n}}^2)$ and set $g_i = \mu_i - \gamma$. Then, at score d_i,*

$$\mathbb{E}\Big[\frac{\partial\ell_i}{\partial d_i}\Big] = \Pr(\bar n_i < d_i + \gamma) = \Phi_{\mathrm{G}}\Big(\frac{d_i - g_i}{\tau_{\mathrm{n}}}\Big),$$

$$\mathbb{E}[\ell_i] = \tau_{\mathrm{n}}\,\psi\Big(\frac{d_i - g_i}{\tau_{\mathrm{n}}}\Big), \qquad \psi(z) = z\,\Phi_{\mathrm{G}}(z) + \varphi_{\mathrm{G}}(z),$$

and $\psi' = \Phi_{\mathrm{G}}$, so the expected loss is the Gaussian softplus of $d_i - g_i$ at temperature τ_{n} and the expected force is its sigmoidal gate.

Proof. Almost surely $\partial\ell_i/\partial d_i = \mathbf{1}[d_i - \bar n_i + \gamma > 0]$, whose expectation is $\Pr(\bar n_i < d_i + \gamma) = \Phi_{\mathrm{G}}((d_i+\gamma-\mu_i)/\tau_{\mathrm{n}})$. For the loss, write $\bar n_i = \mu_i + \tau_{\mathrm{n}}\xi$ with $\xi \sim \mathrm{Normal}(0,1)$ and $z = (d_i - g_i)/\tau_{\mathrm{n}}$. Then $\ell_i = \tau_{\mathrm{n}}(z - \xi)_+$ and

$$\mathbb{E}[(z-\xi)_+] = \int_{-\infty}^{z}(z-\xi)\varphi_{\mathrm{G}}(\xi)\,d\xi = z\,\Phi_{\mathrm{G}}(z) + \varphi_{\mathrm{G}}(z),$$

using $\int_{-\infty}^{z}\xi\varphi_{\mathrm{G}}(\xi)\,d\xi = -\varphi_{\mathrm{G}}(z)$. Differentiating gives $\psi'(z) = \Phi_{\mathrm{G}}(z) + z\varphi_{\mathrm{G}}(z) - z\varphi_{\mathrm{G}}(z)$, and the last two terms cancel, so $\psi'(z) = \Phi_{\mathrm{G}}(z)$. □

The one-sided residual as the zero-slack bound. Define the slack of edge i as $\varsigma_i = g_i - d_i(\theta^\circ)$. An edge with $\varsigma_i > 0$ is inert under θ° and becomes active only once the edit has raised it by ς_i. As $\tau_{\mathrm{n}} \to 0$, $\tau_{\mathrm{n}}\psi(a/\tau_{\mathrm{n}}) \to a_+$ for every real a, so the expected loss tends to $(d_i - d_i(\theta^\circ) - \varsigma_i)_+$ and the expected force to $\mathbf{1}[d_i > d_i(\theta^\circ) + \varsigma_i]$. Setting every slack to zero gives the one-sided residual $r_i = \mathrm{relu}(d_i(\theta) - d_i(\theta^\circ))$ of equation 4. This choice bounds the force the update applies from above in the following sense. For every slack $\varsigma_i \geq 0$ and every temperature $\tau_{\mathrm{n}} > 0$, the increase of the expected loss over its value at θ° satisfies

$$\mathbb{E}[\ell_i](d_i) - \mathbb{E}[\ell_i](d_i(\theta^\circ)) \leq \mathrm{relu}\left(d_i - d_i(\theta^\circ)\right),$$

because $\mathbb{E}[\ell_i]$ is non-decreasing with slope $\Phi_{\mathrm{G}} \leq 1$. A raise of $a > 0$ above the original level increases the expected loss by at most a, and a lowering does not increase it. The one-sided residual credits every raise in full and no lowering at all, which is the conservative reading of the force the update applies. The gate of Section 4.1, with its measured slack and temperature, stores suppression along the features of edges with large slack. Table 17 reports it as the per-relation gate beside equation 4.

Why inert edges move. The gate also explains why edges that never violate their margin are restored. Let $\mathcal{A}_\gamma \subseteq P_v(N)$ be the active edges of a draw, those that violate their margin and the only ones on which the margin loss of the update exerts force. A response built from those edges alone, $\Delta \boldsymbol{e}_v = -\boldsymbol{\Phi}_{\mathcal{A}_\gamma}^\top (\boldsymbol{\Phi}_{\mathcal{A}_\gamma}\boldsymbol{\Phi}_{\mathcal{A}_\gamma}^\top + \sigma_v \mathbf{I})^{-1} r_{\mathcal{A}_\gamma}$, changes the scores of the inert edges by $\boldsymbol{\Phi}_{\neg\mathcal{A}_\gamma}\Delta\boldsymbol{e}_v$, which is in general nonzero when the inert features are not orthogonal to the active ones. The edges an edit δ_v raises most are those whose features are most aligned with δ_v. Correcting them moves $\boldsymbol{e}_v$ along $-\phi_i^{(v)}$, hence back along $-\delta_v$, and every inert edge whose feature has a component along δ_v follows. This side effect is automatic in a response built from the active edges, and it is the mechanism by which an edit is undone although most of the update edges are never active. The response of equation 6 differs on this point. It keeps every drawn edge in the kernel, so it holds a drawn edge of zero residual near its current score (part (b) of the proof of Theorem A.1), and under equation 4 every raised edge is active.

A.10 Expected loss over the update class

This subsection supports the statement of Section 4.1 that the loss is averaged over draws and is never applied to an averaged response.

Lemma A.3 (Expected loss over draws)**.** *Let $\ell_f(d) = \mathrm{relu}(t_f - d)^2$ be the forgetting loss of equation 2, which is convex and non-increasing in d. Under first-order evaluation, $d_f^{\mathrm{lin}}(\theta; N) = d_f(\theta) + \sum_v \phi_f^{(v)\top}\Delta\boldsymbol{e}_v(\theta; N)$ is affine in the response, so $\mathbb{E}_N[d_f^{\mathrm{lin}}(\theta; N)]$ is the first-order score of the expected response $\mathbb{E}_N[\Delta\theta(\theta; N)]$, and*

$$\ell_f\big(\mathbb{E}_N[d_f^{\mathrm{lin}}(\theta; N)]\big) \leq \mathbb{E}_N\big[\ell_f(d_f^{\mathrm{lin}}(\theta; N))\big] \leq \max_N \ell_f\big(d_f^{\mathrm{lin}}(\theta; N)\big).$$

Proof. The left inequality is Jensen's inequality for the convex function ℓ_f, applied to the finite distribution of the draw. The right inequality bounds an average by its maximum over the finite support. □

The loss of the expected response therefore underestimates the expected loss, and the expected loss lies between it and the worst case over the class. This is why equation 3 averages the loss over draws and never applies it to an averaged response. The expected-draw row of Table 17 replaces the random inclusion of edges by the response $\Delta\theta_{\boldsymbol{W}} = -\boldsymbol{\Phi}^\top \boldsymbol{W}^{1/2}(\boldsymbol{W}^{1/2}\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{W}^{1/2} + \sigma\mathbf{I})^{-1}\boldsymbol{W}^{1/2} r$, with $\boldsymbol{W}$ the diagonal matrix of inclusion probabilities. This response is not $\mathbb{E}_N[\Delta\theta(\theta; N)]$, because the response is a nonlinear function of the draw, so the lemma gives no ordering for it. Under exact evaluation the score is not affine in the response, and the lemma gives no ordering either. The expected loss is then used as the objective over the update class without this comparison.

A.11 Block-diagonal versus joint response

This subsection supports the statement of Section 4.2 that the per-entity form counts the residual of a shared edge once at each endpoint. Stack the involved rows as $\Delta = (\Delta\boldsymbol{e}_v)_{v\in\mathcal{V}_\mathcal{F}} \in \mathbb{R}^{m|\mathcal{V}_\mathcal{F}|}$ and

let $\mathbf{\Phi} \in \mathbb{R}^{|N|\times m|\mathcal{V}_\mathcal{F}|}$ be the full Jacobian of the drawn edges, whose row i carries the block $\phi_i^{(h)\top}$ at its head and $\phi_i^{(t)\top}$ at its tail, a block being absent when that endpoint is not an involved entity. With $\mathbf{\Sigma} = \mathrm{diag}(\sigma_v \mathbf{I}_m)_v$, the joint response solves $(\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Sigma})\Delta = -\mathbf{\Phi}^\top r$, and the block-diagonal response solves, for each v separately, $(\mathbf{\Phi}_v^\top\mathbf{\Phi}_v + \sigma_v\mathbf{I})\Delta\boldsymbol{e}_v^{\mathrm{bd}} = -\mathbf{\Phi}_v^\top r_v$, that is, $\Delta\boldsymbol{e}_v^{\mathrm{bd}} = -\boldsymbol{A}_v r_v$, stacked as Δ^{bd}.

Lemma A.4 (Block-diagonal versus joint response)**.** *The off-diagonal block of $\mathbf{\Phi}^\top\mathbf{\Phi}$ between entities $v \neq u$ is $\sum_{i\ni v,u} \phi_i^{(v)}\phi_i^{(u)\top}$, a sum over the drawn edges whose two endpoints are v and u. If no drawn edge has both endpoints among the involved entities, the two responses coincide for every residual. Moreover, in the limit in which every $\sigma_v \to 0$, with the features of the drawn edges at each entity linearly independent (which makes the full Jacobian of full row rank), a drawn edge i with both endpoints involved and $r_i \neq 0$ has its residual removed twice by the block-diagonal response and once by the joint one: $\phi_i^\top\Delta^{\mathrm{bd}} = -2r_i$ whereas $\phi_i^\top\Delta = -r_i$.*

Proof. The block structure follows from $\mathbf{\Phi}^\top\mathbf{\Phi} = \sum_i \phi_i\phi_i^\top$ and the support of ϕ_i on the blocks of its two endpoints. If no edge couples two involved entities, every off-diagonal block is a sum over an empty set, $\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Sigma}$ is block diagonal, and the joint system separates into the block-diagonal systems, which proves the first claim. For the last claim, in the limit $\sigma_v \to 0$ with $\mathbf{\Phi}_v$ of full row rank, $\mathbf{\Phi}_v\Delta\boldsymbol{e}_v^{\mathrm{bd}} = -\mathbf{\Phi}_v\mathbf{\Phi}_v^\top\mathbf{K}_v^{-1}r_v = -r_v$ at every entity, so each endpoint's own solve interpolates the full residual of every edge incident to it, and an edge with two involved endpoints has $\phi_i^\top\Delta^{\mathrm{bd}} = \phi_i^{(h)\top}\Delta\boldsymbol{e}_h^{\mathrm{bd}} + \phi_i^{(t)\top}\Delta\boldsymbol{e}_t^{\mathrm{bd}} = -2r_i$. The joint system in the same limit interpolates all drawn edges at once, $\mathbf{\Phi}\Delta = -r$, because $\mathbf{\Phi}(\mathbf{\Phi}^\top\mathbf{\Phi} + \mathbf{\Sigma})^{-1}\mathbf{\Phi}^\top = \tilde{\mathbf{K}}(\tilde{\mathbf{K}} + \mathbf{I})^{-1}$ with $\tilde{\mathbf{K}} = \mathbf{\Phi}\mathbf{\Sigma}^{-1}\mathbf{\Phi}^\top \succeq \mathbf{\Phi}\mathbf{\Phi}^\top / \max_v \sigma_v$, and this matrix tends to the identity when $\mathbf{\Phi}$ has full row rank, so $\phi_i^\top\Delta = -r_i$. □

The joint system is symmetric positive definite of dimension $m|\mathcal{V}_\mathcal{F}|$ and is solved by conjugate gradients (Hestenes & Stiefel, 1952) without forming a matrix. Each iteration costs one product with $\mathbf{\Phi}$ and one with $\mathbf{\Phi}^\top$, both linear in the number of drawn edges, plus the diagonal $\mathbf{\Sigma}$. The block-diagonal response is a single block-Jacobi sweep of the same system.

A.12 Within-row isotropy and the ridge

This subsection supports the statement of Section 4.2 that the response leaves out Adam's per-coordinate scaling.

Remark A.4 (Within-row isotropy). For the TransE score the coordinates of the slot feature in equation 10 are $\phi_{ij}^{(v)} = \pm\big(s_{ij} - \bar{e}_{vj}\,(\bar{\boldsymbol{e}}_v^\top\boldsymbol{s}_i)\big)/\|\boldsymbol{e}_v\|_2$ with $\boldsymbol{s}_i$ the sign vector of the translation error and $s_{ij} \in \{\pm 1\}$, so

$$\Big|\,|\phi_{ij}^{(v)}|\,\|\boldsymbol{e}_v\|_2 - 1\,\Big| \le |\bar{e}_{vj}|\,|\bar{\boldsymbol{e}}_v^\top\boldsymbol{s}_i| \qquad \text{for every coordinate } j,$$

by expanding $(\mathbf{I} - \bar{\boldsymbol{e}}_v\bar{\boldsymbol{e}}_v^\top)\boldsymbol{s}_i$ coordinatewise and using $|s_{ij}| = 1$. For a row whose mass is spread over its m coordinates, $|\bar{e}_{vj}|$ is of order $m^{-1/2}$, and $|\bar{\boldsymbol{e}}_v^\top\boldsymbol{s}_i|$ is of order one for a sign pattern not aligned with the row, so every coordinate of every feature at v has magnitude close to $1/\|\boldsymbol{e}_v\|_2$. Features are therefore isotropic within a row and differ across rows by the scalar $1/\|\boldsymbol{e}_v\|_2$. Two consequences follow. Adam's per-coordinate second-moment preconditioner $\boldsymbol{V}$ (Kingma & Ba, 2015) is then approximately a scalar per row, so the Adam-preconditioned kernel $\mathbf{\Phi}\boldsymbol{V}^{-1}\mathbf{\Phi}^\top$ differs from the Euclidean kernel only by per-row scalars, the sign-gradient kernel $\langle \mathrm{sign}\,\phi_i, \mathrm{sign}\,\phi_j\rangle$ of Malladi et al. (2023) is, for this score, approximately the Euclidean kernel up to the same scalars, and the metric orthogonality of Zhou et al. (2025) is Euclidean orthogonality up to row scalars. Those scalars can be absorbed in a per-entity ridge, which is the second consequence, and it is stated as a hypothesis. Dense Adam over the embedding table keeps a second-moment average per coordinate, so for a row that receives gradient in a fraction p_v of the steps the average is about p_v times the squared active gradient, each active step moves the row by about $\eta/\sqrt{p_v}$, and the movement over T steps is about $\eta T\sqrt{p_v}$, against $\eta T p_v$ under stochastic gradient descent. Reading this as an effective per-row learning rate $\eta_v \propto \sqrt{p_v}$ and inserting it in Remark A.2 gives

$$\sigma_v = \sigma_0\,(p_v/p_{\mathrm{med}})^{-\kappa},$$

$$\kappa = \tfrac{1}{2}\ \text{(Adam)}, \quad \kappa = 1\ \text{(stochastic gradient descent)}, \quad \kappa = 0\ \text{(degree-independent)},$$

with p_v proportional to the degree of v in the update class and p_{med} its median. Under this hypothesis rare entities are fitted less completely by the update than entities of high degree, but only as $\sqrt{p_v}$. Every experiment in this paper uses the degree-independent choice $\kappa = 0$, with σ_v scaled to the feature norms at v only (Section 4.2).

A.13 Piecewise-linear scores and exact evaluation

The anticipated score is evaluated at the shifted rows for a reason that holds on every backbone. The score is nonlinear in the rows, and the displacement the update produces is not small on the scale of the coordinate differences from which the score is built. On an L_1 score the gap between the first-order evaluation and the exact one has an elementary closed form.

Lemma A.5 (Piecewise-linear scores)**.** *For real x and D, with the convention* $\operatorname{sign} 0 = 1$*,*

$$|x + D| - |x| = \operatorname{sign}(x)\, D + 2\big(|D| - |x|\big)_+ \mathbf{1}\big[\operatorname{sign} D \neq \operatorname{sign} x\big]. \tag{15}$$

The second term is non-negative, vanishes whenever $|D| \leq |x|$, and equals $2(|D| - |x|)$, linear in $|D|$ with slope two, once $|D| > |x|$ and D opposes x.

Proof. If $\operatorname{sign} D = \operatorname{sign} x$, then $|x + D| = |x| + |D|$ and $\operatorname{sign}(x)D = |D|$, so both sides equal $|D|$ and the indicator is zero. If $\operatorname{sign} D \neq \operatorname{sign} x$ and $|D| \leq |x|$, then $x + D$ has the sign of x, $|x + D| = |x| - |D| = |x| + \operatorname{sign}(x)D$, and the second term vanishes because $(|D| - |x|)_+ = 0$. If $\operatorname{sign} D \neq \operatorname{sign} x$ and $|D| > |x|$, then $x + D$ has the sign of D, so the left-hand side is $(|D| - |x|) - |x| = |D| - 2|x|$, and the right-hand side is $\operatorname{sign}(x)D + 2(|D| - |x|) = -|D| + 2|D| - 2|x| = |D| - 2|x|$. The case $x = 0$ falls under the convention. With $\operatorname{sign} 0 = 1$, $D > 0$ is the first case and $D < 0$ the third, and both give $|D|$. The listed properties of the second term are read off the three cases. □

For a triple with translation error $\boldsymbol{x} = \bar{\boldsymbol{e}}_h + \bar{\boldsymbol{w}}_r - \bar{\boldsymbol{e}}_t$, so that $d_x = \|\boldsymbol{x}\|_1$, and a displacement $\boldsymbol{D}$ of that vector induced by the response, summing equation 15 over coordinates gives $d_x(\theta + \Delta\theta) - d_x(\theta) = \boldsymbol{s}^\top \boldsymbol{D} + \varkappa(\boldsymbol{D})$ with $\boldsymbol{s} = \operatorname{sign}(\boldsymbol{x})$ and $\varkappa \geq 0$. The first term is, up to the normalization's second-order term, the first-order evaluation $\phi_x^\top \Delta\theta$ that every operator acting on tangent features controls. The second is non-negative, vanishes on the box $|D_j| \leq |x_j|$, is not linear in $\boldsymbol{D}$, and depends on the magnitudes $|x_j|$ as well as on the signs, so no operator built from the tangent feature alone models it. The second term matters when $|D_j|$ exceeds $|x_j|$ in some coordinates, and Appendix D.5 measures the consequence. Evaluating the score at the shifted rows captures both terms at the cost of one call to the score function, which is why KAU does so.

B Method details

Section 4 defines the objective and names its parts. This appendix builds them. The first six subsections give the update draw, the targets, the composition rule, the retraction rule, the score-preservation term and the selection rule. The seventh lists one run as Algorithm 1. The eighth gives the three backbones and the way a slot feature is taken from each. The last three give every hyperparameter, the cost of a run, and three constructions the derivation admits that we do not use. Everything here is computed from the original model θ°, the forget set $\mathcal{F}$ and the retained edges $\mathcal{D}_R$. No update edge, no held-out edge and no retrained model enters at run time.

B.1 The update draw and its stand-ins

The unlearner never learns which edges the update will use, so it models that edge set as a random draw with two channels. The known channel is made of the unlearner's own retained edges. At an involved entity v, each retained edge incident to v enters a draw independently with probability ρ_{kn}. Whatever retained edges the user's update includes at v are a subset of these, so this channel contains genuine members of the update class.

The unseen channel stands in for true edges that were in no training set. The unlearner has never seen such an edge, so it manufactures stand-ins. It keeps a pool of 32 stand-in triples at every involved entity, and each stand-in is built in two steps. A relation is drawn from the empirical relation profile

of v over its retained edges. A partner entity is then drawn from the entities that occur with that relation in the retained set. Four candidates are built this way and the one the original model finds most plausible is kept, which is the candidate with the lowest score. A candidate that already occurs in $\mathcal{F}$ or in $\mathcal{D}_R$ is rejected, so a stand-in is neither a forgotten nor a retained fact. The pool is refreshed every five epochs. Each stand-in then enters a draw with probability $\min\{1, \rho_{\mathrm{un}} \deg(v)/32\}$, which puts the expected number of unseen edges at v at $\min\{32, \rho_{\mathrm{un}} \deg(v)\}$ and lets that number grow with the entity's degree up to the pool size.

Every draw obeys two constraints. At most $k_{\max} = 96$ edges enter per entity. The cap lies below the largest rank the features at v can reach, which is $m - 1$ on TransE and TransH, whose normalization removes the radial direction (Appendix B.8), and m on RotatE. Every draw therefore leaves directions at v off its span, where an edit survives the response entirely (Section 4.2, Appendix A.6). The second constraint is that a fresh draw is taken at every forgetting step, so the objective is the expected loss over the class. Over a run each known edge is drawn several times.

The residual of equation 4 is evaluated on the drawn edges at the current parameters. For a stand-in it uses that triple's own score under the original model. A forgotten fact carries no residual, and its loss is measured against the target of Appendix B.2.

B.2 Targets

The target t_f is the score at which the forgetting loss stops pushing, and it is built from the original model alone. For every forgotten fact we replace one endpoint at a time by a uniformly drawn entity and score 128 corrupted copies under θ°, 64 with the head replaced and 64 with the tail replaced. Let q_f be the 5% quantile of those scores, the level at which the original model holds a fact it does not believe. The target of equation 7 is $t_f = d_f(\theta^\circ) + s\ \mathrm{relu}(q_f - d_f(\theta^\circ))$, with the default $s = 0.15$ and the value selected per backbone in Table 3, computed once before the run and never updated. A fact the original model already scores near that tail is asked to move very little, and a well-believed fact is asked to move a fixed fraction of the way toward it. Because the loss is one-sided and exerts no force past t_f, the operating point of the release is set by these targets rather than by the epoch at which the run happens to end. The score-preservation term of Appendix B.5 and the two ancillaries of Appendices B.4 and B.6 leave those targets alone and act on the rows instead.

B.3 Pessimistic composition

The loss of equation 3 acts on $\tilde{d}_f = \min\{d_f(\theta), d_f(\theta + \Delta\theta)\}$, so a fact counts as forgotten only if it is forgotten both now and after the predicted update. The gradient is routed through the anticipated path in either case. The composed score is computed as

$$\tilde{d}_f = d_f(\theta + \Delta\theta) - \mathrm{sg}\left[\,\mathrm{relu}\left(d_f(\theta + \Delta\theta) - d_f(\theta)\right)\right],$$

with sg the stop-gradient, which has the value of the minimum and the gradient of the anticipated score. The optimizer step therefore always targets the post-update position. Because of the minimum, a predicted rise of the fact's score cannot lower the loss, which closes the shortcut of Section 4.1.

B.4 Retraction

After every optimizer step, each edited entity row is rescaled to the length it had under the original model, $\boldsymbol{e}_v \leftarrow \|\boldsymbol{e}_v^\circ\|_2\, \boldsymbol{e}_v / \|\boldsymbol{e}_v\|_2$. The reason is that the slot feature at v scales as $1/\|\boldsymbol{e}_v\|_2$. A model whose edited rows have grown therefore presents a smaller gradient to the update after release, on exactly the rows that were edited. Its resistance is then a matter of scale rather than of where the edit was stored. That scale is free to grow. Under a loss that does not see the length of a row the gradient is orthogonal to the row, so the length rises on its own (Arora et al., 2019; Heo et al., 2021). For TransE and TransH the length is a gauge, because the score normalizes the rows inside its own forward pass, and retraction only fixes that gauge. For RotatE the length enters the score, so retraction is a genuine constraint on the edit. KAU retracts on all three backbones, and Appendix D.3 reads the row without retraction beside the full method and shows that the result does not rest on norm inflation.

B.5 SCORE PRESERVATION

The retained pass carries the ordinary margin-ranking loss on retained edges and the second term of the objective of equation 9 in Section 4, $\beta\ \mathrm{mean}_i(d_i(\theta) - d_i(\theta^\circ))^2$, over retained edges incident to the involved entities and resampled at every step. Proposition A.2 identifies this term with the squared residual of the known channel. It is the first of the two controls of Section 4.4. Table 16 switches it off and Table 17 replaces it by a relational anchor, rows that Appendix D.3 reads beside the retraction row. A retained edge that the edit has pushed above its original score is an edge the update will pull back, and pulling it back drags the forgotten facts with it.

B.6 SELECTION

Once per epoch the run evaluates its current checkpoint and scores it by $0.7\,(1 - \mathrm{MRR}_{\mathcal{F}}) + 0.3\,\mathrm{MRR}_R$ on the development subsets of the forget set and of the retained set that the unlearner holds, disjoint from the evaluation subsets (Appendix C.1). The released model is the maximizer over the epochs of the run. This is GraphDPO's own rule applied unchanged, and it consults neither the retrained model nor any edge of the update. Its weight 0.7 is the value that Appendix C.2 selects for the checkpoint-rule weight. It chooses the checkpoint within a run. Hyperparameters are chosen before the run, by the criterion that every method shares (Appendix C.2).

B.7 THE FULL PROCEDURE

Algorithm 1 lists one run of KAU. During the run only the rows of $\mathcal{V}_{\mathcal{F}}$ are trainable, and the relation rows stay at their values in θ°. Each epoch makes one retained pass over batches $\mathcal{B}$ of retained edges at $\mathcal{V}_{\mathcal{F}}$, then J forgetting steps. The margin-ranking loss on $\mathcal{B}$ is $\ell_{\mathrm{MR}}(\mathcal{B})$, and $\ell_R(\theta; S)$ is the mean in equation 9 over a sample S redrawn at every step. The call $\textsc{Adam}(\theta, L)$ takes one step of the pass's own Adam optimizer on L. The call $\textsc{Retract}$ rescales each row of $\mathcal{V}_{\mathcal{F}}$ to its norm under θ° (Appendix B.4). The stop-gradient sg keeps the features out of the gradient, and CG solves equation 6 by conjugate gradients without forming the matrix. The checkpoint score $\textsc{Select}$ is the rule of Appendix B.6, and the stand-in pools of Appendix B.1 are rebuilt every five epochs. The epoch count E, the forgetting learning rate and the target fraction s are selected per backbone by the rule that every method shares (Appendix C.2).

Algorithm 1 KAU, one unlearning run

Require: original model θ°, forget set $\mathcal{F}$, retained edges $\mathcal{D}_R$, modelled class $\hat{\mathcal{N}}$, ridge scale σ_{rel}, weight β, target fraction s, epochs E, forgetting steps J
Ensure: released model θ_u

$\theta \leftarrow \theta^\circ$
$t_f \leftarrow d_f(\theta^\circ) + s\ \mathrm{relu}\left(q_f - d_f(\theta^\circ)\right)$ for $f \in \mathcal{F}$ ▷ (7)
for $e = 1$ **to** E **do**
  for each batch $\mathcal{B}$ of retained edges at $\mathcal{V}_{\mathcal{F}}$ **do**
    $S \leftarrow$ fresh sample of retained edges at $\mathcal{V}_{\mathcal{F}}$
    $\theta \leftarrow \textsc{Retract}\big(\textsc{Adam}(\theta,\ \ell_{\mathrm{MR}}(\mathcal{B}) + \beta\,\ell_R(\theta; S))\big)$ ▷ (9)
  for $j = 1$ **to** J **do**
    $N \sim \hat{\mathcal{N}}$ ▷ §4.1
    $\boldsymbol{\Phi} \leftarrow \mathrm{sg}\left[\partial d_N(\theta)/\partial \boldsymbol{e}_{\mathcal{V}_{\mathcal{F}}}\right], \quad r \leftarrow r_N(\theta)$ ▷ (4)
    $\boldsymbol{\Sigma} \leftarrow \mathrm{diag}(\sigma_v \mathbf{I}),\ \sigma_v = \sigma_{\mathrm{rel}}\ \mathrm{mean}_i\ \|\phi_i^{(v)}\|_2^2$
    $\Delta \leftarrow \mathrm{CG}\big(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{\Sigma},\ -\boldsymbol{\Phi}^\top r\big)$ ▷ (6), differentiable in r
    $\tilde{d}_f \leftarrow d_f(\theta + \Delta) - \mathrm{sg}\left[\mathrm{relu}\left(d_f(\theta + \Delta) - d_f(\theta)\right)\right]$ for $f \in \mathcal{F}$ ▷ App. B.3
    $\theta \leftarrow \textsc{Retract}\big(\textsc{Adam}\big(\theta,\ \frac{1}{|\mathcal{F}|}\sum_f \mathrm{relu}(t_f - \tilde{d}_f)^2\big)\big)$ ▷ (3)
  $\theta_e \leftarrow \theta, \quad c_e \leftarrow \textsc{Select}(\theta)$
return $\theta_u \leftarrow \theta_{e^\star}$ with $e^\star = \arg\max_e c_e$

B.8 BACKBONES AND WHERE THE FEATURES ARE TAKEN

The method needs a score and its gradients with respect to the two endpoint rows, so it applies to any backbone that keeps one entity table and one relation table and is differentiable almost everywhere in those rows. We report three, all with 200 latent coordinates per entity. TransE stores one real row of $m = 200$ coordinates per entity and translates normalized rows (Bordes et al., 2013). RotatE packs 200 complex coordinates into a real row of $m = 400$ and keeps one phase per coordinate in the relation row (Sun et al., 2019). Its rows are used without normalization, and its trained distances are about six times TransE's. TransH keeps entity rows of $m = 200$ and normalizes them as TransE does, packs the translation and the hyperplane normal into a relation row of 400 coordinates, and projects both endpoints onto the relation's hyperplane before translating (Wang et al., 2014). Section 3 gives the TransE score, and the other two follow the cited definitions.

The slot feature $\phi_x^{(v)} = \partial d_x / \partial \boldsymbol{e}_v$ is taken with respect to the raw parameter row, not with respect to whatever the model exposes as an embedding. Any transform the score applies to a row therefore sits inside the differentiated graph and reaches the feature through its own Jacobian. For a normalized row that Jacobian contributes a factor $(\mathbf{I} - \bar{\boldsymbol{e}}\bar{\boldsymbol{e}}^\top)/\|\boldsymbol{e}\|_2$, which removes the radial direction the score cannot see. For a packed complex row it contributes the map from the two real coordinates of a block to that block's modulus. The response is then a displacement of the same raw rows the optimizer updates, and the anticipated score is obtained by adding the displacement to those rows and calling the score function again, with no correction term anywhere. Porting the method to a new backbone requires the score function and nothing else.

B.9 HYPERPARAMETERS

Table 3 lists the settings of the unlearner, of the update and of the models. The unlearner's values are the same on the three backbones except for the forgetting learning rate, the epoch count and the target fraction s, which are selected per backbone (Appendix C.2). The retained-span projection of Table 18 shares this table and differs in two places. It applies the plain forgetting loss to the pre-update score, and it projects that loss's gradient off the span of the incident retained edges through a ridge-damped projector, with KAU's ancillaries.

B.10 COST

This subsection gives the computational cost of KAU and compares its time to release with that of the baselines. A forgetting step draws about ten edges per involved entity, recomputes their features in one backward pass, solves for the response over all involved rows at once, and scores every forgotten fact at the shifted rows. Each conjugate-gradient iteration costs one product with $\boldsymbol{\Phi}$ and one with its transpose, both linear in the number of drawn edges, so no matrix is ever formed. Start-up adds an incidence list per involved entity, the first stand-in pool, and one pass of 128 corrupted scores per forgotten fact for the targets. Every run reported in this paper uses one NVIDIA RTX PRO 5000 Blackwell GPU.

Table 4 gives the time to release one unlearned model on CoDEx-L with TransE, over unlearning seeds 21 to 27, with four CPU cores. KAU releases 5.4 times faster than GraphDPO and 18 times faster than retraining.

B.11 CONSTRUCTIONS THE DERIVATION ADMITS AND WE DO NOT USE

Three further constructions follow from the same derivation and are not part of the method. A second Gauss–Newton round re-linearizes the response at the shifted rows on the same draw and adds the correction to the first, which models the drift of the features during the update, and it improved no reported quantity. Letting the features carry gradient opens a channel in which the unlearner reshapes the update's features instead of the residual they act on, and it was worse, so the features carry no gradient within a step. A penalty that holds the stand-ins at their original scores leaves the unseen channel free of residual at release, and it costs utility without moving the leakage. A fourth item is a hypothesis. Adam's per-coordinate scaling suggests a ridge that falls with the number of update edges at an entity, as the inverse square root of its share of the steps (Remark A.4 in Appendix A.12), and every experiment reported here uses the degree-independent ridge.

Table 3: Hyperparameters on FB15k-237 (FB). The CoDEx-L settings are selected by the same rule (Appendix C.2) and listed in Appendix C.4.

Models	
Backbones	TransE, RotatE, TransH at 200 latent coordinates, Appendix B.8
Training	margin ranking with Adam (Kingma & Ba, 2015), loss $[d_x(\theta)-\bar{n}_x+\gamma]_+$ with $\bar{n}_x$ the mean score of $K = 10$ random corruptions of x and margin $\gamma = 8$ (Section 3)
Original and retrained models	one training run each per graph and backbone with that recipe, the original model on $\mathcal{D}_{\text{train}} \setminus B$, with B the never-trained set of Section 5, and the retrained model on $\mathcal{D}_R$ alone, never visible to the unlearner. Table 9 adds two further retrained models on FB15k-237, at retraining seeds 2 and 3
Update class assumed by the unlearner	
Known channel	each retained edge at v with probability $\rho_{\text{kn}} = 0.2$
Unseen channel	32 stand-ins per entity, best of four by plausibility under θ°, refreshed every 5 epochs, each entering with probability $\min\{1, \rho_{\text{un}} \deg(v)/32\}$, $\rho_{\text{un}} = 0.05$
Cap and draw	at most $k_{\max} = 96$ edges per entity, one fresh draw per forgetting step
Response, evaluation, composition	
Residual	$r_i = \text{relu}(d_i(\theta) - d_i(\theta^\circ))$ on every drawn edge
Ridge	$\sigma_v = \sigma_{\text{rel}} \cdot \text{mean}_i \, \|\phi_i^{(v)}\|_2^2$ with $\sigma_{\text{rel}} = 10^{-3}$
Solve	$(\mathbf{\Phi}^\top \mathbf{\Phi} + \mathbf{\Sigma})\Delta = -\mathbf{\Phi}^\top r$ over all involved rows by conjugate gradients (Hestenes & Stiefel, 1952), features recomputed at the current θ at every step and carrying no gradient
Evaluation	d_f at the shifted rows through the model's own score function
Composition	$\tilde{d}_f = \min\{d_f(\theta), d_f(\theta + \Delta\theta)\}$, the gradient routed through the anticipated path
Targets	$t_f = d_f(\theta^\circ) + s \, \text{relu}(q_f - d_f(\theta^\circ))$, $s = 0.15$ by default, selected as 0.15, 0.4 and 0.25 on TransE, TransH and RotatE, q_f the 5% quantile of θ°'s scores over 128 corrupted copies
Optimization and ancillaries	
Trainable rows	the entity rows of $\mathcal{V}_\mathcal{F}$, 11,412 of 14,541 on FB, relations frozen, released model fully trainable
Forgetting pass	2 steps per epoch, at Adam learning rate 3×10^{-5} on TransE and TransH and 3×10^{-4} on RotatE, following its score scale, for 20, 10 and 10 epochs on TransE, TransH and RotatE (Appendix C.2)
Retained pass	margin loss and score preservation over retained edges incident to $\mathcal{V}_\mathcal{F}$, Adam learning rate 10^{-4}, once per epoch. The score-preservation term has weight $\beta = 1$, the anchor weight of Appendix C.2, and 0.3 in the ablation tables of Section 5. It is taken over 4096 of those edges, resampled at every step
Retraction	every edited row rescaled to its original length after every step
Selection	$0.7\,(1-\text{MRR}_\mathcal{F})+0.3\,\text{MRR}_R$ on the development subsets the unlearner holds, once per epoch over the run
Update, and the published baselines	
Update	every parameter trainable, margin loss, Adam, on retained edges and never-trained edges, over the 48 update configurations of Section 5, shared by every method and by the oracle
GraphDPO (Liu et al., 2026)	the authors' implementation, under the same hyperparameter-selection criterion as KAU (Appendix C.2). It is tuned over its own ranges of Appendix C.2 and updates both the entity table and the relation table. Its feasible set therefore contains that of KAU, which edits only the involved entity rows under the restriction defined in Section 3
ZeroFisher (Xiao et al., 2025b)	the authors' implementation, tuned over its perturbation scale, update scale, number of iterations and, on RotatE, damping, under the same criterion. It edits the rows of the local neighbours of the forgotten edges (Section 2)

Table 4: Time to release one unlearned model on CoDEx-L with TransE, median over seven unlearning seeds with the range in brackets.

Method	Time to release (s)
Retraining (exact)	240.4 [240.2, 240.5]
GraphDPO	72.8 [72.7, 72.8]
KAU	13.4 [13.4, 13.4]

C Experimental details

Section 3 defines the update and the leakage, and Section 5 states the protocol in outline. This appendix gives the graphs and their splits, the edge set the update runs on, the budget each method receives, the seeds and tests behind every table, and the CoDEx-L results in full.

C.1 Training details

On FB15k-237 the training graph has 272,115 triples. The never-trained set B holds 6,000 of them (2.2%) and the forget set 27,211 (10%), both sampled uniformly at random with a fixed seed; the retained set holds the remaining 238,904. On CoDEx-L the training graph has 551,193 triples, B holds 12,000 (2.2%), the forget set 55,119 (10%) and the retained set 484,074. For each dataset and backbone, the original model θ° is trained on $\mathcal{D}_{\text{train}} \setminus B$, and the gold model θ_g is trained from scratch on $\mathcal{D}_R$, which also excludes the forget set. From each forget set a probe of 1,000 triples is drawn once and sealed; it serves only the relearning evaluation of Appendix C.3.

Pre-training. Original and gold models use the GraphDPO reference implementation of each backbone, with embedding dimension 200, a margin ranking loss with margin 8, ten negatives per positive, batch size 1,024, and Adam with learning rate 10^{-4} for 100 epochs. For each dataset and backbone, the two models use identical architecture, objective, optimisation settings and seed, differing only in their training triples. All unlearners share the original checkpoint and gold reference. Both models' training curves flatten well before the schedule ends (Figure 4). Every relearning quantity of Appendix C.3 is a difference against this gold model, so each is read relative to the reference rather than as an absolute level.

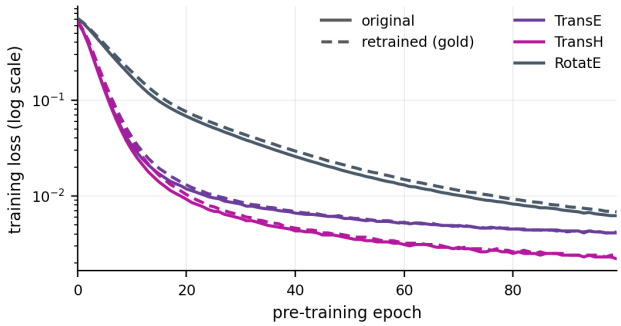


Figure 4: Pre-training loss on FB15k-237, at every epoch of the schedule. Solid lines are the original model and dashed lines the gold model. The pair of each backbone shares the recipe above and its seed, and differs only in the training triples. Both curves flatten well before the schedule ends, and every quantity we report is a difference between the two.

Evaluation. All MRR values are filtered and averaged over head and tail prediction. Forget-set MRR is computed on a held-out subset of the deleted triples that is never used for selecting hyperparameters or update configurations, retain-set MRR on a fixed probe of retained training triples, and test MRR on the standard test split.

C.2 UNLEARNING STAGE

Metrics. We evaluate forgetting and utility immediately after unlearning. **Forget MRR** (M_F) is computed on the evaluation subset of $\mathcal{F}$ and measures how much of the deleted knowledge remains. **Retain MRR** (M_R) is computed on the evaluation subset of $\mathcal{D}_R$ and measures preservation of knowledge that was not deleted. **Test MRR** (M_T) is computed on the test split and measures general link-prediction utility. The original and gold models provide reference values. An unlearned model should match the gold model on all three: a forget MRR above the gold model's indicates residual knowledge, and one below it indicates over-forgetting, i.e., the deleted triples are ranked worse than by a model that never observed them.

Hyperparameter selection. KAU and each baseline are tuned separately for every dataset–backbone setting, with the same criterion for all methods. Each configuration is scored by the larger of two shortfalls, one in forgetting and one in retained performance, each measured on the corresponding development subset, and the configuration with the smallest score is evaluated on disjoint held-out subsets, which produce all reported values. Selection and reporting use the same seven unlearning seeds.

For GraphDPO the search covers the learning rate $\{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$, the number of epochs $\{1, 2, 3, 6, 12, 24, 48\}$, the distillation weight $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$, the preference weight $\{0.03, 0.1, 0.3\}$ and temperature $\{1, 3\}$, the margin $\{4, 8\}$, and the policy and side of negative sampling, followed by a local refinement of the learning rate and distillation weight around the best setting. For ZeroFisher the search covers the perturbation scale $\epsilon \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, the update scale $\{0.05, 0.1, 0.25, 0.4, 0.5, 1, 10, 35\}$, the number of iterations from 3 to 1,550, and, on RotatE, damping in $\{0, -0.01, -0.02, -0.03, -0.04\}$, the ranges recommended by its authors. For KAU the forgetting learning rate $\{10^{-5}, 3\times10^{-5}, 10^{-4}, 3\times10^{-4}, 10^{-3}\}$, the number of epochs $\{2, 3, 5, 10, 20\}$ and the target fraction s of equation 7, $s \in \{0.005, 0.02, 0.05, 0.15, 0.25, 0.4, 0.6\}$, are searched jointly, and the number of forgetting steps per epoch $\{1, 2, 4\}$, the cap $k_{\max} \in \{48, 96, 160\}$ on drawn edges per entity, the inclusion probabilities of the two channels, $\rho_{\mathrm{kn}} \in \{0.05, 0.2, 0.5, 1\}$ and $\rho_{\mathrm{un}} \in \{0.01, 0.05, 0.3\}$, the relative ridge $\sigma_{\mathrm{rel}} \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, the score-preservation weight $\beta \in \{0.3, 1, 3\}$ of equation 9, called the anchor weight, and the weight of the checkpoint rule of Appendix B.6, $\{0.3, 0.5, 0.7, 0.95\}$, are varied one at a time around a reference setting. The selected KAU settings on FB15k-237 are, for TransE, TransH and RotatE respectively, epochs 20, 10 and 10, forgetting learning rate 3×10^{-5}, 3×10^{-5} and 3×10^{-4}, and $s = 0.15$, 0.4 and 0.25, with two forgetting steps per epoch, $k_{\max} = 96$, $\rho_{\mathrm{kn}} = 0.2$, $\rho_{\mathrm{un}} = 0.05$, $\sigma_{\mathrm{rel}} = 10^{-3}$, $\beta = 1$ and a checkpoint-rule weight of 0.7 shared across backbones.

Table 5: Unlearning results on FB15k-237 with 10% triple deletion. M_F, M_R and M_T denote filtered MRR on the forget, retain and test evaluation sets. Original and Gold are the shared reference checkpoints. Method rows report the mean over seven unlearning seeds; standard deviations are in Table 6. Bold marks the method closest to Gold on M_F and the highest on M_R and M_T.

	TransE			TransH			RotatE		
Method	M_F	$M_R\uparrow$	$M_T\uparrow$	M_F	$M_R\uparrow$	$M_T\uparrow$	M_F	$M_R\uparrow$	$M_T\uparrow$
Original	0.2733	0.2800	0.2137	0.2550	0.2596	0.2038	0.3075	0.3176	0.2214
Gold	0.2211	0.2932	0.2143	0.2097	0.2739	0.2051	0.2127	0.3301	0.2203
GraphDPO	0.2301	0.2628	0.2041	0.2121	0.2401	0.1927	0.2298	0.2531	0.1808
ZeroFisher	**0.2249**	0.2322	0.1857	0.2123	0.2164	0.1818	0.2180	0.2261	0.1693
KAU	0.2272	**0.2839**	**0.2137**	**0.2105**	**0.2614**	**0.2041**	**0.2112**	**0.3254**	**0.2213**

Seed variation is below 0.002 MRR for KAU and GraphDPO in every cell, whereas ZeroFisher varies by 0.02 to 0.04 MRR at fixed hyperparameters, so its mean scores average over runs that barely forget and runs that over-forget.

C.3 THE UPDATE

The update is the continued training of Section 5. It trains the released model further with the ordinary margin-ranking loss under Adam, with every parameter trainable. The update data are the

Table 6: Standard deviation over the seven unlearning seeds for every method cell of Table 5. Original and Gold are single checkpoints and have no seed variation. This table prints the population standard deviation over the seven seeds (divisor n). Table 7 and the tests of Table 8 use the sample standard deviation (divisor $n-1$).

	TransE			TransH			RotatE		
Method	M_F	M_R	M_T	M_F	M_R	M_T	M_F	M_R	M_T
GraphDPO	0.0009	0.0012	0.0008	0.0008	0.0013	0.0005	0.0011	0.0013	0.0013
ZeroFisher	0.0327	0.0320	0.0181	0.0340	0.0346	0.0212	0.0440	0.0445	0.0306
KAU	0.0006	0.0008	0.0004	0.0004	0.0005	0.0005	0.0004	0.0007	0.0002

retained triples together with the never-trained set B, and no forgotten triple is among them. The update's locality is defined against the endpoints of the probe: the 1,000 probe triples have 1,596 endpoints, and the two local components, the edges drawn from B and the retained neighbours, are incident to one of these rather than to one of the 11,412 involved entities; the third axis below adds retained edges incident to none of them. Every method and the retrained oracle receive the identical edge set, and the unlearner never sees the probe. Each dataset and backbone has 48 update configurations, shared by every method. The configurations vary four axes. The first is the number of edges drawn from the never-trained set B, in $\{1{,}000,\ 3{,}000,\ 6{,}000\}$. The second is the fraction of each probe endpoint's retained neighbours that the update trains on, in $\{0.15,\ 0.25,\ 0.35,\ 0.5\}$. The third is the fraction of the retained edges touching no probe endpoint, in $\{0,\ 0.02\}$. The fourth is the learning rate, in $\{10^{-4},\ 3\times10^{-4}\}$. Every configuration runs for 20 epochs with batch size 256 and ten negatives per positive. The grid has $3\times4\times2\times2=48$ configurations. The sample of update triples is drawn per seed and is not an axis. On CoDEx-L the counts of edges drawn from B are 2,000, 6,000 and 12,000, the same fractions of B as on FB15k-237, and the other axes are unchanged.

All configurations run with three unlearning seeds. The probe of 1,000 deleted triples is split once into a selection half and a report half. For each method, the configuration under which it leaks most over the run on the selection half is selected; the selected configurations are then re-run with seven unlearning seeds, and each method is reported under the strongest of them against it, at each run's epoch of peak leakage. The selection half serves only this selection, and every reported number is computed on the report half. The retrained oracle (the gold model of Section 5) receives every configuration with the same data and settings. Its runs are computed once per dataset, backbone, configuration and unlearning seed and cached. Every method is therefore read against the oracle run on the same update sample at the same epoch. Because the probe is a sample of the deleted triples, the original and gold levels on it differ from those of Table 5.

Relearning metrics. Let $\mathrm{MRR}_{\mathrm{u},e}$ and $\mathrm{MRR}_{\mathrm{gold},e}$ denote the forget-set MRR of the unlearned and gold models after e epochs of the update, with $e=0$ at release, and $\mathrm{MRR}_{\mathrm{orig}}$ that of the original model. The update restores a share of each model's unlearning-induced drop,

$$\mathrm{Rec}_{\mathrm{u}}(e)=100\times\frac{\mathrm{MRR}_{\mathrm{u},e}-\mathrm{MRR}_{\mathrm{u},0}}{\mathrm{MRR}_{\mathrm{orig}}-\mathrm{MRR}_{\mathrm{u},0}},\qquad \mathrm{Rec}_{\mathrm{gold}}(e)=100\times\frac{\mathrm{MRR}_{\mathrm{gold},e}-\mathrm{MRR}_{\mathrm{gold},0}}{\mathrm{MRR}_{\mathrm{orig}}-\mathrm{MRR}_{\mathrm{gold},0}},\quad(16)$$

where 100 is a return to the original model's forget-set MRR. The excess recovery, the leakage of equation 1 read at epoch e, and the relative excess recovery are

$$\mathrm{ExRec}(e)=\mathrm{Rec}_{\mathrm{u}}(e)-\mathrm{Rec}_{\mathrm{gold}}(e),\quad(17)$$

$$\mathrm{Leak}(e)=\big(\mathrm{MRR}_{\mathrm{u},e}-\mathrm{MRR}_{\mathrm{u},0}\big)-\big(\mathrm{MRR}_{\mathrm{gold},e}-\mathrm{MRR}_{\mathrm{gold},0}\big),\quad(18)$$

$$\mathrm{RER}(e)=100\times\frac{\mathrm{Leak}(e)}{\mathrm{MRR}_{\mathrm{orig}}-\mathrm{MRR}_{\mathrm{gold},0}}.\quad(19)$$

Each is reported at $e^*=\arg\max_e \mathrm{Leak}(e)$, and the gold model's terms are taken at the same e^* of the same update run, so they depend on the method under evaluation. RER divides Leak by one denominator per dataset and backbone, shared by every method. ExRec divides each model by its own deficit and therefore also reflects the difference in forgetting depth between the unlearned and the gold model.

C.4 BUDGETS AND SELECTION

Two selections are made. The first picks the hyperparameters of each method, and the second picks the released checkpoint within a run.

The hyperparameters are selected per method and per dataset–backbone setting by the shortfall criterion of Appendix C.2, with the same criterion for every method. The number of epochs and the learning rates are among the searched hyperparameters. GraphDPO searches its epoch count over $\{1, 2, 3, 6, 12, 24, 48\}$ and KAU over $\{2, 3, 5, 10, 20\}$, so each method runs for the number of epochs its own selection picks. The selected KAU settings on FB15k-237 are listed in Appendix C.2 and in Table 3. The selected settings of GraphDPO on FB15k-237 are, for TransE, TransH and RotatE respectively, learning rate 10^{-4}, 10^{-4} and 3×10^{-4}, epochs 3, 3 and 26, and distillation weight 3×10^{-3}, 6×10^{-3} and 3×10^{-4}. The preference weight 0.1, the temperature 1 and the margin 8 are shared by the three backbones. ZeroFisher uses perturbation scale $\epsilon = 10^{-3}$, update scale 10 and 10 iterations on TransE and TransH, and $\epsilon = 10^{-4}$, update scale 1.6 and 300 iterations on RotatE. On CoDEx-L the selected KAU settings are, for TransE, TransH and RotatE respectively, epochs 2, 10 and 20, forgetting learning rate 3×10^{-4}, 3×10^{-4} and 3×10^{-4}, and $s = 0.6$, 0.02 and 0.4, with the remaining settings as on FB15k-237. The selected settings of GraphDPO are learning rate 5×10^{-5}, 10^{-4} and 10^{-3}, epochs 48, 48 and 96, and distillation weight 3×10^{-3}, 3×10^{-3} and 10^{-3}, with preference weight 0.1, temperature 1 and margin 8. ZeroFisher uses perturbation scale $\epsilon = 10^{-5}$, update scale 0.5 and 50 iterations on TransE, $\epsilon = 10^{-4}$, update scale 10 and 75 iterations on TransH, and $\epsilon = 10^{-5}$, update scale 10, 350 iterations and damping -0.01 on RotatE.

The released checkpoint within a run is chosen by the rule of Appendix B.6. Once per epoch the run scores its current checkpoint by $0.7\,(1 - \mathrm{MRR}_{\mathcal{F}}) + 0.3\,\mathrm{MRR}_{R}$ on the development subsets the unlearner holds. The released model is the checkpoint with the highest score. The rule is applied unchanged to every method. It consults neither the oracle nor any edge of the update.

C.5 RELEARNING-RESISTANT ADD-ONS

Table 2 adds three relearning defences to GraphDPO on FB15k-237. Each add-on is built on GraphDPO at its selected setting on that backbone (Appendix C.4).

NTK. We apply the scrubbing step of Golatkar et al. (2020b) once to GraphDPO's released model. The step moves the parameters along the forgotten facts' score gradients, projected off the span of the retained triples' score gradients, with the step length of the authors' code. It then adds Gaussian noise with covariance αF^{-1}, where F is the diagonal Fisher on the retained triples. We search the authors' step with $\alpha \in \{0, 10^{-10}, 10^{-9}\}$ and twice and four times that step with $\alpha = 0$.

SAM. We follow Algorithm A1 of Fan et al. (2025b) at every GraphDPO step. GraphDPO's forgetting terms, its margin and preference losses on the forget batch, are evaluated at parameters perturbed by ρ along their normalized gradient, and its distillation and replay terms at the unperturbed parameters. We search $\rho \in \{0.03, 0.1, 0.3\}$ on TransE and $\rho \in \{0.003, 0.01, 0.03, 0.1, 0.3\}$ on TransH and RotatE, adding epochs where needed so that the update is as large as GraphDPO's. We select $\rho = 0.03$, 0.01 and 0.03 at 3, 3 and 27 epochs on TransE, TransH and RotatE.

TAR. We continue GraphDPO's released model with the adversarial training of Tamirisa et al. (2025) for 750 outer steps. Each outer step simulates one update at learning rate 10^{-4} or 3×10^{-4}, on retained edges around a batch of forgotten facts, with or without the stand-ins of Section 4.1. The tamper-resistance loss penalizes any rise of a forgotten fact's log-odds against its 50 strongest competing answers above its level in GraphDPO's release, both along the simulated update and at the current parameters, and is differentiated to first order as in the authors' code. A retained margin loss and an anchor to GraphDPO's release hold utility. With the tamper-resistance weight fixed at 0.2, we search the outer learning rate over $\{2, 4, 8, 16, 30\} \times 10^{-5}$ and select 2×10^{-5} on TransE and TransH and 8×10^{-5} on RotatE. This add-on differs from the TAR-style objective of Appendix D.7, which replaces KAU's forgetting step.

Selection and evaluation. For each add-on, the five settings are scored by the criterion of Appendix C.2, and the selected setting is kept at every seed. Every row of Table 2, GraphDPO and

KAU included, is run at three of the seven unlearning seeds of Table 1 and receives the same three to six update configurations of Appendix C.3 on each backbone. Each row is reported under the configuration in which it leaks most, at its epoch of peak leakage.

C.6 STATISTICS

Table 5 and Table 1 report means over seven unlearning seeds. Table 6 gives the standard deviations of Table 5 over those seeds, and Table 7 those of the FB15k-237 block of Table 1. The standard deviation of the leakage is 0.0021, 0.0019 and 0.0032 for KAU on TransE, TransH and RotatE, 0.0057, 0.0079 and 0.0078 for GraphDPO, and 0.0470, 0.0421 and 0.0311 for ZeroFisher. For ZeroFisher the excess recovery spreads far more than the leakage, because its release-time drop, the denominator of its excess recovery, varies across seeds. Table 1 prints the excess recovery as the ratio of seed means, and Table 7 gives the mean and spread of the per-seed ratios. Tables 2, 16, 17 and 18 and Figures 3, 5, 6 and 7 report means over three unlearning seeds. Table 13 reports means over the seven seeds of Table 1. Table 12 uses three screening seeds per configuration, and its last row uses the seven seeds of Table 1. Tables 14 and 15 use one unlearning seed. Figure 2 shows means over three unlearning seeds under one update configuration, with every method released at a forgetting rate of 90 to 106 percent of the gold model's. A seed varies the unlearner's own randomness and the sample of update triples, while the split, the original model and the update configurations stay fixed. Selection and reporting use the same seeds (Appendix C.2). Table 8 tests the leakage differences of Table 1, seven KAU seeds against seven baseline seeds, with Welch's t-test and the Mann–Whitney U test, both two-sided. On every backbone every KAU seed leaks less than every GraphDPO seed. Both tests find KAU's leakage below GraphDPO's at the 0.1% level and below ZeroFisher's at the 5% level on every backbone.

Table 7: Seed spread of Table 1 on FB15k-237: mean $\pm$ standard deviation over the seven unlearning seeds at each method's reported update configuration. ExRec is the mean of the per-seed ratios, whereas Table 1 prints the ratio of seed means.

Backbone	Method	ExRec (%)	Leak	RER (%)	M_R
TransE	KAU	48.91 ± 4.96	0.0215 ± 0.0021	37.45 ± 3.66	0.2839 ± 0.0008
	GraphDPO	102.13 ± 10.07	0.0437 ± 0.0057	76.12 ± 9.95	0.2628 ± 0.0013
	ZeroFisher	231.11 ± 222.71	0.0741 ± 0.0470	129.00 ± 81.77	0.2322 ± 0.0346
TransH	KAU	6.44 ± 2.05	0.0211 ± 0.0019	71.99 ± 6.48	0.2614 ± 0.0005
	GraphDPO	72.95 ± 23.00	0.0394 ± 0.0079	134.22 ± 26.94	0.2401 ± 0.0014
	ZeroFisher	343.97 ± 447.50	0.0687 ± 0.0421	234.02 ± 143.34	0.2164 ± 0.0374
RotatE	KAU	23.71 ± 2.47	0.0239 ± 0.0032	23.36 ± 3.13	0.3254 ± 0.0007
	GraphDPO	78.34 ± 8.47	0.0641 ± 0.0078	62.65 ± 7.66	0.2531 ± 0.0015
	ZeroFisher	175.23 ± 122.56	0.0857 ± 0.0311	83.75 ± 30.34	0.2261 ± 0.0481

Table 8: Significance of the leakage differences in Table 1 on FB15k-237: KAU against each baseline, seven unlearning seeds against seven, at the reported update configurations. Welch's t-test and the Mann–Whitney U test, both two-sided.

Backbone	Comparison	ΔLeak	Welch t	p	U	p
TransE	KAU vs. GraphDPO	-0.0222	-9.67	$< 10^{-4}$	0	0.0006
	KAU vs. ZeroFisher	-0.0526	-2.96	0.025	7	0.026
TransH	KAU vs. GraphDPO	-0.0183	-5.96	0.0007	0	0.0006
	KAU vs. ZeroFisher	-0.0475	-2.98	0.025	7	0.026
RotatE	KAU vs. GraphDPO	-0.0402	-12.62	$< 10^{-4}$	0	0.0006
	KAU vs. ZeroFisher	-0.0618	-5.23	0.0018	2	0.0023

Section 3 places the retrained oracle at zero leakage, so the width of that zero is the oracle's own variation across retraining seeds. Table 9 measures it. We retrain the oracle twice more per backbone with the recipe of Appendix C.1, under retraining seeds 2 and 3. Each retrained oracle receives the update of Table 1 at every method's reported update configuration for the seven unlearning seeds.

Every run is read at the same per-seed peak epoch as Table 1. The table reports the oracle's own rise in forget-MRR under the update at the three oracle seeds. It also reports each method's leakage recomputed against each oracle seed and each method's own rise over its seven seeds. The oracle's rise varies across retraining seeds by a standard deviation of 0.0036 to 0.0043 on TransE, 0.0032 to 0.0042 on TransH and 0.0018 to 0.0031 on RotatE. Its retained MRR at release varies by 0.0017, 0.0024 and 0.0025. Recomputed against each oracle seed, KAU's leakage stays the smallest of the three methods on every backbone, and its reduction over GraphDPO stays between 44% and 67%. The reductions of Table 1 hold whichever oracle seed is the reference. KAU's leakage in Table 1 is 5.0, 5.0 and 7.7 times the largest oracle-seed standard deviation on TransE, TransH and RotatE. It is a rise beyond the oracle and not seed variation of the reference. In the last column, every method's rise exceeds the oracle's at every oracle seed on every backbone.

Table 9: The retrained oracle at three retraining seeds on FB15k-237. Each oracle receives the update of Table 1 at every method's reported update configuration for the seven unlearning seeds and is read at the same peak epochs. Oracle rise is the oracle's own gain in forget-MRR under the update, mean over the seven update samples, at oracle seeds 1, 2 and 3. Leak is each method's leakage against each oracle seed. Method rise is the method's own gain, mean $\pm$ standard deviation over the seven seeds. The per-backbone lines give the oracle's M_R at release at the three seeds and the leakage reduction of KAU over GraphDPO against each seed.

Backbone	Method	Oracle rise (seeds 1/2/3)	Leak vs. seeds 1/2/3	Method rise
TransE	GraphDPO	0.0351/0.0342/0.0417	0.0437/0.0447/0.0372	0.0789 $\pm$ 0.0024
	ZeroFisher	0.0192/0.0165/0.0235	0.0741/0.0768/0.0698	0.0933 $\pm$ 0.0499
	KAU	0.0419/0.0385/0.0471	0.0215/0.0249/0.0163	0.0634 $\pm$ 0.0049
	Oracle M_R at release 0.2932/0.2899/0.2906			
	Leakage reduction of KAU over GraphDPO 50.8/44.4/56.0%			
TransH	GraphDPO	0.0303/0.0365/0.0344	0.0394/0.0332/0.0353	0.0697 $\pm$ 0.0137
	ZeroFisher	0.0217/0.0290/0.0271	0.0687/0.0614/0.0632	0.0904 $\pm$ 0.0362
	KAU	0.0387/0.0471/0.0431	0.0211/0.0127/0.0167	0.0598 $\pm$ 0.0084
	Oracle M_R at release 0.2739/0.2695/0.2703			
	Leakage reduction of KAU over GraphDPO 46.4/61.7/52.5%			
RotatE	GraphDPO	0.0308/0.0319/0.0366	0.0641/0.0629/0.0583	0.0949 $\pm$ 0.0099
	ZeroFisher	0.0151/0.0126/0.0160	0.0857/0.0882/0.0848	0.1008 $\pm$ 0.0257
	KAU	0.0414/0.0424/0.0464	0.0239/0.0229/0.0190	0.0653 $\pm$ 0.0058
	Oracle M_R at release 0.3301/0.3278/0.3328			
	Leakage reduction of KAU over GraphDPO 62.7/63.6/67.4%			

C.7 CoDEx-L

CoDEx-L is run on TransE, TransH and RotatE under the protocol of this appendix, with the splits of Appendix C.1 and the update grid of Appendix C.3. Table 10 gives link prediction at release, with Gold the gold model of Section 5. Table 11 gives the seed spread and the tests behind the CoDEx-L block of Table 1.

The ordering of Table 1 holds on a second graph with over five times the entities and under a third of the relations, on all three backbones. KAU attains the lowest excess recovery, leakage and relative excess recovery on every backbone. Its leakage is 59%, 50% and 82% below GraphDPO's on TransE, TransH and RotatE. Its retained MRR exceeds GraphDPO's by 0.032, 0.044 and 0.103. On TransE it is 0.018 below the gold model's, where GraphDPO and ZeroFisher lose 0.050 and 0.092, and KAU's forget-MRR at release is within 0.002 of the gold model's. On TransE all three methods are read at one update configuration, the one that leaks most against each of them on the selection half. On every backbone every KAU seed leaks less than every GraphDPO seed. Both tests find KAU's leakage below ZeroFisher's at the 5% level on every backbone.

D Additional results

This appendix holds the measurements that the main text uses in summary. The first is the leakage across the 48 update configurations, and the second reads the update built from never-trained edges alone. The next two attribute the TransE result between the forgetting term and the ancillaries, and the recovery between parameter blocks. The next two are the calibration of the anticipated response as a ranking and the relation of forgetting depth to leakage and retained utility. The definitions and

Table 10: Link prediction at release on CoDEx-L with TransE, TransH and RotatE: filtered MRR on the forget, retain and test evaluation sets. Original and Gold are single checkpoints. Method rows are mean ± standard deviation over seven unlearning seeds. Bold marks the method closest to Gold on M_F and the highest on M_R and M_T.

Backbone	Method	M_F	$M_R \uparrow$	$M_T \uparrow$
TransE	Original	0.2354	0.2462	0.1501
	Gold	0.1487	0.2655	0.1492
	GraphDPO	0.1630 ± 0.0004	0.2154 ± 0.0013	0.1389 ± 0.0006
	ZeroFisher	0.1655 ± 0.0655	0.1734 ± 0.0683	0.1081 ± 0.0404
	KAU	**0.1504** ± 0.0002	**0.2474** ± 0.0005	**0.1466** ± 0.0002
TransH	Original	0.2412	0.2535	0.1505
	Gold	0.1509	0.2704	0.1534
	GraphDPO	0.1609 ± 0.0011	0.2091 ± 0.0008	0.1334 ± 0.0011
	ZeroFisher	0.1446 ± 0.0607	0.1515 ± 0.0641	0.0956 ± 0.0355
	KAU	**0.1475** ± 0.0004	**0.2530** ± 0.0008	**0.1461** ± 0.0003
RotatE	Original	0.2783	0.2916	0.1557
	Gold	0.1554	0.3235	0.1574
	GraphDPO	0.1900 ± 0.0011	0.2019 ± 0.0010	0.1041 ± 0.0007
	ZeroFisher	0.1586 ± 0.0532	0.1666 ± 0.0563	0.1006 ± 0.0324
	KAU	**0.1561** ± 0.0002	**0.3051** ± 0.0004	**0.1556** ± 0.0001

Table 11: Seed spread and significance of the CoDEx-L block of Table 1. Cells give mean ± standard deviation over seven unlearning seeds at each method's reported update configuration. The test lines give two-sided Welch t and Mann–Whitney U tests of KAU against each baseline on the leakage. ExRec is the mean of the per-seed ratios.

Backbone	Method	ExRec (%)	Leak	RER (%)	M_R
TransE	GraphDPO	77.35 ± 3.60	0.0524 ± 0.0031	31.30 ± 1.83	0.2154 ± 0.0013
	ZeroFisher	218.34 ± 122.10	0.1080 ± 0.0844	64.49 ± 50.37	0.1734 ± 0.0683
	KAU	26.94 ± 1.60	0.0214 ± 0.0012	12.80 ± 0.71	0.2474 ± 0.0005
Leak, KAU vs. GraphDPO: Welch $t = -24.93$, $p < 10^{-4}$, and $U = 0$, $p = 0.0006$					
Leak, KAU vs. ZeroFisher: Welch $t = -2.71$, $p = 0.035$, and $U = 7$, $p = 0.026$					
TransH	GraphDPO	75.61 ± 4.80	0.0621 ± 0.0034	39.41 ± 2.15	0.2091 ± 0.0008
	ZeroFisher	168.63 ± 65.18	0.1438 ± 0.0438	91.25 ± 27.80	0.1515 ± 0.0641
	KAU	27.10 ± 1.57	0.0310 ± 0.0034	19.68 ± 2.14	0.2530 ± 0.0008
Leak, KAU vs. GraphDPO: Welch $t = -17.22$, $p < 10^{-4}$, and $U = 0$, $p = 0.0006$					
Leak, KAU vs. ZeroFisher: Welch $t = -6.79$, $p = 0.0005$, and $U = 0$, $p = 0.0006$					
RotatE	GraphDPO	108.82 ± 6.45	0.1000 ± 0.0063	53.66 ± 3.37	0.2019 ± 0.0010
	ZeroFisher	192.86 ± 133.16	0.1328 ± 0.0210	71.29 ± 11.26	0.1666 ± 0.0563
	KAU	24.76 ± 4.30	0.0179 ± 0.0052	9.62 ± 2.79	0.3051 ± 0.0004
Leak, KAU vs. GraphDPO: Welch $t = -26.63$, $p < 10^{-4}$, and $U = 0$, $p = 0.0006$					
Leak, KAU vs. ZeroFisher: Welch $t = -14.06$, $p < 10^{-4}$, and $U = 0$, $p = 0.0006$					

settings behind the ablation tables of Section 5 follow, and a note on evaluating the anticipated score at the shifted rows closes the appendix. Leakage is Leak of equation 18, in MRR points, and RER of equation 19 is leakage as a share of the erased knowledge.

D.1 Leakage across update configurations

Section 5 runs 48 update configurations per dataset and backbone, shared by every method. Each configuration is one draw of the update of Section 3, the continued training of Section 5, run on retained triples together with the never-trained triples B. The retrained oracle, the gold model of Section 5, receives every configuration with identical data and settings. Table 1 reports each method

under the configuration that leaks most against it on the selection half of the probe, read on the report half at the epoch of peak leakage. Table 12 shows what lies behind that selected cell on TransE. The grid is that of Appendix C.3, and every configuration runs at three screening seeds. A configuration's leakage is the mean over those seeds of the leakage at each seed's peak epoch. Each cell of the table is the largest such value over the configurations of its group. The last row gives each method under its own configuration in Table 1, over seven unlearning seeds. It can therefore lie outside the three-seed range of the rows above it, as it does for ZeroFisher, whose seed spread is the largest (Table 7).

Table 12: Leakage on FB15k-237 with TransE by group of update configurations. Each cell is the largest leakage over the configurations of the group, where a configuration's leakage is the mean over the three screening seeds of the leakage at that seed's peak epoch. The last column is the ratio of KAU's cell to GraphDPO's. The last row gives each method under its own configuration in Table 1, over seven seeds.

Axis	Value	GraphDPO	ZeroFisher	KAU	KAU/GraphDPO
$\|B\|$ drawn	1,000	0.0467	0.0469	0.0248	0.53
	3,000	0.0462	0.0475	0.0247	0.53
	6,000	0.0470	0.0440	0.0249	0.53
neighbour fraction	0.15	0.0467	0.0471	0.0249	0.53
	0.25	0.0456	0.0421	0.0215	0.47
	0.35	0.0470	0.0475	0.0248	0.53
	0.5	0.0462	0.0401	0.0247	0.53
random retained fraction	0	0.0470	0.0475	0.0248	0.53
	0.02	0.0425	0.0395	0.0249	0.58
learning rate	10^{-4}	0.0470	0.0440	0.0247	0.53
	3×10^{-4}	0.0462	0.0475	0.0249	0.54
All 48		0.0470	0.0475	0.0249	0.53
Reported (7 seeds)		0.0437	0.0741	0.0215	0.49

The table is read down each column. KAU leaks less than GraphDPO and less than ZeroFisher in every one of the 48 configurations. Its leakage is 0.47 to 0.58 of GraphDPO's in every group. Leakage is flat along every axis. GraphDPO's group maxima span 0.0425 to 0.0470, and KAU's span 0.0215 to 0.0249. The worst configuration therefore sets the scale, and no axis moves KAU from the bottom of the ordering. KAU's place below both baselines in Table 1 is not a property of one configuration. The configuration reported in Table 1 is chosen by the selection of Appendix C.3, which reads leakage over the run on the selection half, whereas this table reads peak leakage on the report half; the two readings need not pick the same configuration, and under either KAU is lowest in every group.

D.2 The never-trained-edge update

Table 13 reads one update outside the 48 configurations on FB15k-237 with TransE, beside the standard update, over the seven unlearning seeds of Table 1. Each update is applied to the released checkpoint of every method and to the retrained oracle. ExRec, Leak and RER are read at the epoch of peak leakage as in Table 1. The standard update is the configuration of Table 14. It draws 3,000 edges from the never-trained set B and 25 percent of the retained neighbours of each probe endpoint, 40,021 edges in all, at learning rate 10^{-4} for 20 epochs. The never-trained-only update holds the 6,000 edges of B and nothing else, under the same learning rate and epoch budget.

The never-trained-only update holds only edges that no model has ever trained on. This is the channel the unlearner can model only through stand-ins (Section 4.1). Under it GraphDPO keeps 68 percent of its leakage and ZeroFisher 79 percent. Removing the retained channel therefore does not remove the baselines' leakage. The residual they leave on the involved rows is restored by any true edge at those rows, as Section 4.4 says. KAU keeps 42 percent of its leakage. Its leakage is 0.33 of GraphDPO's under this update, against 0.54 under the standard one. This is what the stand-in channel of Section 4.1 predicts, together with the observation of Section 4.6 that the retained span misses these directions. At the unlearning seed of Table 14, the retrained oracle's own rise ranges

Table 13: The never-trained-edge update on FB15k-237 with TransE, means over seven unlearning seeds. Each row applies one update to the released checkpoint of every method and to the retrained oracle. ExRec, Leak and RER are read at the epoch of peak leakage as in Table 1. The standard update is the configuration of Table 14, and the other lies outside the grid. Columns follow equations 17–19.

	GraphDPO			ZeroFisher			KAU		
Update	ExRec↓	Leak↓	RER↓	ExRec↓	Leak↓	RER↓	ExRec↓	Leak↓	RER↓
standard	95.26	0.0402	69.93	132.11	0.0697	121.35	48.53	0.0215	37.45
never-trained edges only	61.12	0.0274	47.75	104.15	0.0553	96.30	19.36	0.0089	15.55

from $+0.022$ to $+0.041$ under the standard update. Under the never-trained-only update it falls to a range of $+0.011$ to $+0.015$. The ratio between the methods is the reading.

Running the standard update for longer changes the leakage little. Over 40 epochs, KAU's leakage is 0.0217 against 0.0215 at 20 epochs, and GraphDPO's is 0.0412 against 0.0402. ZeroFisher's is unchanged at 0.0697. The peak-epoch reading of Section 5 therefore captures nearly all the leakage a longer update produces, and KAU still leaks about half as much as GraphDPO.

D.3 The forgetting term against the ancillaries

KAU runs its anticipated forgetting loss with two ancillaries, the score-preservation term of equation 9 and the norm retraction of Appendix B.4. The TransE result therefore has to be attributed among the three. The rows of Tables 16, 18 and 17 that change one of them at a time settle the attribution. Every such row except the relational anchor and the row without a forgetting term is released at KAU's forgetting depth, so the rows compare leakage at equal forgetting. Section 4.4 predicts the pattern they show. Its first control is the residual the edit leaves on the known edges, which Proposition A.2 ties to the score-preservation term. Without that term the update pulls the raised retained edges back and drags the forgotten facts with them. The row without a score anchor in Table 16 shows exactly this. Its leakage rises by almost half and its retained utility falls by 0.007, the lowest utility among KAU's variants. The relational anchor of Table 17 holds pairwise distances rather than scores, so it leaves the residual free. It leaks the most of all KAU's variants except the rows without the anticipated response or with a damped response, although it forgets the deepest. What matters on the known channel is therefore the score residual and not the geometry of the rows, as the first control says. The second control is where the forgetting is placed. The plain forgetting loss of Table 18, which is KAU without the anticipated response, leaks more than the anticipated loss at the same depth. The anticipated term therefore lowers leakage on its own and not through a deeper release.

The same rows test the two statements about the filter. Section 4.6 reads subspace methods as hard thresholds of the survival filter, and Corollary A.2 places null-space projection at its zero threshold. The retained-span projection of Table 18, the hard threshold without stand-ins, leaks more than KAU at the same depth. Adding the projector on top of the anticipated loss, the projector row of Table 16, raises leakage instead of lowering it. The hard threshold therefore adds nothing that the soft weights of the filter do not already provide, and it removes directions that the update would have left alone (Remark A.1). Section 4.5 keeps the norm retraction because norm inflation could mimic resistance on TransE and TransH (Appendix B.4). The row without retraction in Table 17 leaks almost the same as the full method at the same utility, so the low leakage does not rest on norm control. The anticipated term is what lowers leakage. The score anchor holds utility while that term acts, and the retraction is a safeguard rather than a source of the result. One reading note applies at unequal depth. The excess recovery of the relational anchor barely moves, because a deeper release enlarges the denominator of equation 16. Leakage in MRR points is therefore the column to compare there.

D.4 Which parameters the update restores

The unlearner edits only the rows of the involved entities, and the update restores the forgotten facts through the same rows. To read off which block carries the recovery we build hybrid models. A hybrid model takes one parameter block from the updated model and every other block from the

released model, and the forget-MRR is evaluated on it. Table 14 gives the result on TransE for KAU, GraphDPO, ZeroFisher and the retrained oracle. All four face one update configuration with one unlearning seed, and each model is read at its own epoch of peak leakage. Forget-MRR is measured on the 500-edge report half of the held-out forget probe. The blocks are the involved rows, the relation rows and the entity rows the edit never touched. The involved rows are the endpoint rows of the forgotten facts, 11,412 of the 14,541 entities. The involved rows alone reproduce 95 to 105 percent of every model's full rise. The relation rows reproduce at most 5.4 percent of the rise for the three unlearners and 9.5 percent for the oracle. The other entity rows move the forget-MRR by at most 0.0014. Each hybrid replaces one block at a time, so the three rises need not add up to the full rise, and two rows exceed 100 percent. Over the seven seeds of Table 1, the involved rows reproduce 94 to 105 percent of the rise for KAU, GraphDPO and the oracle, and 91 to 105 for ZeroFisher. A forgotten fact therefore relearns through its own two endpoint rows, for an edited model and for the retrained oracle alike. This is the premise of Section 3 and of the anticipated response.

Table 14: Block attribution of the rise in forget-MRR under the update, FB15k-237 with TransE, one update configuration and one unlearning seed. Released is the forget-MRR of the released model. Each further column gives the rise in forget-MRR of the hybrid that takes that block alone from the updated model, and the every-parameter column gives the rise of the updated model itself. Each model is read at its own epoch of peak leakage.

Model	Released	Every parameter	Involved rows	Relation rows	Other entity rows
KAU	0.2265	$+0.0580$	$+0.0553$	$+0.0002$	$+0.0005$
GraphDPO	0.2292	$+0.0875$	$+0.0845$	$+0.0012$	$+0.0014$
ZeroFisher	0.2527	$+0.0632$	$+0.0635$	$+0.0034$	-0.0011
retrained oracle	0.2240	$+0.0381$	$+0.0400$	$+0.0036$	$+0.0007$

D.5 The anticipated response as a per-fact ranking

The objective uses the response to order the forgotten facts by how much the update would restore them, so we evaluate it as a ranking. We take KAU's released TransE checkpoint and its updated checkpoint at the epoch of peak leakage under one update configuration and one unlearning seed. The predicted change of a forgotten fact's score is the response at the released checkpoint, averaged over 32 draws. The realized change is the fact's score at release minus its score after the update, so a positive change is a recovery. Table 15 compares the two over the $n = 1{,}000$ held-out forgotten facts for the three ways of computing the response.

Table 15: The anticipated response as a per-fact ranking on KAU's released TransE checkpoint on FB15k-237, under one update configuration and one unlearning seed. Per-fact correlation between the predicted and the realized change of the forgotten facts' scores under the update, at the epoch of peak leakage, for the three ways of computing the response. The last two columns give the mean predicted and the mean realized change, a positive change being a recovery. $n = 1{,}000$ facts, response averaged over 32 draws, 95 percent bootstrap intervals in brackets. The last line gives the mean realized recovery within each quintile of predicted recovery.

Response	Pearson	Spearman	Mean predicted	Mean realized
joint solve, exact evaluation (KAU)	0.526 [0.294, 0.674]	0.207 [0.145, 0.266]	-0.009	$+0.159$
per-entity block solve	0.303 [0.167, 0.422]	0.197 [0.135, 0.259]	-0.308	$+0.159$
first-order evaluation	0.605 [0.333, 0.746]	0.125 [0.061, 0.188]	$+0.145$	$+0.159$
Realized recovery by quintile of prediction (joint solve): $+0.003$, $+0.052$, $+0.085$, $+0.196$, $+0.459$				

The joint response with exact evaluation reaches a Pearson correlation of 0.53 and a Spearman correlation of 0.21 per fact. Its quintiles of predicted recovery are monotone in realized recovery, from $+0.003$ in the lowest quintile to $+0.459$ in the highest. This is the ranking property the objective uses. The per-entity block solve lowers the Pearson correlation to 0.30. A fact whose two endpoints are both edited therefore has to be solved once, jointly, for the prediction to be right. First-order evaluation ranks the facts worse, at a Spearman correlation of 0.13, although its Pearson correlation of 0.61 is higher. Over the seven unlearning seeds of Table 1, the joint response has median Pearson

and Spearman correlations of 0.50 and 0.23. At every one of those seeds it has a higher Pearson correlation than the per-entity block solve and a higher Spearman correlation than first-order evaluation. First-order evaluation matches the level of the change, with a mean predicted change of $+0.145$ against a realized $+0.159$. The exactly evaluated response predicts the ranking and not the level, with a mean predicted change of -0.009. The target t_f therefore supplies the level, as Section 4.3 states, and the response supplies the ranking. Table 16 places the joint solve with exact evaluation first in leakage as well. First-order evaluation raises RER from 19.38 to 29.07, and the per-entity block solve raises it to 30.50.

D.6 Leakage and retained utility against forgetting depth

Leakage depends on how deeply a method forgets at release, so Figure 3 in Section 5 plots RER against forgetting depth. Depth is the forgetting rate FR, the forget-MRR drop at release as a percentage of the retrained oracle's. Each method is released at three depths by varying only its budget knob, under one update configuration per backbone. Figure 3 shows means over three unlearning seeds. The rung nearest each of the three bands, 40 to 60, 60 to 85 and 85 to 110 percent, is shown. On TransE, from the shallowest to the deepest band the mean RER rises by about 59 points for ZeroFisher, 50 for GraphDPO and 15 for KAU. In the deepest band it is about 131 for ZeroFisher, 77 for GraphDPO and 26 for KAU. The mean ordering of the three methods holds in every band on every backbone. The baselines' RER grows with depth more than twice as fast as KAU's, so the gap widens toward the deepest band.

The figure is the reason every row of Tables 16 and 18 is released at KAU's forgetting depth. A method released deeper earns part of its leakage from that depth alone, so leakage is compared at equal FR.

Forgetting deeper does not cost KAU retained utility. Figure 5 gives the retained MRR at release of the same releases as Figure 3, as a percentage of the gold model's. KAU keeps more than 94 percent of the gold model's retained MRR in every band on every backbone, and its retention rises slightly with depth. The baselines pay for depth in retained utility. From the shallowest to the deepest band, GraphDPO, the stronger baseline, loses 4.7, 5.1 and 13.6 points on TransE, TransH and RotatE and ends at 89.8, 88.0 and 70.8 percent, and ZeroFisher loses 6.7, 6.7 and 13.8 points. At every depth KAU therefore both leaks less and retains more.

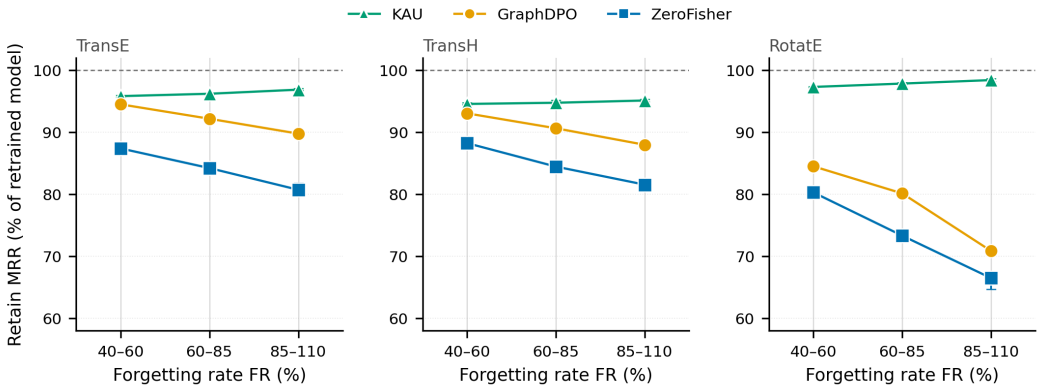


Figure 5: Retained MRR at release against forgetting depth on FB15k-237, as a percentage of the gold model's retained MRR, for the releases of Figure 3. Points average three unlearning seeds and bars span the seeds; the dashed line at 100 is the gold model. KAU keeps more than 94% of the gold model's retained MRR at every depth, whereas both baselines lose retained utility as they forget deeper.

Figure 6 follows the relearning epoch by epoch on RotatE, under the update configuration of Figure 3 and at three unlearning seeds. ZeroFisher's RER peaks within the first five epochs, and GraphDPO's keeps rising until the last epochs. In every band and at every seed, KAU's peak RER stays below 15, whereas the lowest peak of either baseline is 36.

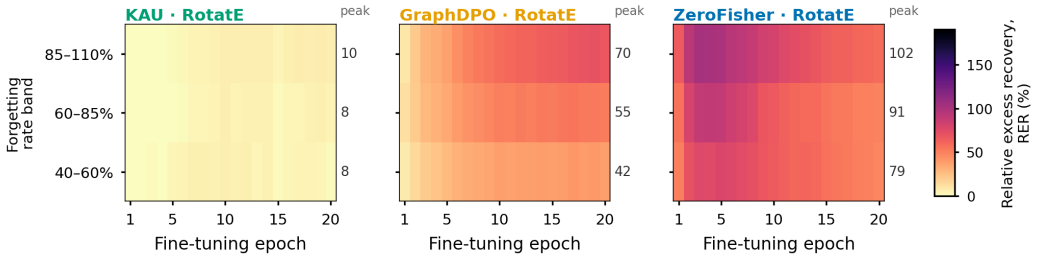


Figure 6: RER along the update on FB15k-237 with RotatE, by forgetting-rate band at release, under the update configuration of Figure 3. Each cell is the mean RER over three unlearning seeds at that epoch, and the number after each row is the mean over seeds of each seed's peak RER. An RER of zero is the retrained oracle's own recovery. The three panels share one colour scale.

D.7 Ablation details

Tables 16 and 18 are run in one ablation setting. Every variant starts from KAU's forgetting budget and keeps the retained pass, per-fact targets and release rule at KAU's selected TransE hyperparameters. The score-preservation weight is $\beta = 0.3$, the ablation setting of Table 3. All rows, including KAU, are released within a fixed band of forgetting rates, so that leakage is compared at equal forgetting, which is why KAU's values in these tables differ from those of Table 1. All rows face one update configuration of Table 1 with three unlearning seeds and are read at the epoch of peak leakage on the report half. A variant that forgets less than KAU at its budget is given a larger forgetting budget, set from its forgetting rate at release alone, so that it falls within the band. Figure 7 shows the leakage of every row of Tables 16 and 18 with its spread over the three seeds.

Table 16: KAU ablations on FB15k-237 with TransE, means over three unlearning seeds. Figure 7 gives both tables' seed spreads. Each row changes one component. All rows, including KAU, are released within a fixed forgetting-rate band, so KAU's values differ from Table 1.

Variant	ExRec↓	Leak↓	RER↓
KAU (full)	31.91	0.0111	19.38
no anticipated response (Δ=0)	48.60	0.0250	43.53
first-order evaluation	37.40	0.0167	29.07
per-entity block solve	42.24	0.0175	30.50
signed residual	30.78	0.0138	24.05
no unseen channel (ρ_{un}=0)	37.55	0.0151	26.29
no known channel (ρ_{kn}=0)	36.16	0.0145	25.28
projector added on top	31.20	0.0152	26.53
no score anchor	36.57	0.0161	28.06

Variants of KAU. *No anticipated response* sets the response to zero, so that the forgetting loss acts on the current score. *Response damped* raises the relative ridge from 10^{-3} to 10^{3}, which shrinks the response to a small fraction of its size. *First-order evaluation* replaces the score at the shifted rows by its linearisation around the current rows. *Per-entity block solve* solves the response separately for each entity's block instead of jointly. It therefore counts the residual of a drawn edge between two involved entities once at each endpoint (Lemma A.4). *Signed residual* takes the residual of every drawn edge with its sign, crediting the update with restoring lowered edges as well as raised ones. *No pessimistic composition* applies the loss to the anticipated score alone rather than to the smaller of the current and anticipated scores. *Per-relation gate* scales each edge's residual by its margin-violation probability; it binds on 4% of the modelled update edges. *No unseen channel* and *no known channel* remove, respectively, the stand-in edges and the retained neighbours from the draw. *No best-of-Q stand-in choice* keeps the first plausible stand-in candidate instead of the best of four. *Hinge of the expected draw* applies the loss to the response of the expected draw, with inclusion probabilities as weights, instead of averaging the loss over sampled draws. *Ordered $k_{\max}$ cut* keeps

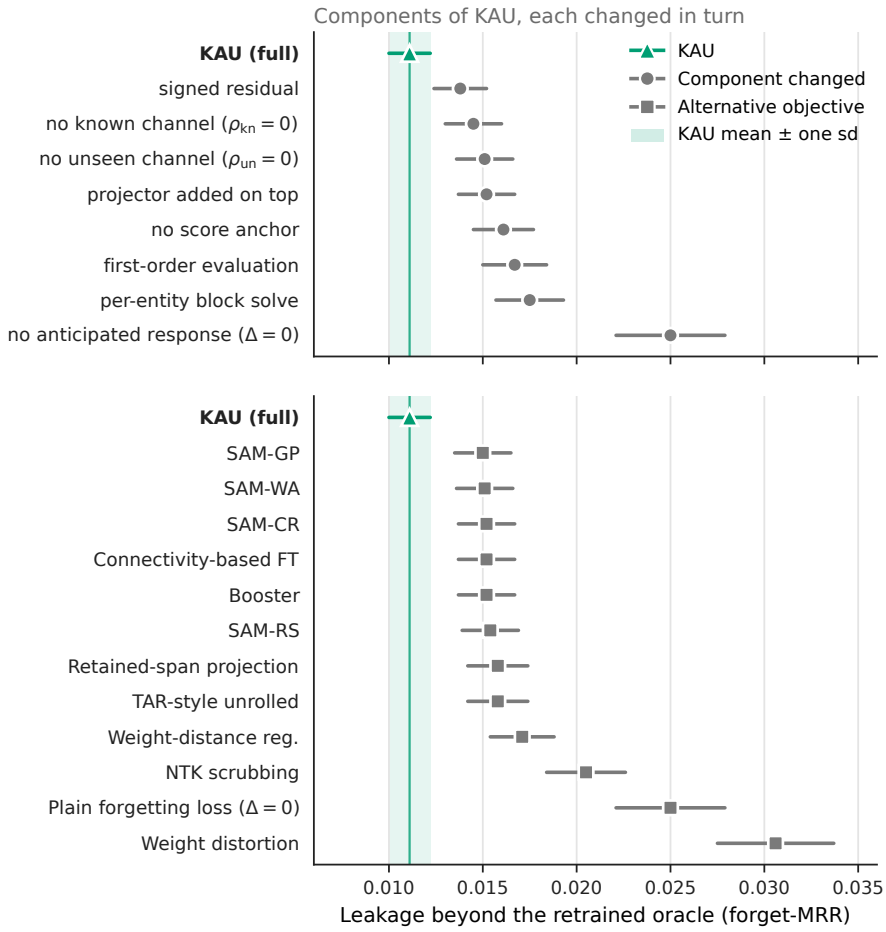


Figure 7: Leakage of every row of Tables 16 and 18 on FB15k-237 with TransE, in the ablation setting of those tables. The upper panel holds the rows of Table 16 and the lower panel those of Table 18. Markers are means over three unlearning seeds, and whiskers span one standard deviation over the seeds. The shaded band is KAU's mean plus or minus one standard deviation. Rows are sorted by leakage within each table.

the first $k_{\max}$ candidate edges in a fixed order instead of a random subset. The rows $\rho_{\mathrm{kn}} = 1$ and $k_{\max} = 16$ draw every retained edge and cap the draw at 16 edges, respectively. These two and the Q=1 row also draw fewer stand-in edges. *No score anchor* sets the score-preservation weight to zero. *Relational anchor* holds pairwise distances between rows instead of scores; it is released deeper than the gold model. *No norm retraction* omits the rescaling of edited rows, *projector added on top* applies the retained-span projector to KAU's update, and *no forgetting term* runs the retained pass alone; it forgets less than half of what the gold model forgets and cannot be brought to KAU's depth. Every other forgetting rate lies within 9 points of KAU's. Table 17 gives the variants that Table 16 omits.

Table 17: Remaining variants of KAU on FB15k-237 with TransE, in the setting of Table 16; the KAU row is repeated for reference. Means over three unlearning seeds.

Variant	ExRec↓	Leak↓	RER↓	M_R↑
KAU (full)	31.91	0.0111	19.38	0.2860
response damped (relative ridge 10^3)	47.13	0.0238	41.45	0.2869
no pessimistic composition	32.60	0.0110	19.18	0.2854
per-relation gate	34.18	0.0110	19.12	0.2859
no best-of-Q stand-in choice (Q=1)	37.90	0.0147	25.52	0.2857
hinge of the expected draw	31.98	0.0148	25.81	0.2857
ordered $k_{\max}$ cut	36.97	0.0155	27.02	0.2860
$\rho_{\mathrm{kn}} = 1$	33.92	0.0126	22.00	0.2864
$k_{\max} = 16$	38.56	0.0151	26.34	0.2856
relational anchor	33.49	0.0213	37.01	0.2868
no norm retraction	32.92	0.0121	21.00	0.2864
no forgetting term	77.39	0.0013	2.27	0.2830

Table 18: Alternative objectives replacing only KAU's forgetting step, in the setting of Table 16. Means over three unlearning seeds.

Forgetting objective	ExRec↓	Leak↓	RER↓
Plain forgetting loss (Δ=0)	48.60	0.0250	43.53
Retained-span projection	33.32	0.0158	27.51
SAM-RS	40.15	0.0154	26.85
SAM-GP	42.51	0.0150	26.09
SAM-CR	36.58	0.0152	26.55
SAM-WA	43.00	0.0151	26.30
Weight-distance regularisation	40.30	0.0171	29.73
Connectivity-based fine-tuning	42.31	0.0152	26.39
TAR-style unrolled	46.15	0.0158	27.54
Booster	43.73	0.0152	26.42
NTK scrubbing	55.20	0.0205	35.63
Weight distortion	53.56	0.0306	53.23
KAU	31.91	0.0111	19.38

Alternative forgetting objectives. Each alternative in Table 18 replaces KAU's forgetting step and keeps everything else. The setting of each objective's own hyperparameters is given in brackets. *Plain forgetting loss* is the variant of Table 16 without the anticipated response. *Retained-span projection* projects the forgetting gradient on each edited entity row off the span of the score gradients of that entity's retained facts (Chen et al., 2024; Hoang et al., 2024; Zhou et al., 2025), with projectors rebuilt every five epochs (ridge 10^{-2}). The *sharpness-aware* rows follow Fan et al. (2025b): SAM-RS averages the forgetting gradient over Gaussian perturbations of the edited rows ($\rho = 10^{-4}$, one perturbation), SAM-GP adds a penalty on the norm of the forgetting gradient ($\rho = 3 \times 10^{-5}$), SAM-CR adds a finite-difference curvature penalty ($w = 10^{-6}$, $\rho = 10^{-2}$), and SAM-WA writes an exponential average of the edited rows back once per epoch (decay 0.9). *Weight-distance regularisation* subtracts a multiple of the mean squared displacement of the edited rows from the original model (Siddiqui et al., 2025) ($\gamma = 10$). *Connectivity-based fine-tuning* adds, for the retained and the forgetting loss, a barrier term at the midpoint between the current and the original rows (Lubana et al., 2023) ($w = 10^{-3}$). *TAR-style unrolled* copies the entity table, runs Adam steps of the update's own loss on the retained facts of the current draw, and evaluates the forgetting loss at the displaced rows (Tamirisa et al., 2025). *Booster* adds a penalty on the reduction of the retained loss that one normalised gradient step would obtain (Huang et al., 2025) ($\alpha = 10^{-2}$). *NTK scrubbing* is a one-shot edit (Golatkar et al., 2020b): for each deleted fact its two endpoint rows are moved along the component of the fact's score gradient orthogonal to the span of the retained facts' gradients until the fact reaches its target to first order ($\epsilon = 0.32$), after which Fisher-scaled Gaussian noise is added at the largest scale on a ladder that keeps the retain-set MRR within 1% of the noise-free edit's; no non-zero scale did, so the noise term is zero. *Weight distortion* adds isotropic Gaussian noise to

the edited rows ($\sigma = 10^{-2}$) and then runs the retain fine-tuning through the retained pass, as in the original method (Siddiqui et al., 2025).

The forgetting rates at release lie between 88 and 100 percent of the gold model's, against 93 for KAU.

D.8 Why exact evaluation on piecewise-linear scores

The anticipated score is computed by adding the response to the rows and calling the score function again, rather than by adding the first-order term $\phi_f^\top \Delta\theta$ to the current score. Two facts motivate this on any backbone. The score is not linear in the rows, because of the row transform and of the form of the score itself, and the displacement the update is predicted to make is not small on the scale of the coordinate differences from which the score is built. Appendix D.5 measures the consequence, which is that the exactly evaluated response ranks facts by realized recovery better than the linear one.

On an L_1 score the size of the difference can be written down. Let $\boldsymbol{x} = \bar{\boldsymbol{e}}_h + \bar{\boldsymbol{w}}_r - \bar{\boldsymbol{e}}_t$ be the residual vector of a triple, so that $d_x = \|\boldsymbol{x}\|_1$, and let $\boldsymbol{D}$ be the displacement of that vector produced by the response. Lemma A.5, applied coordinate by coordinate, reads

$$|x_j + D_j| - |x_j| \;=\; \operatorname{sign}(x_j)\, D_j \;+\; 2\big(|D_j| - |x_j|\big)_+ \mathbf{1}\big[\operatorname{sign}(D_j) \neq \operatorname{sign}(x_j)\big]. \tag{20}$$

Up to the normalization's second-order term, the first term is what a first-order evaluation sees. The second is non-negative and is present only in the coordinates where the displacement opposes the residual and exceeds it in size, that is, where the coordinate crosses zero. Whether it matters is a question about two scales, the size of a coordinate $|x_j|$ against the size of the displacement $|D_j|$ the response predicts. A first-order evaluation of the anticipated score is blind to the second term by construction, and evaluating at the shifted rows captures it at the cost of one further call to the score function.

E Extended related work

This appendix gives the full related work that Section 2 summarizes. It places KAU among the relearning-robust unlearning methods and the unlearners for graphs. Table 19 records, for each study, the update it resists or measures and the object its objective reasons about. It also records whether that object couples retained to forgotten data, whether it computes what the update's data do to the forgotten items, and how the study is evaluated. Appendix E.1 states what the table shows, and Appendix E.2 states the scope of this paper relative to these studies. Appendices E.3, E.4, E.5 and E.6 expand Section 2. They cover unlearning in knowledge graphs and graph networks, benign relearning, the relearning-robust methods and the tangent-kernel response. Appendices E.7 and E.8 place the closest methods and the remaining studies one at a time.

E.1 What the table shows

No knowledge-graph or GNN unlearner has been evaluated for recovery of the forgotten facts under continued training. GraphDPO, ZeroFisher, Schema and MetaEU report the released model's forget and retained ranking and no training after the edit (Liu et al., 2026; Xiao et al., 2025a;b; Xu et al., 2025). GNNDelete, GIF and the OpenGU benchmark report membership or ranking readouts of the released model and no continued training (Cheng et al., 2023; Wu et al., 2023; Fan et al., 2025a). Projector fine-tunes after its edit, as a check that fine-tuning from the unlearned solution is cheaper than retraining, and reads utility from that fine-tune (Cong & Mahdavi, 2023).

Almost no relearning-robust unlearning method reports its gap to an identically fine-tuned retrained model. TAR compares against an undefended model and a random baseline, MUDMAN against the unedited model after the same relearning, MCU against the original model and RepSelect against the base model (Tamirisa et al., 2025; Sondej et al., 2025; Xiao et al., 2026; Sondej et al., 2026). Meta-unlearning, CIR, SAM, ILU, the spectral method of Rosati et al. (2026), GU and UNSC report no retrained reference under the fine-tune (Gao et al., 2025; Sondej & Yang, 2025; Fan et al., 2025b; Wang et al., 2025; Zhou et al., 2025; Chen et al., 2024). FRAG fine-tunes a retrained model identically but ranks methods by their own score change rather than by their distance to it (Chen

Table 19: Relearning-robust unlearning methods, retained-span projection methods and unlearners for graphs, by what their objective reasons about and how they are evaluated. Update model: the update the study resists or measures, benign (no forgotten item in it) or forget-set (forgotten items in it), and whether it adapts to the method. Coupling: whether the objective relates retained to forgotten data, and through what. Data effect: whether the objective computes what training on the update's data does to the forgotten items. Reference: whether a retrained model is the reference. Matched: whether the unlearned and the retrained model receive the same fine-tune.

Study	Domain	Update model	Object reasoned about	Coupling	Data effect	Reference	Matched	Cost
Relearning-robust unlearning methods								
TAR (Tamirisa et al., 2025)	LLM	forget-set, on forget, retain and mixed corpora, adaptive only by adding fine-tunes that broke earlier versions	weights and residual-stream activations, entropy of one fixed forget batch along a simulated multi-step trajectory	no, retain loss at the weights before the simulated fine-tune	partial, simulated, one fixed batch, straight-through gradient	no	none	a simulated fine-tune per outer step, multi-GPU
Meta-unlearning (Gao et al., 2025)	diffusion	forget-set, on forget images, benign set as test only, not adaptive	weights, two whole-set gradients, a few unrolled inner steps	partial, one inner product of whole-set gradients	partial, forget-data fine-tune, unrolled	no	none	second-order autodiff through the inner steps
ResAlign (Li et al., 2025)	diffusion	forget-set, sampled fine-tunes on a benign and harmful pool, benign at test, two adaptive attacks	weights, harmful loss at a simulated end point, implicit proximal Jacobian	no, retain as distillation at the current weights	partial, simulated, mean harmful loss at the end point	no	none	three simulated fine-tunes per outer step, about one GPU hour in total
MUDMAN (Sondej et al., 2025)	LLM	forget-set, on the forget set, fixed rate, not adaptive	per-parameter gradients, retain momentum, a forked copy	partial, per-coordinate sign agreement	partial, first-order, interleaved fine-tune on the forget set	no, unedited model after the same relearning	none	several passes per loop, hours of search
CIR (Sondej & Yang, 2025)	LLM	forget-set, long fine-tune on most of the forget set, not adaptive	principal components of forget-set activations and gradients	no, retain step optional	no, a hypothesis that broken representations are repaired	no	none	seconds per fact
RepSelect (Sondej et al., 2026)	LLM	forget-set, held-out same-domain data, an adaptive variant is tested	singular vectors of the accumulated forget-set gradient after a LoRA fine-tune	no, needs no retain set	partial, LoRA fine-tune on the forget set plus a proportionality assumption	no, base model	none	minutes per run, no search
SAM (Fan et al., 2025b)	LLM	forget-set, few forget samples, benign in one figure, not adaptive	weight vector, forget-loss gradient and Hessian inside a norm ball	no, additive retain loss at the unperturbed weights	no, the perturbation is the unlearner's own forget gradient	no	none	about twice the base cost
ILU (Wang et al., 2025)	LLM	benign, downstream tasks to convergence, forget relearning secondary, not adaptive	weight vector, one stationarity scalar per environment	no, additive terms	no, data enter through one scalar	no	none	several times the base cost, needs an auxiliary set
Siddiqui et al. (2025)	vision	benign, retain-only, one rate, not adaptive	whole weight vector, distance and loss barrier	no, norms are direction-blind	no	yes	yes, and the reference does not move	long unlearning runs
Rosati et al. (2026)	LLM, diffusion	forget-set and adaptive, rate sweeps, second-order optimizers, layer insertion, no benign	weight-matrix spectrum, global over all data	no	no, data-blind	no	none	seconds
MCU (Xiao et al., 2026)	LLM	forget-set, on most of the forget set, adaptive representation-matching variant, no benign	principal components of forget-set representations, isotropic kernel assumed	no, retain enters only the host's norm penalty	no, shared eigenstructure assumed	no, original model	none	one randomized SVD, rank chosen by full relearning runs
FRAG and FRP (Chen et al., 2026)	LLM	benign retain-only by default, also forget-only and mixed, rate, optimizer and epoch sweeps, not adaptive	per-weight importances from forget-to-retain channel activation ratios	partial, channel-support ratio, no relation among examples	partial, which channels the retain data activate, not how far outputs move	yes, retain-trained, identically fine-tuned	yes, but the ranked metric is the method's own score change	seconds to minutes, no training
UNDO (Lee et al., 2025)	LLM	forget-set, forget-set relearning, worst case over rates, not adaptive	weights and output distributions, noise and distillation	no	no	yes	yes, under forget-set relearning	a large fraction of retraining compute
Yoon et al. (2026)	LLM	benign, topical and syntactic relearn sets, not adaptive	dataset-mean hidden states and gradients, as a diagnostic	partial, aggregate alignment, post hoc	no	yes, as a benignness check	yes, to certify benignness only	one paraphrase pass
Retained-span projection and its origin								
Golatkar et al. (2020b)	vision, small subsets	relearning time on the complete data, forget set included, no benign	sample-gradient tangent kernel, projector off the retain span, Fisher noise	yes, pairwise kernel, used to compute the edit	no, analysis stops at release	yes	yes, under relearning on the complete data	memory quadratic in the retained set
UNSC (Chen et al., 2024)	vision	none, no fine-tune after release	per-layer retained activation spans per class, projector fixed at the original weights	partial, class activation span	no	yes, at release	no	one SVD per layer per class
GU (Zhou et al., 2025)	LLM	none, no relearning run	retain-gradient span in the Adam metric	partial, one-step entanglement norm	no	no	no	small overhead
GPM (Saha et al., 2021)	continual learning	none, GPM is itself the later optimizer	per-layer activation bases of past tasks	partial, span of old inputs	no	no	no	one SVD per layer per task
Adam-NSCL (Wang et al., 2021)	continual learning	none, Adam-NSCL is itself the later optimizer	null space of per-layer feature covariance	partial, covariance span of old features	no	no	no	one SVD per layer per task
Unlearners for graphs								
GraphDPO (Liu et al., 2026)	KGE	none	entity and relation rows, one-hop boundary replay, pointwise distillation to a frozen original	partial, boundary replay	no	yes, at release	no	faster than fine-tuning, frozen copy doubles memory
ZeroFisher (Xiao et al., 2025b)	KGE	none	Fisher curvature of the full loss, k-hop restriction	partial, implicit through the Fisher	no	yes, at release	no	iterations tuned from 3 to 1,550 (Appendix C.2)
Schema (Xiao et al., 2025a)	KGE	none	pairwise difference alignment in the two-hop neighbourhood to a schema-matched subgraph	partial, pairwise geometry, no forget-score term	no	yes, at release	no	minutes
MetaEU (Xu et al., 2025)	KGE	none	embeddings generated from relations and neighbours, loss maximization on forget queries	no	no	yes, at release	no	not reported
GNNDelete (Cheng et al., 2023)	GNN	none	layer-wise node representations on the k-hop subgraph, a deletion operator	partial, neighbourhood-influence term	no	yes, at release	no	much faster than retraining
GIF (Wu et al., 2023)	GNN	none	Hessian of the full loss, neighbour loss change within k hops	yes, neighbour loss term, aimed at the retraining optimum	no	yes, at release	no	about a second per request
Projector (Cong & Mahdavi, 2023)	linear GNN	benign fine-tune, as a utility check only	weight vector in the feature span, projection off deleted features	no, node dependencies folded into scalars	no	yes, at release	no, the retrained model is not fine-tuned	well under a second
KAU (this paper)	KGE	benign, 48 update configurations shared by every method, the strongest against each method chosen on a held-out selection half, not adaptive	block-sparse tangent kernel over triples, per-fact closed-form response, score at the shifted rows	yes, per-fact kernel blocks over shared rows	yes, the linearized response to a fresh draw each step	yes	yes	one linear solve per step, 13 s per release on CoDEx-L

et al., 2026). Siddiqui et al. (2025), Lee et al. (2025) and Dehghani et al. (2026) use the reference, with a reference that does not move under the fine-tune, a fine-tune on forget data, and a head-only audit with a frozen extractor respectively. The reference matters here because the retrained oracle itself gains forget-MRR under the update on FB15k-237 with TransE (Figure 2), so an absolute post-update score cannot separate a leak from legitimate learning.

No prior method writes in closed form what the update's data do to each forgotten item. The update enters as a simulation, an assumption, or a data-free proxy. TAR back-propagates the entropy of one fixed forget batch through a simulated fine-tune with a straight-through estimator (Tamirisa et al., 2025). Meta-unlearning unrolls a fine-tune on a subset of the forget data, MUDMAN interleaves a first-order fine-tune on the forget set, and RepSelect fits a LoRA fine-tune on the forget set and assumes the update's weight change is proportional to the accumulated gradient (Gao et al., 2025; Sondej et al., 2025; 2026). MCU assumes the relearning distribution shares its dominant eigenstructure with the forget representation, FRAG reads channel activation statistics and needs no relearning run, and CIR rests on the hypothesis that broken representations are repaired (Xiao et al., 2026; Chen et al., 2026; Sondej & Yang, 2025). SAM perturbs along the unlearner's own forget gradient, ILU reduces each environment to one stationarity scalar, Siddiqui et al. (2025) use a weight distance and Rosati et al. (2026) a spectrum (Fan et al., 2025b; Wang et al., 2025). None of these is closed form, none is per fact, and none uses a relation among examples.

Coupling between retained and forgotten data is absent or a single aggregate quantity. Meta-unlearning reduces it to one inner product of whole-set gradients, MUDMAN to a per-coordinate sign test, GU to a one-step entanglement norm, and FRAG to a channel-support ratio (Gao et al., 2025; Sondej et al., 2025; Zhou et al., 2025; Chen et al., 2026). TAR evaluates the retain loss at the weights before the simulated fine-tune, SAM and ILU add retain terms, the retain data of MCU enter neither its principal components nor its projector, RepSelect needs no retain set, and the retain step of CIR is optional (Tamirisa et al., 2025; Fan et al., 2025b; Wang et al., 2025; Xiao et al., 2026; Sondej et al., 2026; Sondej & Yang, 2025). Golatkar et al. (2020b) alone carry pairwise couplings, through the tangent kernel between forget and retain samples, and they spend them on computing the scrubbing edit, after which the analysis ends. A knowledge graph needs the pairwise object, because an entity row's retained gradient aggregates over every incident edge, so no scalar and no sign mask can say which neighbour edge restores which fact.

Projection off the retained span cancels the edit where the forgotten and the retained gradients share rows. The appendix of GU records that the projected step vanishes when the forget gradient lies in the retain tangent space (Zhou et al., 2025). Adam-NSCL states that zero singular values of the feature covariance cannot be guaranteed, and GPM reports that most of the gradient space is constrained after a handful of dissimilar tasks (Wang et al., 2021; Saha et al., 2021). On a knowledge graph a forgotten triple's gradient shares its rows with every retained neighbour, so this case is the entity of high degree. That is why the operator of Theorem A.1 is ridge-shrunk rather than a projector, and why the loss acts on the anticipated score rather than on the update direction. It is also why the retained-span projection, which approaches the second limit of Corollary A.3 as its ridge vanishes, leaks more than KAU at matched forgetting depth on TransE, 0.0158 against 0.0111 forget-MRR (Table 18).

Benign relearning is measured by several methods and modelled by none of the unrolled or subspace methods. Meta-unlearning states that self-destruction is meant to occur only under harmful fine-tuning and confirms that it does not occur on the benign set (Gao et al., 2025). MUDMAN, RepSelect, MCU, CIR and UNDO fine-tune on forget-domain data, and TAR includes retain-only fine-tunes in its sampled set but optimizes only their effect on the fixed forget batch (Sondej et al., 2025; 2026; Xiao et al., 2026; Sondej & Yang, 2025; Lee et al., 2025; Tamirisa et al., 2025). ILU trains on a benign fine-tune and FRAG evaluates one, so benign relearning is modelled there, through a global stationarity scalar and through channel support, never through the response of specific retained examples (Wang et al., 2025; Chen et al., 2026).

Resistance is often bought with utility or with self-destruction. TAR lowers retained accuracy and accepts a low post-update retained accuracy by design, and meta-unlearning makes the retained knowledge self-destruct under the fine-tune (Tamirisa et al., 2025; Gao et al., 2025). Siddiqui et al. (2025) find that the methods best at resisting relearning are those with the lowest test accuracy. The utility drop of FRP exceeds that of the unlearners it is compared with, and UNDO pays in general capability (Chen et al., 2026; Lee et al., 2025). KAU keeps the retained utility within 0.01 of the

oracle's on TransE, 0.2839 against 0.2932, where GraphDPO and ZeroFisher lose 0.03 and 0.06 (Table 5).

The robustness of a method that samples fine-tunes is tied to the sampled distribution. TAR reports sensitivity to parameter-efficient fine-tunes and to learning rates outside its sampled set and closes each hole by adding the offending fine-tune to that set, and meta-unlearning weakens against malicious fine-tuning as its number of inner steps grows (Tamirisa et al., 2025; Gao et al., 2025). KAU also draws from a class, but the draw is over the data of a benign recipe and the response to a draw is computed in closed form to first order, not sampled as a trajectory.

A method that only slows relearning is provably undone. Rosati et al. (2026) prove that a fine-tune designed against the method restores fast convergence at a cost linear in model size, and their sweep finds no earlier method that withstands standard fine-tuning once the learning rate varies. KAU relocates the edit rather than slowing convergence, and the results read each method under the configuration that leaks most against it on the selection half, at the epoch where its leakage peaks (Section 5).

Curvature and norm-ball proxies do not describe the update on a graph. The perturbation ball of SAM is small on the scale of the weights, its own results show its unlearning efficacy falling as the number of forget samples grows, and its mechanism, a bound on Hessian-vector norms, says nothing about which rows two triples share (Fan et al., 2025b).

E.2 Scope relative to these studies

The relearning-robust unlearning methods for language and diffusion models handle deep networks with no closed-form kernel, and several of them face fine-tunes that adapt to the method. The closed form of Lemma 1 holds for any differentiable score. It is explicit in the rows and cheap enough for every step because the score depends on three parameter rows, which a deep encoder does not offer. Our update is a benign recipe whose 48 configurations are fixed in advance and shared by every method. Each method is reported under the one that leaks most against it on the selection half of the probe (Section 5).

Several of these methods face fine-tunes that hold forgotten data. A fine-tune that includes forgotten edges re-inserts the data, the retrained oracle learns them too, and the gap then measures relearning speed against the oracle, which is the question of Lee et al. (2025).

Several methods are evaluated under multi-step trajectories, unseen update types, and the worst case over learning rates, epochs and seeds. TAR simulates a multi-step fine-tune with momentum optimizers, and ILU fine-tunes to convergence and holds on unseen downstream tasks. Four studies report the maximum over rates, epochs, seeds or a suite of fine-tunes (Deeb & Roger, 2024; Lee et al., 2025; Qi et al., 2025; Che et al., 2025). Here the response is the linearized response of the whole run of the update under frozen features (Appendix A.4), and Figure 2 shows each model's forget-MRR along the update, so the reader sees where the linearization stops predicting. The never-trained-edge update is the unseen case, and an update outside the class, on edges that share no entity and no relation row with a forgotten fact, has zero first-order coupling to that fact.

Several studies compare more unlearners and more checkpoints. Siddiqui et al. (2025) cover many unlearners across datasets and architectures, Che et al. (2025) many checkpoints, and OpenGU many graph unlearners (Fan et al., 2025a). This paper compares two published knowledge-graph unlearners, GraphDPO and ZeroFisher, three relearning-resistant add-ons on GraphDPO (Table 2), and the alternative forgetting objectives of Table 18, among them the retained-span projection. The projector of that row approaches, as its ridge vanishes, the zero-ridge limit of the per-entity survival filter with every retained edge drawn and no stand-ins (Proposition A.1 and Theorem A.1). It is the per-entity counterpart of the threshold at level zero in Corollary A.2, and its step direction is the matching limit of Corollary A.3.

Siddiqui et al. (2025) find that data unseen during training relearn less than seen data, and Appendix D.2 reads every method under an update built from never-trained edges alone (Table 13). An unseen true edge lands on the same shared rows as a seen one with a larger residual, because the edit never saw it.

FRAG gives a predictor of relearning that needs no relearning run (Chen et al., 2026). The anticipated post-update score of KAU is such a predictor, computed per fact, and it ranks forgotten facts by their realized recovery. Its per-fact Pearson correlation with that recovery is 0.53, and its Spearman correlation is 0.21 (Appendix D.5).

GU projects in the optimizer's own metric and Adam-NSCL guarantees plasticity for the later task (Zhou et al., 2025; Wang et al., 2021). The response here is Euclidean, and a degree-dependent ridge for Adam is stated as a hypothesis in Appendix A.12.

Golatkar et al. (2020b) bound what a reader of the scrubbed weights can extract, and GNNDelete, OpenGU, Projector and UNSC audit by membership inference (Cheng et al., 2023; Fan et al., 2025a; Cong & Mahdavi, 2023; Chen et al., 2024). The metric here is a ranking gap against the updated oracle, and a membership readout on a released embedding is a separate question.

GraphDPO and GNNDelete handle sequential requests, GraphDPO runs on bilinear backbones as well, and ZeroFisher and Schema delete entities and relations (Liu et al., 2026; Cheng et al., 2023; Xiao et al., 2025a;b). KAU is defined per request for any score that depends on three rows, and a new request reruns it with the new forget set.

ZeroFisher, RepSelect, CIR and the spectral method edit in seconds to minutes (Xiao et al., 2025b; Sondej et al., 2026; Sondej & Yang, 2025; Rosati et al., 2026). A KAU step costs one linear solve over all involved rows and no simulated fine-tune (Appendix B.10).

Meta-unlearning uses self-destruction as a deterrent and the spectral method uses slowing (Gao et al., 2025; Rosati et al., 2026). The goal here is equivalence to the oracle under benign use, so the released model must remain fine-tunable at a retained utility close to the oracle's, which rules out both.

Lee et al. (2025) show that a model matched to the oracle's outputs relearns faster than the oracle. That corroborates the reading that a pointwise anchor to a reference, such as the distillation of GraphDPO to the frozen original, confers no persistence, and it is why the score-preservation term of KAU holds the scores of retained edges rather than the rows (Proposition A.2).

E.3 Unlearning in knowledge graphs and graph networks

Exact unlearning retrains the model on the retained data (Cao & Yang, 2015; Bourtoule et al., 2021). Approximate unlearning edits the released model instead. One branch takes a single step that approximates the effect of removing the forgotten examples, with the inverse Hessian or its Fisher approximation (Koh & Liang, 2017; Guo et al., 2020; Izzo et al., 2021; Sekhari et al., 2021; Golatkar et al., 2020a). The other optimizes an objective for a few steps on the forgotten examples, sometimes with retained ones (Jang et al., 2023; Zhang et al., 2024).

The object of the edit depends on the model. In the translational and rotational backbones we study (Section 3), the trainable parameters are the entity and relation tables. Their score function has no weights of its own (Bordes et al., 2013; Wang et al., 2014; Sun et al., 2019). A row edit changes the scores of the facts at that row alone, so the change and its cost can be confined to the neighbourhood of the deletion. A graph neural network instead computes every node's representation from shared encoder weights by message passing. An edit to those weights changes every representation, so the collateral change, and the recomputation needed to control it, grow with the graph. Knowledge-graph unlearners therefore edit the tables themselves, and they differ in how much of the tables they update.

Liu et al. (2026) state the task as an update of the two tables, and their method GraphDPO updates both tables. It trains a preference objective that prefers a reconstructed alternative over each forgotten triple, with the original model as reference. It also replays the retained triples adjacent to the forgotten ones. ZeroFisher (Xiao et al., 2025b) applies an influence update in which a Fisher matrix replaces the Hessian and zeroth-order estimates replace the gradients. It iterates that update for a tuned number of steps (Appendix C.2). It restricts the update to the rows of the local neighbours of the forgotten edges, because the cost of the full update grows with the size of the graph. Schema (Xiao et al., 2025a) re-embeds the two-hop neighbourhood of the deleted entities so that it matches a subgraph with the same schema and without the deleted facts. MetaEU (Xu et al., 2025) meta-

learns a shared generator that regenerates the rows of the forgotten entities from their relations and neighbours.

Graph-neural-network unlearners edit the encoder or its output representations. GNNDelete inserts a deletion operator on the node representations of the affected subgraph and freezes the encoder (Cheng et al., 2023). GIF applies an influence update to the encoder weights with a term for the affected neighbours (Wu et al., 2023). Projector projects the weights of a linear encoder off the deleted node features (Cong & Mahdavi, 2023), and OpenGU benchmarks such methods (Fan et al., 2025a).

These knowledge-graph and graph-network unlearners use graph structure as the scope of the edit or as a preservation term for neighbours. None has a term that computes what training on those neighbours does to the forgotten score. They report the ranking the released model gives before any fine-tune (Appendix E.1). We extend that reading and ask what the edit is worth once training resumes.

E.4 Benign relearning and the update model

An unlearned model gives back what it was asked to forget once training resumes. In language models, fine-tuning on data that contains none of the forgotten answers restores them (Hu et al., 2025; Łucki et al., 2025; Yoon et al., 2026). Hu et al. (2025) are the origin of benign relearning and of the reading that unlearning may obfuscate rather than delete. We transplant that setting to knowledge-graph embeddings, score it against a retrained model that receives the same fine-tune, and replace an informal correlation between forgotten and retained data with a closed-form account of the linearized response of the fine-tune. Deeb & Roger (2024) show that fine-tuning on an accessible subset of the forget set restores held-out forgotten facts, and they name the absence of a reliable reference as the open problem. Our update holds no forgotten fact at all, and the reference is the retrained model after the same update.

Recovery grows as the fine-tune proceeds, so evaluations read the worst point over its steps and learning rates (Deeb & Roger, 2024; Lee et al., 2025). Others read the worst case over seeds, configurations and update recipes (Qi et al., 2025; Che et al., 2025). Section 5 follows both, reading the epoch of peak leakage and the worst update configuration, but averages over seeds.

Siddiqui et al. (2025) establish the reading we adopt. They show that retain-only fine-tuning restores forget accuracy for most vision unlearners while a retrained model stays robust, they define tamper resistance as the requirement that the relearned unlearner not beat the relearned retrained model, which is the leakage of equation 1 read against the oracle's zero here, and they report that unseen data relearn less. Two things differ on a knowledge graph. The retrained model itself recovers under the update (Figure 2), so the reference is not a fixed point. And an update built only from never-trained edges is read for every method in Appendix D.2 (Table 13).

Yoon et al. (2026) attribute benign relearning in language models to syntactic alignment between the relearning data and the forgotten targets, and they defend by paraphrasing the forget set. A triple cannot be paraphrased, and the alignment between a forgotten fact and its retained neighbours is exact, which removes the data-side remedy and forces a storage-side one. Suriyakumar et al. (2026) show the same failure in diffusion models, where fine-tuning on concepts disjoint from the erased one resurrects it, and they lower-bound the leaked gradient mass by the overlap of the two spans. We sharpen that observation into an objective in which the edit is chosen against the overlap rather than measured after the fact.

Dehghani et al. (2026) recover forgotten classes with no forget, retain or auxiliary data, by probing the representation space and updating the head alone, and they measure excess recoverability relative to a retrained model under an identical audit. We follow that convention and report the leakage beyond the retrained oracle in MRR points and as a share of the erased knowledge, beside the forgetting at release, because a method can buy one with the other (Section 5).

E.5 Relearning-robust unlearning methods

Three families anticipate the update, and they differ in whether they simulate it, replace it by a proxy, or choose the subspace the edit occupies. Appendix E.1 sorts the same methods by how the update enters their objective, as a simulation, an assumption or a data-free proxy.

Unrolled methods anticipate the update by simulation. They simulate the update inside the objective and evaluate the forgetting loss at the simulated weights. Meta-unlearning takes the gradient through the simulation exactly (Gao et al., 2025), and TAR takes it by a straight-through estimator (Tamirisa et al., 2025). ResAlign does similarly, taking the gradient implicitly through a proximal model of the simulation's end point (Li et al., 2025). MUDMAN does not differentiate through the simulation. It evaluates the forgetting gradient at a simulated copy and applies it to the original weights (Sondej et al., 2025). KAU replaces the simulation by the closed-form linearized response of the update for a shallow relational kernel.

Proxy methods replace the update by a cheaper quantity that does not describe the update itself. SAM penalizes the sharpness of the forgetting loss under weight perturbations over a finite ball (Fan et al., 2025b). ILU penalizes the variation of the loss across auxiliary datasets, a stationarity penalty with one scalar per dataset (Wang et al., 2025). Siddiqui et al. (2025) regularize a weight-space distance to the original model. Rosati et al. (2026) control the weight spectrum and show that a fine-tune designed against any method that only slows recovery undoes it. That result, detailed in Appendix E.1, is why we report the gap along the fine-tune and not only at its end. Relearning-speed measures (Lee et al., 2025; Tran & Woo, 2026) are the unpaired version of that reading.

Subspace methods, the third family, choose where the edit lives. Retained-span projection places the edit off the span of the retained gradients to protect utility (Golatkar et al., 2020b; Chen et al., 2024; Zhou et al., 2025). Slow-direction methods place the edit in directions that an update restores slowly. MCU and CIR take those directions from the principal components of the forget set's representations and activations respectively (Xiao et al., 2026; Sondej & Yang, 2025). RepSelect takes them from the weight gradient that a low-rank fine-tune on the forget set accumulates (Sondej et al., 2026). FRAG places the edit on weights whose input channels the retained data barely activate (Chen et al., 2026). Appendix E.6 traces the retained-span projector to continual learning and discusses MCU, and Appendix E.7 discusses FRAG and RepSelect.

These methods are built for deep networks. Each reads its subspace or its proxy off the network's weights and activations, or runs one simulated update. None of these quantities says which training example moved which parameter. A deep model does not expose that attribution in closed form. Carried over to a knowledge-graph embedding, they leave its structure unused. An unrolled method returns the rows after one simulated update and no expression for which edge moved which row. A proxy method reads sharpness, weight distance or spectrum off the whole table, and those quantities say nothing about which edges drive which rows. A projection method projects off the retained-gradient span over all rows rather than off the span of the update's edges at the shared rows. KAU instead writes the update's effect on each forgotten fact in closed form, as a function of drawn edges the update may train on, and differentiates through it. Appendix E.7 reads both projection families through the survival filter of Section 4.6, the slow-direction ones on a spectrum estimated from the forget or retained data.

E.6 The tangent-kernel response and its origin

The function KAU differentiates through is the linearized response of training, and it can be built up from one gradient step. Take one gradient step on one example. To first order, the model's output on a second example moves in proportion to the inner product of the two examples' output gradients. That inner product is the tangent kernel (Jacot et al., 2018), and it says how much training on one example moves the prediction for another. Over a run of small steps the same rule holds at every step, with the kernel held at the current parameters. To first order the total change in any output is then the solution of a ridge-regularized linear system in that kernel (Section 4.2). The ridge term stands in for the finite length of the run. In a knowledge-graph embedding a triple's score depends on three rows. The kernel between two triples is then an explicit expression in the rows of the two triples, and it is zero unless they share an entity or a relation. The kernel therefore names, for each forgotten fact, which trained edges move it and through which rows. That is the attribution the

methods of Appendix E.5 lack. The effect of the update on a forgotten fact can then be written down as a linear system in that kernel rather than simulated.

The linearized cross-task response we differentiate through is not ours. Doan et al. (2021) and Bennani et al. (2020) derive it for continual learning, where one phase of training moves the function on an earlier phase's data through the tangent kernel. Both use it to bound the resulting interference. Hidekel & Raviv (2026) give a recent function-space account of the same motion. Malladi et al. (2023) describe the fine-tuning of a pretrained model by the same kernel. Li et al. (2025) differentiate a forgetting loss through a proximal model of a simulated fine-tune, after Rajeswaran et al. (2019). What is new here is the use we make of it. The second phase is the downstream fine-tune, and the residual that drives it on the retained edges is created by the unlearning edit instead of being given by the data. The resulting closed form is placed inside an objective the unlearner minimizes over a class of draws.

Golatkar et al. (2020b) run the same kernel algebra in the opposite direction. They derive a one-shot scrubbing update from the linearized dynamics, project the forgetting gradient off the span of the retained gradients, add Fisher-shaped noise, and bound what a reader of the scrubbed weights can extract. Their kernel couples each forget sample to each retain sample pairwise, and that coupling is spent on computing the edit. Their analysis ends at the scrubbed weights, and their relearning readout fine-tunes on the complete data, forget set included. Ours begins with the benign fine-tune after release. We use the same algebra with the roles inverted, so that the kernel predicts the update instead of computing the edit. Corollary A.3 identifies their projector, restricted to one entity row, with the zero-ridge limit of our operator once the unseen channel is switched off. Read through the survival filter of Corollary A.2, a projector introduced to protect utility also marks the edits that continued training on retained data cannot undo to first order.

The projector off the span of earlier data has a second origin in continual learning. GPM keeps a new task's gradient off the activation bases of past tasks and Adam-NSCL keeps it in the approximate null space of their feature covariance (Saha et al., 2021; Wang et al., 2021). Both are optimizers of the later phase, so their invariance holds while they control the update, and the unlearners that reuse the construction (Chen et al., 2024; Zhou et al., 2025) protect the retained loss from the edit.

Xiao et al. (2026) give the closest statement about which components a fine-tune restores. Under an isotropic-kernel linearization, an edit placed in a high-variance component of the representation is reversed at a rate proportional to that variance, which is what their minor-component method exploits. The isotropic kernel drops exactly the off-diagonal blocks that carry the coupling in an embedding model, where the kernel between two triples is nonzero only when they share a row. Our ridge σ_v plays a comparable role in the eigenvalues of a block-sparse graph-local kernel, and the subspace we avoid is the span of the update's data rather than the forget representation's principal directions. In the survival filter of Proposition A.1 their minor components correspond to the directions of small eigenvalue. Confining the edit to such directions is a hard threshold of the filter (Corollary A.2). Their shared-eigenstructure assumption is what lets the spectrum of the forget representation stand for the spectrum of the update.

E.7 Methods closest in design

Chen et al. (2026) argue that robustness is governed by where an update lives rather than by how far it moves, and they give a score computed per checkpoint that needs no relearning run. Their pruning rule places the edit on weights that the retain data barely activate, so that a retain-only fine-tune leaves them inert. In a knowledge graph the rows a forgotten fact uses are the rows its retained neighbours use, so no row is inert under a retain-only update. The placement question then has to be asked inside the span of an entity's edges rather than across channels. For a shallow embedding model that question can be made exact, because the span of the update's data at an entity is a closed-form graph-local tangent space. The response goes one step further by evaluating the score at the shifted rows instead of along the tangent.

Sondej et al. (2026) confine the edit to the low-variance complement of the forget gradient's principal subspace and summarize the principle as selectivity rather than update strength. We share the idea of storing suppression outside the subspace a later fine-tune reaches. We define that subspace by drawn edges the update may train on, through a kernel available in closed form. In the linearized model the anticipated loss is the plain forgetting objective on the part of the edit that survives the

update (Corollary A.1). Projection onto the complement is the zero-ridge threshold of that survival filter, a consequence instead of a starting point (Corollary A.2).

Yin et al. (2026) calibrate a one-sided loss against a reference model's margins with a budget-bounded certificate. Our loss is one-sided in the same way and is applied to the anticipated post-update score, and its per-fact targets are computed from quantiles of corrupted variants under the original model, so it needs no access to the retrained model.

These three, together with the retained-span methods above, share one object with us and use it differently. A null space enters prior work in three roles. It is a fixed layer operator that completes an ill-posed inverse (Yin et al., 2025; Fang et al., 2025). It is the null space of retained features or gradients that keeps a later update off earlier data in continual-learning optimizers (Saha et al., 2021; Wang et al., 2021) and that protects utility from the unlearning edit (Golatkar et al., 2020b; Chen et al., 2024; Hoang et al., 2024; Zhou et al., 2025). It is the low-variance part of the forget representation that a fine-tune restores slowly (Xiao et al., 2026; Sondej et al., 2026; Sondej & Yang, 2025). Ours is the complement of the span of the update's data, chosen for persistence. In the linearized model it is the set of edits the update leaves untouched (Proposition A.1). Read through the survival filter, the second role is its hard threshold at level zero, restricted to the involved entities' rows, when the draw is exactly their retained edges. The third has the form of a hard threshold at a positive level, taken on a proxy spectrum (Corollary A.2). In the linearized model under a uniform ridge, KAU's gradient instead weights each direction by its surviving fraction, computed from the spectrum of its own update draws. Unseen edges in the update can move edits that retained-span projection preserves, whereas KAU's draws include stand-ins for those edges (Section 4.1). Where the forgotten gradient lies inside the retained span, a hard projector removes the whole edit, which the appendix of GU records as a local trade-off (Zhou et al., 2025). On a graph that is the entity of high degree. There the cap $k_{\max}$ keeps every draw below full rank, and the ridge of Theorem A.1 keeps the operator from annihilating the forgetting direction inside the drawn span (Section 4.2).

E.8 Remaining placements

Three further threads touch the mechanism. The image of an alignment edit that is pulled back toward its origin is due to Ji et al. (2025). Arora et al. (2019) and Heo et al. (2021) study how a growing row norm shrinks an effective learning rate. The retraction rule of Appendix B.4 removes this effect. Studies of when an unlearned model is reversible (Ha et al., 2026; Chang & Khanna, 2026; Xu et al., 2026) ask the same question at the level of the loss landscape.