QUANTINUE: QUANTIZATION FOR UNLEARNING OPERATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning aims to make a trained model behave like a model retrained without designated forget data, and the resulting model is typically quantized for efficient deployment. Low-bit quantization, however, can undo unlearning by discarding parameter changes introduced during unlearning and recovering knowledge that was previously forgotten. Existing methods address this by making the unlearning stage more robust to quantization, but they still treat unlearning and quantization as two separate steps. We instead replace the two stages, unlearning and quantization, with quantization itself the unlearning operator. Quantization maps continuous weights to discrete codes while seeking to retain the original model behavior. We guide this mapping not only to retain the original model behavior on the retain data, but also to induce forgetting behavior on the forget data, directly constructing an unlearned low-bit model without a preceding continuous-space unlearning update. Activation statistics from forget and retain calibration data define an element-wise score that determines where and how strongly the assigned code deviates from the reconstruction-driven code. The resulting process separates ordinary quantization error from code-displacement error and compensates them separately, reducing compensation along forget-sensitive directions for the code displacement error. Both code assignment and error compensation use only forward calibration statistics, without gradient computation or fine-tuning. On the MUSE and TOFU benchmarks, our method achieves the lowest aggregate distance to the retrained reference among the compared methods in 4 bit precision. These results suggest that quantization, previously a threat to unlearning, can instead serve as an effective means of achieving it.

1 INTRODUCTION

Machine unlearning aims to update a trained model in response to a request to forget designated forget data while preserving its utility on the remaining retain data (Ginart et al., 2019; Bourtoule et al., 2021; Cao & Yang, 2015; Guo et al., 2020; Golatkar et al., 2020; Jang et al., 2023; Zhang et al., 2024; Fan et al., 2025; Shen et al., 2025; Turani et al., 2026; Liu et al., 2025a). The reference solution is a model retrained from scratch without the forget data, but repeating full training is prohibitively expensive for large models, which motivates efficient methods that approximate this retrained model (Ginart et al., 2019; Bourtoule et al., 2021; Cao & Yang, 2015; Guo et al., 2020; Golatkar et al., 2020; Fan et al., 2024; Foster et al., 2024; Turani et al., 2026; Liu et al., 2025a). This retraining-based criterion defines unlearning in terms of the forget data itself, and recent work distinguishes it from broader objectives that aim to suppress undesired model behavior (Yoon et al., 2026; Yao et al., 2024). For deployment, low-bit quantization reduces model size and inference cost by mapping high-precision model weights to a discrete set of low-precision values (Frantar et al., 2023; Lin et al., 2024; Xiao et al., 2023; Shao et al., 2024; Chee et al., 2023; Tseng et al., 2024; Egiazarian et al., 2024; Ashkboos et al., 2024; Liu et al., 2025b). Post-training quantization (PTQ) applies this transformation to a pretrained model without retraining, and recent methods enable LLM weights to be represented at 3-4 bit precision with limited degradation in model utility (Frantar et al., 2023; Lin et al., 2024; Shao et al., 2024; Ashkboos et al., 2024; Liu et al., 2025b). When applied after machine unlearning, however, this transformation does not necessarily preserve the achieved forgetting. Zhang et al. (2025); Mishra & Mehreen (2026); Ranjan et al. (2026); Khan & Sohel (2026) show that 4-bit post-training quantization can recover a substantial portion of the knowl-

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.

edge removed by machine unlearning. This recovery is attributed to parameter changes induced by utility-preserving unlearning that are not consistently preserved after discretization (Zhang et al., 2025). These results indicate that forgetting achieved in continuous parameter space is not necessarily preserved after quantization (Zhang et al., 2025; Abitante et al., 2026; Ranjan et al., 2026; Khan & Sohel, 2026). Quantization-robust unlearning methods address this failure by incorporating robustness to subsequent quantization into the unlearning process (Abitante et al., 2026; Mishra & Mehreen, 2026; Ranjan et al., 2026; Khan & Sohel, 2026). These methods retain a two-stage formulation in which unlearning precedes quantization (Abitante et al., 2026). Prior methods treat quantization-induced distortion as a perturbation that the preceding unlearning update is designed to withstand (Mishra & Mehreen, 2026). We reverse this view by turning quantization-induced distortion into an unlearning intervention through discrete code assignment. Quantization maps continuous weights to discrete codes while seeking to retain the original model outputs for the same inputs. We guide this mapping to account for the forget data instead of relying on reconstruction alone, directly constructing an unlearned low-bit model without a preceding continuous-space unlearning update. Specifically, we perform unlearning by modifying the discrete code assignment of PTQ. Activation statistics from forget and retain calibration data define an element-wise score that determines where and how strongly the assigned code deviates from the nearest code. The resulting quantization process contains two error components: the quantization error of the nearest code and the code displacement error introduced by the score-guided deviation. The quantization error follows reconstruction-based compensation, whereas the code displacement error propagates along a modified direction determined by forget and retain statistics. Both code assignment and error compensation use only activation statistics computed from forward passes on calibration data, without gradient computation or fine-tuning. We evaluate our method on the MUSE and TOFU benchmarks at 4-bit precision (Shi et al., 2025; Maini et al., 2024). Our formulation directly constructs an unlearned low-bit model without a separate post-unlearning quantization stage, removing the transition associated with knowledge recovery in prior work. Our method achieves the lowest aggregate distance to the retrained reference among the compared methods on both benchmarks, and this result holds for the 1B and 7B models on TOFU.

Our contributions are as follows:

- We reformulate quantization-robust unlearning as a single-stage procedure that integrates unlearning into quantization, removing the post-unlearning quantization stage in which prior work reports knowledge recovery.
- We realize this formulation through score-guided discrete code assignment based on forget and retain activation statistics, together with an error compensation rule that separates the code displacement error from the quantization error and propagates it along a modified direction with reduced correlation to the output change on forget data.
- We show that, at 4-bit precision, our method achieves the lowest aggregate distance to the retrained reference among the compared methods on MUSE and TOFU.

2 Related Work

2.1 Machine Unlearning

Machine unlearning addresses deletion requests for designated forget data while preserving utility on retain data (Ginart et al., 2019; Bourtoule et al., 2021; Cao & Yang, 2015; Guo et al., 2020; Golatkar et al., 2020; Jang et al., 2023; Liu et al., 2025a). A common formulation optimizes a forgetting loss on the forget data together with a loss or regularizer that preserves behavior on the retain data (Maini et al., 2024; Zhang et al., 2024; Yao et al., 2024; Fan et al., 2025; Yuan et al., 2025). These procedures differ in the specific objectives, but all implement forgetting through gradient-based optimization of continuous parameters (Jang et al., 2023; Yao et al., 2024; Zhang et al., 2024; Fan et al., 2025; Hong et al., 2024; Yoon et al., 2026; Yuan et al., 2025; Cha et al., 2025). Benchmarks evaluate the resulting models against a retrained reference, separating targeted forgetting from general degradation (Dorna et al., 2025; Wang et al., 2025): TOFU pairs forget quality with model utility (Maini et al., 2024), and MUSE adds privacy leakage and sustainability under sequential requests (Shi et al., 2025).

2.2 POST-TRAINING QUANTIZATION

Post-training quantization represents trained parameters with a finite set of low-bit values, and calibration-based methods reduce the resulting accuracy loss using statistics of calibration data (Lin et al., 2024; Xiao et al., 2023; Frantar et al., 2023; Nagel et al., 2020; Li et al., 2021; Frantar & Alistarh, 2022; Shao et al., 2024; Chee et al., 2023; Tseng et al., 2024; Egiazarian et al., 2024; Ashkboos et al., 2024; Liu et al., 2025b). Activation-informed methods use these statistics to rescale weights and activations before quantization (Lin et al., 2024; Xiao et al., 2023). Reconstruction-based methods instead use calibration data to select quantized weights whose layer outputs stay close to the original ones (Frantar et al., 2023; Nagel et al., 2020; Li et al., 2021; Frantar & Alistarh, 2022; Shao et al., 2024; Ding et al., 2025). GPTQ performs this reconstruction sequentially and compensates for the error of each code assignment by updating the remaining unquantized weights with approximate second-order information (Frantar et al., 2023). Code assignment and compensation in GPTQ both reduce the discrepancy between the original and quantized layer outputs on calibration data. Our method reuses this sequential structure but redirects both steps toward unlearning. Code assignment displaces the codes of weights associated with the forget data, and compensation preserves the layer output on retain data without offsetting the displacement.

2.3 UNLEARNING UNDER QUANTIZATION

Prior work reports that 4-bit quantization recovers a substantial portion of the knowledge removed by unlearning (Zhang et al., 2025; Khan & Sohel, 2026). Quantization-robust unlearning addresses this recovery by modifying the unlearning procedure to account for subsequent quantization (Zhang et al., 2025; Abitante et al., 2026; Mishra & Mehreen, 2026; Ranjan et al., 2026; Khan & Sohel, 2026). Existing methods restrict the unlearning update to subsets of parameters, such as salient modules (Zhang et al., 2025), low-rank adapters (Abitante et al., 2026), and selected layers (Ranjan et al., 2026), or incorporate discretization into the training objective (Mishra & Mehreen, 2026; Khan & Sohel, 2026). Beyond language models, Tong et al. (2025) rebalance the forget and retain updates of quantized classifiers through label reassignment and gradient reweighting. These methods treat unlearning and quantization as two separate stages, and unlearning before quantization requires the update to remain effective after discretization. Our method instead combines both stages into a single stage, performing forgetting during code assignment using forget and retain activation statistics.

3 PRELIMINARIES

3.1 MACHINE UNLEARNING

Let $\mathbb{D}$ denote the original training dataset, partitioned into designated forget data $\mathbb{D}_F$ and retain data $\mathbb{D}_R = \mathbb{D} \setminus \mathbb{D}_F$. Let θ_0 denote the parameters of a model trained on $\mathbb{D}$. An unlearning procedure $\mathcal{U}$ produces

$$\theta_u = \mathcal{U}(\theta_0; \mathbb{D}_F, \mathbb{D}_R). \tag{1}$$

Retraining from scratch on $\mathbb{D}_R$ yields the reference parameters θ_r, against which θ_u is evaluated in terms of forgetting behavior and retain utility (Ginart et al., 2019; Bourtoule et al., 2021).

3.2 POST-TRAINING QUANTIZATION

Consider a linear layer $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and $\boldsymbol{X} \in \mathbb{R}^{d_{\text{in}} \times n}$. Let $\mathbb{Q}$ denote the set of admissible quantized weight matrices. Layer-wise reconstruction can be written as

$$\widehat{\boldsymbol{W}}^* = \arg\min_{\widehat{\boldsymbol{W}} \in \mathbb{Q}} \left\| \boldsymbol{Y} - \widehat{\boldsymbol{W}}\boldsymbol{X} \right\|_F^2. \tag{2}$$

GPTQ performs reconstruction-based quantization with sequential error compensation using the Hessian $\boldsymbol{H} = 2\boldsymbol{X}\boldsymbol{X}^\top$ (Frantar et al., 2023). Let $\boldsymbol{H}^{-1} = \boldsymbol{U}^\top \boldsymbol{U}$ be the Cholesky factorization, where $\boldsymbol{U}$ is upper triangular. Suppose that coordinate w_j of an output row is quantized to $\widehat{w}_j$, and let $\mathbb{T}_j$ denote the coordinates remaining unquantized after j. Let $h_j = |U_{jj}|$ and $e_j = (w_j - \widehat{w}_j)/h_j$ denote the diagonal scaling and the scaled quantization error. GPTQ updates the remaining coordinates of the row as

$$\boldsymbol{W}_{\mathbb{T}_j} \leftarrow \boldsymbol{W}_{\mathbb{T}_j} - e_j \ell_j, \qquad \ell_j = U_{j,\mathbb{T}_j}, \tag{3}$$

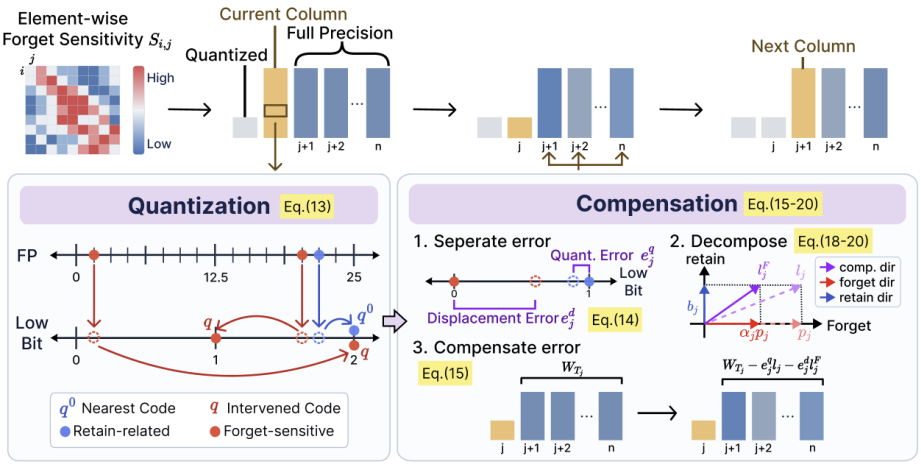

Figure 1: Overview of our method. The intervention score displaces the codes of high-score weights from their nearest codes (Quantization), and the code displacement error propagates along a direction whose forget component is attenuated (Compensation).

where ℓ_j is the error-propagation direction.

3.3 Quantization-Robust Unlearning

Quantization-robust unlearning modifies the preceding unlearning procedure to account for subsequent quantization (Zhang et al., 2025; Abitante et al., 2026; Mishra & Mehreen, 2026; Ranjan et al., 2026; Khan & Sohel, 2026). Let $\mathcal{U}_Q$ denote such an unlearning procedure and $\mathcal{Q}$ a post-training quantization operator. Existing approaches follow the sequential formulation

$$\theta_u = \mathcal{U}_Q(\theta_0; \mathbb{D}_F, \mathbb{D}_R), \qquad \widehat{\theta}_u = \mathcal{Q}(\theta_u). \tag{4}$$

The continuous-parameter unlearned model θ_u is therefore constructed before the final low-bit model $\widehat{\theta}_u$ is obtained through quantization.

4 Unlearning with Quantization

Problem reformulation. Conventional quantization-robust unlearning constructs a continuous-parameter unlearned model θ_u before applying a separate quantization step, as described in Equation 4. We instead directly construct the low-bit unlearned model from the original model:

$$\widehat{\theta}_u = \mathcal{Q}_u\left(\theta_0; \mathbb{D}_F, \mathbb{D}_R\right), \tag{5}$$

where $\mathcal{Q}_u$ denotes our unlearning-aware quantization procedure. This formulation removes the intermediate continuous-parameter unlearned model and performs the unlearning intervention as part of quantization itself.

Overview. We realize $\mathcal{Q}_u$ within the sequential quantization of GPTQ by modifying two of its steps. Score-guided code assignment displaces the codes of weights associated with the forget data away from their nearest codes. The compensation of GPTQ in Equation 3 can offset this displacement. Unlearning-aware error compensation therefore modifies the propagation of the displacement error, preserving the layer output on retain data without offsetting the displacement.

Figure 1 illustrates the procedure in five steps. For each linear layer, we (i) compute an element-wise intervention score from activation statistics on the forget and retain calibration data. For each input coordinate in the order of GPTQ, we then (ii) displace the nearest codes according to the intervention score, (iii) separate the resulting error into the quantization error and the code displacement error, (iv) attenuate the forget-related component in the propagation direction of the code displacement error, and (v) compensate both errors on the remaining coordinates.

4.1 SCORE-GUIDED QUANTIZATION CODE ASSIGNMENT

Starting from the sequential quantization of GPTQ, we use forget and retain activation statistics to determine where to intervene and how strongly to displace the assigned code.

Where to intervene. For a linear layer with weight matrix $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we define the element-wise intervention score as

$$\boldsymbol{S} = \bar{\boldsymbol{s}}^{\text{out}} \left(\bar{\boldsymbol{s}}^{\text{in}}\right)^{\top} \in [0,1]^{d_{\text{out}} \times d_{\text{in}}}, \tag{6}$$

where $\bar{(\cdot)}$ denotes the normalized form. The output score $\boldsymbol{s}^{\text{out}}$ and the input score $\boldsymbol{s}^{\text{in}}$ determine how strongly the assigned codes are displaced across output and input coordinates.

For an input activation $\boldsymbol{x}$ and the corresponding output $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$, we compute two activation statistics on the forget and retain calibration data, indexed by $D \in \{F, R\}$:

$$\boldsymbol{H}_D = \mathbb{E}\left[\boldsymbol{x}\boldsymbol{x}^{\top}\right], \qquad u_{D,i} = \mathbb{E}\left[y_i^2\right], \tag{7}$$

where each expectation is taken over calibration activations from data split D. The diagonal of $\boldsymbol{H}_D$ and the values $u_{D,i}$ measure how strongly each input coordinate and each output coordinate are activated on data split D, respectively. The output and input scores compare these statistics between the forget and retain splits.

For output coordinate i, we define the output score as

$$s_i^{\text{out}} = \left(\log \frac{u_{F,i}}{u_{R,i}}\right)^{+} \log\left(1 + \bar{u}_{F,i}\right), \tag{8}$$

where $x^{+} = \max(x, 0)$. The first factor selects output coordinates that are activated more strongly on forget data than on retain data and ranks them by the ratio $u_{F,i}/u_{R,i}$. The second factor weights these coordinates by their normalized forget activation $\bar{u}_{F,i}$.

The input score compares forget and retain statistics of each input coordinate, as the output score does for output coordinates. For input coordinate j, the score uses $H_{F,jj}$ and the diagonal entry U_{jj} of the upper-triangular Cholesky factor in $\boldsymbol{H}_R^{-1} = \boldsymbol{U}^{\top}\boldsymbol{U}$. The value $1/U_{jj}^2$ is the second moment, on retain data, of the part of input coordinate j that the coordinates in $\mathbb{T}_j$ cannot linearly predict. We combine the two statistics in two ways:

$$s_j^{\text{sep}} = \left(\overline{\log H_{F,jj}} + \overline{\log U_{jj}^2}\right)^{+}, \qquad s_j^{\text{joint}} = \left(\overline{\log\left(H_{F,jj}\, U_{jj}^2\right)}\right)^{+}. \tag{9}$$

In s_j^{sep} the two terms are normalized separately and contribute on a common scale, whereas in s_j^{joint} they are normalized as a sum and the term with the larger spread across input coordinates contributes more. We normalize s_j^{sep} and s_j^{joint} separately and define the input score as their pointwise minimum:

$$\bar{s}_j^{\text{in}} = \min\left(\bar{s}_j^{\text{sep}}, \bar{s}_j^{\text{joint}}\right). \tag{10}$$

The minimum yields a conservative score, since an input coordinate receives a large input score only when both $\bar{s}_j^{\text{sep}}$ and $\bar{s}_j^{\text{joint}}$ are large.

How strongly to intervene. We convert the intervention score into an element-wise strength:

$$\rho_{i,j} = 1 + \kappa S_{i,j}, \qquad \kappa \geq 0, \tag{11}$$

where κ controls the global intervention strength. The resulting code displacement depends on both $\rho_{i,j}$ and the offset of each weight from its assigned code.

How the code is changed. Let Δ_i denote the quantization step for output row i, and let $Q_b(\cdot)$ return the nearest valid integer code under b-bit quantization. During sequential quantization, we compute the nearest code of the current weight $w_{i,j}$ and the quantization residual of $w_{i,j}$ in units of Δ_i:

$$q_{i,j}^0 = Q_b\left(\frac{w_{i,j}}{\Delta_i}\right), \qquad \delta_{i,j} = \frac{w_{i,j}}{\Delta_i} - q_{i,j}^0. \tag{12}$$

The code $q^0_{i,j}$ is the code that the GPTQ rounding rule assigns before the score-guided displacement. The strength $\rho_{i,j}$ amplifies the residual $\delta_{i,j}$ before the final discrete code is assigned, and the final code determines the integer code displacement and the quantized weight:

$$q_{i,j} = Q_b\left(q^0_{i,j} + \lfloor \rho_{i,j}\delta_{i,j} \rceil\right), \qquad k_{i,j} = q_{i,j} - q^0_{i,j}, \qquad \widehat{w}_{i,j} = \Delta_i q_{i,j}, \tag{13}$$

where $\lfloor\cdot\rceil$ denotes nearest-integer rounding. A zero score gives $\rho_{i,j} = 1$, for which Equation 13 yields the nearest code $q^0_{i,j}$ except at rounding ties. The score $\boldsymbol{S}$ therefore determines where codes can be displaced, and κ controls the global strength of the displacement.

4.2 Unlearning-Aware Error Compensation

GPTQ propagates quantization error to the remaining unquantized weights to preserve the layer output on calibration data. The code displacement in our assignment introduces an additional error. The GPTQ update compensates for this error together with the quantization error, although the displacement is the intended unlearning intervention. We therefore separate the two errors and modify the propagation direction only for the code displacement error.

Error decomposition. For a fixed output row i, we omit its index from the weights and the errors. We write $\boldsymbol{W}$ for the current weights after the preceding error-propagation updates. At coordinate j, we decompose the scaled error e_j in Equation 3 into the quantization error of the nearest code and the code displacement error, $e_j = e^q_j + e^d_j$:

$$e^q_j = \frac{w_j - \Delta q^0_j}{h_j}, \qquad e^d_j = -\frac{\Delta k_j}{h_j}. \tag{14}$$

We update the remaining weights as

$$\boldsymbol{W}_{\mathbb{T}_j} \leftarrow \boldsymbol{W}_{\mathbb{T}_j} - e^q_j \ell_j - e^d_j \ell^F_j, \tag{15}$$

where ℓ_j is the error-propagation direction in Equation 3 and ℓ^F_j is the modified direction for the code displacement error.

Forget and retain statistics in Error compensation. We define the cross second moments on forget and retain data between input coordinate j and the coordinates in $\mathbb{T}_j$ as

$$f_j = (\boldsymbol{H}_F)_{\mathbb{T}_j,j}, \qquad r_j = (\boldsymbol{H}_R)_{\mathbb{T}_j,j}. \tag{16}$$

Let

$$\boldsymbol{M}_j = \boldsymbol{U}^\top_{\mathbb{T}_j,\mathbb{T}_j}\boldsymbol{U}_{\mathbb{T}_j,\mathbb{T}_j}, \tag{17}$$

which equals the inverse of the principal submatrix of $\boldsymbol{H}_R$ indexed by $\mathbb{T}_j$. The direction ℓ_j is proportional to $\boldsymbol{M}_j r_j$ and therefore depends only on retain statistics.

Decomposition of the propagation direction. We decompose the direction ℓ_j into a component b_j orthogonal to f_j and the remaining component p_j:

$$\ell_j = b_j + p_j, \qquad b_j^\top f_j = 0, \qquad p_j = \frac{\ell_j^\top f_j}{f_j^\top \boldsymbol{M}_j f_j}\boldsymbol{M}_j f_j. \tag{18}$$

The component p_j is the part of ℓ_j that correlates with the output change of the code displacement at coordinate j on forget data, so propagating the code displacement error along p_j can reduce the unlearning effect of the assigned code. The modified direction ℓ^F_j therefore keeps b_j and attenuates p_j, balancing the unlearning effect against the preservation of the layer output on retain data. We determine the attenuation of p_j by three factors, which depend on the output row, the alignment of the forget and retain statistics, and the input coordinate.

Direction and coordinate factors. Two of the three factors compare the forget and retain statistics, one over the whole direction and one at each coordinate $t \in \mathbb{T}_j$:

$$\cos^2\theta_j = \frac{(f_j^\top \boldsymbol{M}_j r_j)^2}{(f_j^\top \boldsymbol{M}_j f_j)(r_j^\top \boldsymbol{M}_j r_j)}, \qquad \eta_t = \frac{|\ell_{j,t}|}{|p_{j,t}| + |\ell_{j,t}|}\,\frac{\bar{E}^R_t}{\bar{E}^F_t + \bar{E}^R_t}, \tag{19}$$

where $\bar{E}_t^F$ and $\bar{E}_t^R$ denote the contributions $E_t^F = |f_{j,t}(\boldsymbol{M}_j f_j)_t|$ and $E_t^R = |r_{j,t}(\boldsymbol{M}_j r_j)_t|$ of coordinate t, normalized separately over $\mathbb{T}_j$. The first factor of η_t strengthens the attenuation at coordinates where $|p_{j,t}|$ is large relative to $|\ell_{j,t}|$. The second factor of η_t is the share of the retain contribution at coordinate t. The factor $\cos^2\theta_j$ is the squared cosine similarity between f_j and r_j in the inner product defined by $\boldsymbol{M}_j$. Its value equals the share of p_j in the second moment of the output change on retain data induced by the update along ℓ_j.

Modified direction. Combining the three factors, the modified direction for output row i has entries

$$\ell_{j,t}^F = b_{j,t} + (1 - \bar{s}_i^{\text{out}}) \cos^2\theta_j \, \eta_t p_{j,t}. \tag{20}$$

Each factor of the coefficient of $p_{j,t}$ lies in $[0, 1]$, so $\ell_{j,t}^F$ falls between $b_{j,t}$ and $\ell_{j,t}$. The coefficient is small, and $p_{j,t}$ is strongly attenuated, in output rows with a large output score, when f_j and r_j are weakly aligned, and at coordinates where the forget contribution dominates or $|p_{j,t}|$ is large relative to $|\ell_{j,t}|$. The attenuation therefore concentrates on the parts of p_j associated with forget data and keeps the parts that account for the output change on retain data.

5 Experimental results

5.1 Experimental setup

Benchmarks and models. We evaluate our method on MUSE (Shi et al., 2025) and TOFU (Maini et al., 2024). On MUSE, we use the NEWS and BOOKS datasets with Llama-2-7B. On TOFU, we use the forget01 and forget05 splits with Llama-2-7B and Llama-3.2-1B. In each setting, the retrained model is trained only on the retain data.

Evaluation metrics. On MUSE, we report memorization on the forget data (KnowMem and VerbMem), privacy leakage (PrivLeak), and model utility on the retain data (MU). On TOFU, we report forget quality (FQ), model utility (MU), and truth ratio (TR). Our evaluation target is agreement with the retrained model rather than the best value of any single metric. For readability, we summarize this agreement with the normalized retrain distance (NRD), which averages the absolute deviations from the full-precision retrained model over the reported metrics after dividing each deviation by the value range of its metric. A lower NRD indicates closer agreement with the full precision retrained model, the reference behavior that unlearning aims to match. The Appendix A.2.5 defines NRD and the value range of each metric.

Implementation details. We compare against GA (Jang et al., 2023), NPO (Zhang et al., 2024), and SimNPO (Fan et al., 2025), each combined with the Gradient Descent on the Retain Set (GDR) retain loss, following Zhang et al. (2025). On MUSE, we apply SURE (Zhang et al., 2025) to each of the three methods to obtain quantization-robust variants. On TOFU, we compare against NPO and SimNPO. We report each unlearned model and the retrained model at full precision and after 4-bit weight-only post-training quantization. Our method produces a 4-bit model in a single stage and therefore has no full-precision counterpart. The hyperparameters of all methods, including ours, are set separately for each setting. The Appendices A.2.2–A.2.4 lists these hyperparameters together with the quantization configuration.

5.2 Main Results on MUSE

Table 1 reports the results on MUSE. Consistent with Zhang et al. (2025), we observe that unlearned models close to the retrained model at full precision recover part of the forgotten knowledge after 4-bit quantization. Quantization raises the VerbMem of GA, NPO, and SimNPO on both corpora, most sharply on BOOKS, where the VerbMem of SimNPO grows from 16.35 to 91.08. On NEWS, quantization further shifts the PrivLeak of GA and SimNPO from large positive to large negative values, both far from the value of the retrained model. SURE suppresses this recovery in memorization and keeps KnowMem and VerbMem close to the retrained model, but its PrivLeak remains far below the retrained value on both corpora, and its MU drops below that of the retrained model. In contrast, our method is closest to the retrained model in KnowMem and VerbMem on NEWS and in PrivLeak on BOOKS, where its PrivLeak of 4.11 approaches the retrained value. Compared with

Table 1: Results on MUSE at full precision and 4-bit precision. Retrain denotes the retrained model. NRD denotes the normalized retrain distance to the full-precision retrained model, with each metric normalized by its range across the methods in this table. A lower NRD indicates closer agreement with the full precision retrained model, the reference behavior that unlearning aims to match. Bold indicates the lowest NRD among 4-bit models excluding the retrained reference.

bit	Method	News					Books				
		KnowMem	VerbMem	PrivLeak	MU	NRD ↓	KnowMem	VerbMem	PrivLeak	MU	NRD ↓
FP	Retrain	32.42	20.50	-0.06	55.69	-	30.33	14.53	-0.14	68.98	-
	GA	50.56	18.91	104.95	42.08	0.21	26.89	21.49	-58.15	53.29	0.14
	NPO	31.51	21.72	-60.58	34.07	0.13	34.77	25.46	-60.67	62.02	0.14
	SimNPO	52.96	21.48	104.51	40.33	0.22	29.08	16.35	-58.60	58.52	0.11
4 bit	Retrain	34.65	17.47	-0.82	43.37	0.05	27.85	14.14	-4.64	59.20	0.04
	GA	56.71	30.77	-98.32	44.60	0.23	32.87	47.63	-56.17	47.88	0.22
	NPO	34.58	22.09	-61.54	33.09	0.14	39.05	68.64	-61.83	56.18	0.27
	SimNPO	52.43	26.34	-86.97	39.61	0.21	39.73	91.08	-58.60	50.91	0.34
	GA+SURE	33.53	20.19	-95.66	31.65	0.18	32.71	13.34	-56.68	50.96	0.13
	NPO+SURE	35.55	18.04	-99.03	29.00	0.20	33.64	16.72	-59.46	45.21	0.15
	SimNPO+SURE	34.68	19.97	-99.17	35.37	0.18	36.39	17.36	-68.51	53.95	0.15
	Ours	32.97	20.68	-67.20	37.30	**0.13**	26.10	16.03	4.11	47.30	**0.07**

Table 2: Results on TOFU with Llama-2-7B and Llama-3.2-1B. Retrain denotes the retrained model. FQ, MU, TR, and NRD denote forget quality, model utility, truth ratio, and the normalized retrain distance to the full-precision retrained model. Bold indicates the lowest NRD among 4-bit models excluding the retrained reference.

Bit	Method	LLama 2 7B								LLama 3-2 1B							
		Forget01				Forget05				Forget01				Forget05			
		FQ	MU	TR	NRD ↓	FQ	MU	TR	NRD ↓	FQ	MU	TR	NRD ↓	FQ	MU	TR	NRD ↓
FP	Retrain	1.00	0.63	0.68	-	1.00	0.63	0.67	-	1.00	0.60	0.65	-	1.00	0.60	0.63	-
	NPO	0.92	0.53	0.70	0.07	0.79	0.57	0.71	0.10	0.99	0.57	0.66	0.02	0.92	0.55	0.63	0.04
	SimNPO	0.77	0.44	0.74	0.16	0.27	0.40	0.74	0.34	0.92	0.57	0.64	0.04	0.96	0.59	0.63	0.02
4 bit	Retrain	0.99	0.55	0.70	0.04	1.00	0.58	0.68	0.02	0.92	0.54	0.68	0.06	0.92	0.53	0.64	0.05
	NPO	0.92	0.46	0.71	0.09	0.63	0.56	0.71	0.16	0.77	0.49	0.68	0.12	0.47	0.50	0.64	0.21
	SimNPO	0.58	0.45	0.75	0.22	0.27	0.39	0.74	0.35	0.58	0.51	0.64	0.17	0.09	0.51	0.61	0.34
	Ours	0.99	0.56	0.64	**0.04**	0.99	0.56	0.68	**0.03**	0.99	0.50	0.67	**0.04**	0.97	0.49	0.63	**0.05**

the SURE variants, our method is closer to the retrained model on all four metrics on NEWS and in PrivLeak on both corpora. NRD, which summarizes these deviations, is lowest for our method among 4-bit unlearning methods and does not exceed that of any full-precision unlearning baseline on either dataset.

5.3 RESULTS ON TOFU

Table 2 reports the results on TOFU. We observe that quantization lowers the FQ of NPO and SimNPO in most settings, with the largest decrease on Llama-3.2-1B, consistent with the knowledge recovery reported previously (Zhang et al., 2025). Our method attains FQ of at least 0.97 in all four settings, which exceeds that of quantized NPO and SimNPO, with equal or higher MU on Llama-2-7B and comparable MU on Llama-3.2-1B. The FQ, MU, and TR of our method differ from those of the quantized retrained model by at most 0.07 across the four settings. NRD summarizes these trends, and our method attains the lowest NRD among the 4-bit unlearning methods for both Llama-2-7B and Llama-3.2-1B. On Llama-3.2-1B, our method attains higher FQ than the quantized retrained model and an NRD no higher than that of the quantized retrained model. These results indicate that, for both evaluated model sizes, the 4-bit model of our method agrees with the full-precision retrained model comparably to the retrained model quantized to the same precision.

5.4 ABLATION STUDY

Table 3 examines the contributions of code selection, error compensation, and κ on MUSE. These components determine which weights to modify, how to compensate for the resulting error, and how far to displace the codes, respectively. We remove code selection (w/o selection), replace our compensation with the standard GPTQ compensation (w/o compensation), or keep the displaced

Table 3: Ablation study on MUSE. Each variant removes one component of the proposed procedure while keeping the other unchanged. Bold indicates the lowest NRD among 4-bit models excluding the retrained reference.

Bit	Method	NEWS dataset					BOOKS dataset				
		KnowMem	VerbMem	PrivLeak	MU	NRD ↓	KnowMem	VerbMem	PrivLeak	MU	NRD ↓
FP	Retrain	32.42	20.50	-0.06	55.69	-	30.33	14.53	-0.14	68.98	-
	Base Model	57.89	64.43	-99.81	55.52	0.29	47.12	99.70	-57.34	69.13	0.33
4bit	Retrain	34.65	17.47	-0.82	43.37	0.05	27.85	14.14	-4.64	59.20	0.04
	w/o selection	28.80	23.43	-98.64	31.48	0.19	2.80	11.96	-64.87	5.59	0.32
	w/o compensation	51.56	39.17	-95.35	45.17	0.23	22.48	22.75	-59.06	32.87	0.21
	w/o κ	39.79	28.55	-90.13	37.52	0.19	30.43	36.20	-24.70	51.99	0.13
	Ours	32.97	20.68	-67.20	37.30	**0.13**	26.10	16.03	4.11	47.30	**0.07**

positions and replace each displacement magnitude with a random in the same layer (w/o κ). Further details of the ablation settings are provided in Appendix A.2.6.

Contribution of code selection. Without code selection, the displaced codes no longer concentrate on the weights that the intervention score associates with the forget data. KnowMem and VerbMem move farther from the retrained model, and MU decreases on both datasets, falling from 47.30 to 5.59 on BOOKS, while PrivLeak remains near the level of the base model. These results indicate that choosing the displaced positions by the intervention score moves the forgetting metrics toward the retrained model while preserving utility on the retain data.

Contribution of error compensation. Without our compensation, the standard GPTQ undoes the forgetting introduced by code selection. GPTQ treats the code displacement as an error to be removed and propagates it to the remaining weights to recover the layer output, which also recovers the output on forget data. PrivLeak returns to the base-model level on both datasets, and on NEWS, KnowMem and VerbMem rise from 32.97 and 20.68 to 51.56 and 39.17, respectively.

Contribution of κ. With random displacement magnitudes, KnowMem and VerbMem increase from 32.97 and 20.68 to 39.79 and 28.55 on NEWS. On BOOKS, VerbMem rises from 16.03 to 36.20 and PrivLeak moves from 4.11 to -24.70, back toward the base model, although KnowMem becomes closer to retraining. MU remains similar on NEWS and is higher on BOOKS, indicating that utility alone does not capture the effect of uncontrolled code displacement. Random displacements thus weaken the intended forgetting even when code selection remains active. Without κ, displacement magnitudes no longer follow the strength selected for each setting, and the model departs from the balance between forgetting and retained utility that the selected κ achieves.

6 Conclusion

In this work, we introduced a new formulation of quantization-robust machine unlearning that treats quantization itself as the unlearning operator rather than as a deployment step after unlearning. Instead of first modifying continuous model parameters and then applying quantization, our method directly constructs a low-bit model with unlearning-oriented behavior through score-guided discrete code assignment based on forget and retain activation statistics.

We further showed that codes displaced according to the forget-related criterion introduce an error distinct from ordinary quantization error, and that compensating it with the standard reconstruction update restores the forgotten knowledge. By separating the two errors and propagating the displacement error only partially along forget-correlated directions, our method preserves the intended forgetting without gradient computation or fine-tuning.

Experiments on MUSE, 4-bit quantization moves the conventional baselines back toward the original model and leaves the privacy leakage of the quantization-robust baselines far from the retrained reference, whereas our method moves both memorization and privacy leakage toward the retrained reference. On TOFU, our method attains higher forget quality than the quantized baselines and, for both evaluated model sizes, agrees with the full-precision retrained reference comparably to the retrained reference quantized to the same precision. Across both benchmarks, these trends are consistent with the lowest NRD among the 4-bit unlearning methods. Quantization, which prior work identifies as a source of knowledge recovery after unlearning, thus serves as a means of performing unlearning at 4-bit precision on the evaluated benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools to edit the text for readability and to identify related work. We have not used generative AI tools to generate research ideas, design experiments, implement methods, interpret results, generate synthetic datasets, or write new text for any section of the paper. We wrote all sections of the paper, and the AI-assisted edits covered grammar, wording, and consistency of terminology in the text we wrote. We checked each edited sentence against our derivations and experimental results before including it in the paper. We read each paper identified with the aid of these tools and verified that it supports the specific claim to which the citation is attached. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 defines the intervention score, the score-guided code assignment, and the unlearning-aware error compensation. Algorithm 1 in Appendix A.1.6 summarizes the full procedure. Appendix A.1 specifies the block structure, the coordinate order, the damping, the activation statistics, the normalization operators, and the implementation of the error compensation. Appendix A.1.5 gives the derivations of the Cholesky-factor identities and the proofs of Propositions 1–3. Appendix A.2.1 lists the models, the publicly released checkpoints, and the calibration data for each setting. Appendix A.2.2 lists the quantization configuration of our method, the baselines, and the retrained model. Tables 5, 6, and 7 list the hyperparameters of the baselines and the intervention strength κ of our method. Appendix A.2.5 defines the evaluation metrics and the normalized retrain distance. Appendices A.2.6 and A.2.7 specify the ablation variants and the runtime measurement, respectively.

REFERENCES

João Vitor Boer Abitante, Joana Meneguzzo Pasquali, Luan Fonseca Garcia, Ewerton de Oliveira, Thomas da Silva Paula, Rodrigo C. Barros, and Lucas S. Kupssinskü. Quantization-robust llm unlearning via low-rank adaptation. *arXiv preprint arXiv:2602.13151*, 2026.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

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

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

Sungmin Cha, Sungjun Cho, Dasol Hwang, and Moontae Lee. Towards robust and parameter-efficient knowledge unlearning for llms. In *International Conference on Learning Representations*, volume 2025, pp. 19276–19298, 2025.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36:4396–4429, 2023.

Xin Ding, Xiaoyu Liu, Zhijun Tu, Yun Zhang, Wei Li, Jie Hu, Hanting Chen, Yehui Tang, Zhiwei Xiong, Baoqun Yin, et al. Cbq: Cross-block quantization for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 7056–7075, 2025.

Vineeth Dorna, Anmol Mekala, Wenlong Zhao, Andrew McCallum, Zico Kolter, Zachary Lipton, and Pratyush Maini. Openunlearning: Accelerating llm unlearning via unified benchmarking of methods and metrics. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-1458. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/

3e4a38f228427ab819ba7899003a44b1-Paper-Datasets_and_Benchmarks_Track.pdf.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12284–12303. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/egiazarian24a.html.

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

Chongyu Fan, Jiancheng Liu, Licong Lin, Jinghan Jia, Ruiqi Zhang, Song Mei, and Sijia Liu. Simplicity prevails: Rethinking negative preference optimization for LLM unlearning. In *Advances in Neural Information Processing Systems*, 2025.

Jack Foster, Stefan Schoepf, and Alexandra Brintrup. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 12043–12051, 2024.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9301–9309. IEEE, 2020.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3832–3842. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/guo20c.html.

Yihuai Hong, Yuelin Zou, Lijie Hu, Ziqian Zeng, Di Wang, and Haiqin Yang. Dissecting fine-tuning unlearning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3933–3941, 2024.

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 (Volume 1: Long Papers)*, pp. 14389–14408, 2023.

Abdullah Ahmad Khan and Ferdous Sohel. Durableun: Quantization-induced recovery attacks in machine unlearning. *arXiv preprint arXiv:2605.02196*, 2026.

Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. {BRECQ}: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=POWv6hDd9XH.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *Proceedings of Machine Learning and Systems*, volume 6, 2024.

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

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025b.

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

Himanshu Mishra and Kanwal Mehreen. Quail: Quantization aware unlearning for mitigating misinformation in llms. *arXiv preprint arXiv:2601.15538*, 2026.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pp. 7197–7206. PMLR, 2020.

Ravi Ranjan, Olivera Kotevska, and Agoritsa Polyzou. Forgetting only what matters: Layer-selective unlearning toward robust llms. *arXiv preprint arXiv:2609.10439*, 2026.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024.

William Shen, Xinchi Qiu, Meghdad Kurmanji, Alexandru-Andrei Iacob, Lorenzo Sani, Yihong Chen, Nicola Cancedda, and Nicholas Lane. Llm unlearning via neural activation redirection. *Advances in Neural Information Processing Systems*, 38:44253–44290, 2025.

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

Yujia Tong, Yuze Wang, Jingling Yuan, and Chuang Hu. Robust machine unlearning for quantized neural networks via adaptive gradient reweighting with similar labels. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20603–20612. IEEE, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

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

Qizhou Wang, Bo Han, Puning Yang, Jianing Zhu, Tongliang Liu, and Masashi Sugiyama. Towards effective evaluations and comparisons for llm unlearning methods. In *International Conference on Learning Representations*, volume 2025, pp. 23327–23355, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Yuanshun Yao, Xiaojun Xu, and Yang Liu. Large language model unlearning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 105425–105475. Curran Associates, Inc., 2024. doi: 10.52202/079017-3346. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/be52acf6bccf4a8c0a90fe2f5cfcead3-Paper-Conference.pdf.

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

Xiaojian Yuan, Tianyu Pang, Chao Du, Kejiang Chen, Weiming Zhang, and Min Lin. A closer look at machine unlearning for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 49483–49508, 2025.

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

Zhiwei Zhang, Fali Wang, Xiaomin Li, Zongyu Wu, Xianfeng Tang, Hui Liu, Qi He, Wenpeng Yin, and Suhang Wang. Catastrophic failure of llm unlearning via quantization. In *International Conference on Learning Representations*, 2025.

A Appendix

A.1 Algorithm Details

A.1.1 Quantization Procedure

Block structure. We partition the input coordinates of each linear layer into consecutive blocks $\mathbb{B}_1, \ldots, \mathbb{B}_G$ of B coordinates and keep only the second moments within each block:

$$(\boldsymbol{H}_D)_{jj'} \leftarrow \begin{cases} (\boldsymbol{H}_D)_{jj'}, & j, j' \in \mathbb{B}_g \text{ for some } g, \\ 0, & \text{otherwise,} \end{cases} \qquad D \in \{F, R\}. \tag{21}$$

Each statistic then consists of the diagonal blocks $\boldsymbol{H}_D^{(g)} = (\boldsymbol{H}_D)_{\mathbb{B}_g,\mathbb{B}_g}$. The error at coordinate $j \in \mathbb{B}_g$ therefore propagates only to the coordinates of $\mathbb{B}_g$. Accordingly, $\mathbb{T}_j \subset \mathbb{B}_g$ denotes the coordinates of $\mathbb{B}_g$ that remain unquantized after j, and $\boldsymbol{U}$, $\boldsymbol{M}_j$, f_j, and r_j are computed from $\boldsymbol{H}_F^{(g)}$ and $\boldsymbol{H}_R^{(g)}$.

Coordinate order. Within block $\mathbb{B}_g$, we quantize the coordinates in the order $j_1, \ldots, j_B$ that satisfies $H_{R,j_1j_1} \geq \cdots \geq H_{R,j_Bj_B}$. Let $\boldsymbol{P}_g$ denote the corresponding permutation matrix. We compute the Cholesky factor in the permuted order, $(\boldsymbol{P}_g \boldsymbol{H}_R^{(g)} \boldsymbol{P}_g^\top)^{-1} = \boldsymbol{U}^\top \boldsymbol{U}$, and permute the weights and $\boldsymbol{H}_F^{(g)}$ in the same way. The order therefore determines $\mathbb{T}_j$, ℓ_j, and the diagonal entry U_{jj} in the input score. After the block is quantized, we apply $\boldsymbol{P}_g^\top$ to restore the original coordinate order.

Damping. Before the factorization, we replace $\boldsymbol{H}_R^{(g)}$ with $\boldsymbol{H}_R^{(g)} + \lambda \bar{d}_g \boldsymbol{I}$, where $\bar{d}_g = \frac{1}{B}\sum_{j \in \mathbb{B}_g} |H_{R,jj}|$, to stabilize the inversion and the Cholesky factorization. The inversion adds a small multiple of the identity to this matrix and increases the multiple tenfold until the Cholesky factorization succeeds. We do not add either term to $\boldsymbol{H}_F^{(g)}$, which the procedure does not invert.

A.1.2 Activation Statistics

For split $D \in \{F, R\}$, let $\boldsymbol{x}_n$ and $\boldsymbol{y}_n$ denote the input and output of a linear layer at token n of the calibration sequences, and let N_D denote the number of these tokens. We compute the statistics as token averages:

$$\boldsymbol{H}_D = \frac{1}{N_D} \sum_{n=1}^{N_D} \boldsymbol{x}_n \boldsymbol{x}_n^\top, \qquad u_{D,i} = \frac{1}{N_D} \sum_{n=1}^{N_D} y_{n,i}^2, \tag{22}$$

Table 4: Normalization and index set for each quantity with the bar notation.

Quantity	Normalization	Index set
$\bar{u}_{F,i}$	Mean	Output coordinates of the layer
$\bar{s}_i^{\text{out}}$	Quantile	Output coordinates of the layer
Logarithms in s_j^{sep} and s_j^{joint}	Robust	Input coordinates of the layer
$\bar{s}_j^{\text{sep}}$, $\bar{s}_j^{\text{joint}}$	Quantile	Input coordinates of the layer
$\bar{E}_t^F$, $\bar{E}_t^R$	Quantile without clipping	$\mathbb{T}_j$

where $\boldsymbol{H}_D$ keeps only the entries within the blocks of Equation 21. We quantize the linear layers sequentially and compute the statistics of each layer on the model whose preceding layers are already quantized. The output $\boldsymbol{y}_n$ uses the weights of the current layer before its quantization.

The retain statistic $\boldsymbol{H}_R$ takes the place of the GPTQ Hessian $\boldsymbol{H} = 2\boldsymbol{X}\boldsymbol{X}^\top$ computed on the retain calibration data. The two matrices satisfy $\boldsymbol{H} = 2N_R\boldsymbol{H}_R$, so the Cholesky factor of $\boldsymbol{H}^{-1}$ equals $\boldsymbol{U}/\sqrt{2N_R}$. The scaled error e_j grows and ℓ_j shrinks by the same factor, which leaves the update in Equation 3 unchanged. The factors $\cos^2\theta_j$ and η_t are ratios and do not depend on this factor. The normalization of the input score removes the constant shift that the factor adds to $\log U_{jj}^2$.

A.1.3 Normalization Operators

The bar notation in the main text denotes one of three normalizations, and Table 4 lists the normalization and the index set for each quantity.

Mean normalization. For the forget output statistic, we divide each entry by the mean over the output coordinates of the layer:

$$\bar{u}_{F,i} = \frac{u_{F,i}}{\frac{1}{d_{\text{out}}}\sum_{i'=1}^{d_{\text{out}}} u_{F,i'}}. \tag{23}$$

Robust standardization. For a vector $\boldsymbol{a}$ over the input coordinates of a layer, we define

$$\bar{a}_j = \frac{a_j - \operatorname{med}(\boldsymbol{a})}{\operatorname{IQR}(\boldsymbol{a})}, \tag{24}$$

where med and IQR denote the median and the interquartile range over all input coordinates of the layer. When the interquartile range is zero, we divide by the standard deviation instead. We apply this operator to $\log H_{F,jj}$, $\log U_{jj}^2$, and their sum in Equation 9, and add a small constant inside each logarithm to keep it finite.

Quantile normalization. For a nonnegative vector $\boldsymbol{s}$, we define

$$\bar{\boldsymbol{s}} = \min\left(\frac{\boldsymbol{s}}{q_{0.95}(\boldsymbol{s})},\, 1\right), \tag{25}$$

where the minimum applies to each entry and $q_{0.95}(\boldsymbol{s})$ is the 95th percentile of the positive entries of $\boldsymbol{s}$. The operator maps the largest 5% of the positive entries to 1, so a few extreme coordinates do not set the scale of the other entries. We apply this operator to s^{out} over the output coordinates of the layer and to s^{sep} and s^{joint} over the input coordinates of the layer, which gives $\boldsymbol{S} \in [0,1]^{d_{\text{out}} \times d_{\text{in}}}$. For the contributions in Equation 19, we divide E_t^F and E_t^R by the 95th percentile of their positive entries over $\mathbb{T}_j$ without the clipping at 1. When the normalizing statistic of a vector is zero, each operator returns a zero vector.

A.1.4 Error Compensation Implementation

Update for all output rows. With the row index restored, the update in Equation 15 for output row i reads

$$w_{i,t} \leftarrow w_{i,t} - e_{i,j}^q \ell_{j,t} - e_{i,j}^d b_{j,t} - (1 - \bar{s}_i^{\text{out}})\cos^2\theta_j\, e_{i,j}^d\, \eta_t\, p_{j,t}, \qquad t \in \mathbb{T}_j. \tag{26}$$

The factors $\ell_{j,t}$, $b_{j,t}$, and $\eta_t p_{j,t}$ do not depend on the output row, and the remaining factors do not depend on t. The update for all output rows at coordinate j is therefore a sum of three outer products, and we apply it without forming ℓ_j^F for each row.

Degenerate cases. When $f_j^\top \boldsymbol{M}_j f_j \le \epsilon$, the component p_j is undefined, and we propagate the code displacement error along ℓ_j. When $r_j^\top \boldsymbol{M}_j r_j \le \epsilon$, we set $\cos^2\theta_j = 0$, so the code displacement error propagates along b_j. When no code at coordinate j is displaced, $e^d_{i,j} = 0$ for all rows, and we skip the computation of the modified direction. The last coordinate of a block has an empty $\mathbb{T}_j$ and propagates no error.

Batched computation. As in GPTQ (Frantar et al., 2023), we process the coordinates of a block in consecutive sub-blocks and apply the updates to the coordinates after a sub-block once the sub-block is quantized. For a sub-block with first coordinate j_0, every $\mathbb{T}_j$ in the sub-block is a subset of $\mathbb{T}_{j_0}$. Let $\boldsymbol{E}_j \in \{0,1\}^{|\mathbb{T}_{j_0}|\times|\mathbb{T}_j|}$ denote the columns of the identity matrix indexed by $\mathbb{T}_j$, so $\boldsymbol{E}_j v$ extends a vector v on $\mathbb{T}_j$ by zeros to $\mathbb{T}_{j_0}$ and $\boldsymbol{E}_j^\top$ restricts a vector on $\mathbb{T}_{j_0}$ to $\mathbb{T}_j$. The principal submatrix satisfies $\boldsymbol{U}_{\mathbb{T}_j,\mathbb{T}_j} = \boldsymbol{E}_j^\top \boldsymbol{U}_{\mathbb{T}_{j_0},\mathbb{T}_{j_0}} \boldsymbol{E}_j$, which gives

$$\boldsymbol{M}_j f_j = \boldsymbol{E}_j^\top \boldsymbol{U}_{\mathbb{T}_{j_0},\mathbb{T}_{j_0}}^\top \left(\boldsymbol{E}_j \boldsymbol{E}_j^\top\right) \boldsymbol{U}_{\mathbb{T}_{j_0},\mathbb{T}_{j_0}} \left(\boldsymbol{E}_j f_j\right), \tag{27}$$

and the same identity holds for $\boldsymbol{M}_j r_j$. The matrix $\boldsymbol{E}_j \boldsymbol{E}_j^\top$ sets the entries outside $\mathbb{T}_j$ to zero. We stack $\boldsymbol{E}_j f_j$ and $\boldsymbol{E}_j r_j$ of all coordinates in the sub-block as columns, multiply the stacked matrix by $\boldsymbol{U}_{\mathbb{T}_{j_0},\mathbb{T}_{j_0}}$, apply $\boldsymbol{E}_j \boldsymbol{E}_j^\top$ to each column, and multiply the result by $\boldsymbol{U}_{\mathbb{T}_{j_0},\mathbb{T}_{j_0}}^\top$. The batched computation therefore returns the same values as the computation for each coordinate, with two matrix multiplications per sub-block.

A.1.5 Derivations

Throughout this subsection, we fix a block, index its coordinates by $1,\dots,B$ in the quantization order, and write $\boldsymbol{H} = \boldsymbol{H}_R^{(g)}$ with $\boldsymbol{H}^{-1} = \boldsymbol{U}^\top \boldsymbol{U}$. Then $\mathbb{T}_j = \{j+1,\dots,B\}$, and we treat ℓ_j, f_j, r_j, b_j, and p_j as column vectors on $\mathbb{T}_j$.

Properties of the Cholesky factor. For a trailing index set $\mathbb{S} = \{j',\dots,B\}$, the principal submatrix satisfies $(\boldsymbol{H}_{\mathbb{S},\mathbb{S}})^{-1} = \boldsymbol{U}_{\mathbb{S},\mathbb{S}}^\top \boldsymbol{U}_{\mathbb{S},\mathbb{S}}$, which underlies the sequential update of GPTQ (Frantar et al., 2023). Setting $\mathbb{S} = \mathbb{T}_j$ gives the first identity below. Setting $\mathbb{S} = \{j\} \cup \mathbb{T}_j$ and comparing the (j,j) entries and the (t,j) entries for $t \in \mathbb{T}_j$ with the block inverse of $\boldsymbol{H}_{\mathbb{S},\mathbb{S}}$ gives the second and third identities:

$$\boldsymbol{M}_j = (\boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j})^{-1}, \qquad \frac{1}{U_{jj}^2} = H_{jj} - \boldsymbol{H}_{j,\mathbb{T}_j} \boldsymbol{M}_j \boldsymbol{H}_{\mathbb{T}_j,j}, \qquad \ell_j = -U_{jj} \boldsymbol{M}_j r_j. \tag{28}$$

The second identity is the Schur complement of $\boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j}$, which equals $\min \mathbb{E}_R\big[(x_j - {}^\top \boldsymbol{x}_{\mathbb{T}_j})^2\big]$, the second moment on retain data of the part of x_j that $\boldsymbol{x}_{\mathbb{T}_j}$ cannot linearly predict.

Output changes of an update. The update of an error e along a direction v changes the layer output at token n by $-e\, v^\top \boldsymbol{x}_{n,\mathbb{T}_j}$. On retain data, this change has the second moment $e^2 v^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} v$. On forget data, its cross second moment with $x_{n,j}$ is proportional to $v^\top f_j$.

Proposition 1. *Among the vectors q with $(\ell_j - q)^\top f_j = 0$, the component p_j minimizes $q^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} q$.*

Proof. The Lagrangian condition of the constrained problem gives $\boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} q = \mu f_j$, so $q = \mu \boldsymbol{M}_j f_j$ by Equation 28. The constraint gives $\mu = \ell_j^\top f_j / (f_j^\top \boldsymbol{M}_j f_j)$, so $q = p_j$. The objective is strictly convex because $\boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j}$ is positive definite, so p_j is the unique minimizer. □

Proposition 2. *The ratio $p_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} p_j / \ell_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} \ell_j$ equals $\cos^2\theta_j$.*

Proof. Let $\mu = \ell_j^\top f_j / (f_j^\top \boldsymbol{M}_j f_j)$, so $p_j = \mu \boldsymbol{M}_j f_j$. The first identity in Equation 28 gives $b_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} p_j = \mu\, b_j^\top f_j = 0$, so $\ell_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} \ell_j = b_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} b_j + p_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} p_j$. The same identity

Algorithm 1 Unlearning with quantization

Require: Original parameters θ_0, calibration data from $\mathbb{D}_F$ and $\mathbb{D}_R$, bit width b, intervention strength κ, block size B, damping ratio λ

Ensure: Low-bit unlearned parameters $\widehat{\theta}_u$

for each linear layer $\boldsymbol{W}$ in forward order **do**
  Compute $\boldsymbol{H}_F$, $\boldsymbol{H}_R$, $u_{F,i}$, and $u_{R,i}$ on the current model ▷ Equation 22
  for each block **do**
    Add the damping term to the block of $\boldsymbol{H}_R$ and order its coordinates
    Compute the Cholesky factor $\boldsymbol{U}$ in the permuted order
  end for
  (i) Compute $\bar{s}^{\text{out}}$, $\bar{s}^{\text{in}}$, $\boldsymbol{S}$, and $\rho_{i,j}$ ▷ Eqs. 6–11
  for each block **do**
    Fix the quantization step Δ_i of each output row
    for each sub-block of consecutive coordinates **do**
      Compute $\boldsymbol{M}_j f_j$ and $\boldsymbol{M}_j r_j$ for all j in the sub-block ▷ Equation 27
      for each coordinate j in the sub-block **do**
        (ii) Compute $q^0_{i,j}$, $\delta_{i,j}$, $q_{i,j}$, and $k_{i,j}$ for all i ▷ Eqs. 12–13
        (iii) Compute $e^q_{i,j}$ and $e^d_{i,j}$ for all i ▷ Equation 14
        if $k_{i,j} \neq 0$ for some i and $f_j^\top \boldsymbol{M}_j f_j > \epsilon$ **then**
          Compute b_j, p_j, $\cos^2\theta_j$, and η_t ▷ Eqs. 18–19
        else
          Set $b_j = \ell_j$ and $p_j = \mathbf{0}$
        end if
        (iv) Update $w_{i,t}$ for $t \in \mathbb{T}_j$ in the sub-block and accumulate the update for the other $t \in \mathbb{T}_j$ ▷ Equation 26
      end for
      Apply the accumulated updates to the coordinates after the sub-block
    end for
    Restore the original coordinate order of the block
  end for
end for
return $\widehat{\theta}_u$

gives $p_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} p_j = (\ell_j^\top f_j)^2/(f_j^\top \boldsymbol{M}_j f_j)$. The third identity gives $\ell_j^\top f_j = -U_{jj} r_j^\top \boldsymbol{M}_j f_j$ and $\ell_j^\top \boldsymbol{H}_{\mathbb{T}_j,\mathbb{T}_j} \ell_j = U_{jj}^2 r_j^\top \boldsymbol{M}_j r_j$. Substituting the two expressions into the ratio yields $\cos^2\theta_j$. □

Proposition 3. *On forget data, the cross second moment between the output change of the code displacement at coordinate j and the output change of its compensation along p_j equals $-(\Delta k_j)^2\, r_j^\top \boldsymbol{M}_j f_j$, and the compensation along b_j contributes zero.*

Proof. The code displacement changes the output at token n by $\Delta k_j\, x_{n,j}$. The compensation along a direction v uses the error $e^d_j = -\Delta k_j / U_{jj}$ and changes the output by $(\Delta k_j/U_{jj})\, v^\top \boldsymbol{x}_{n,\mathbb{T}_j}$. The cross second moment on forget data equals $((\Delta k_j)^2/U_{jj})\, v^\top f_j$. The direction b_j gives zero because $b_j^\top f_j = 0$. The direction p_j gives $p_j^\top f_j = \ell_j^\top f_j = -U_{jj} r_j^\top \boldsymbol{M}_j f_j$ by Equation 28, which yields the stated value. □

When $r_j^\top \boldsymbol{M}_j f_j > 0$, the cross second moment in Proposition 3 is negative, and the compensation along p_j offsets part of the output change of the code displacement on forget data.

A.1.6 Full Algorithm

Algorithm 1 summarizes the procedure, and the labels (i)–(iv) correspond to the four steps in the method overview.

Table 5: Hyperparameters of the baselines on MUSE. Training length denotes the epochs of training or the selected checkpoint. α and γ weight the retain loss and the forget loss, respectively, with $\gamma = 1$ unless the table lists another value, and Threshold denotes the saliency percentile of SURE.

dataset	Method	Training length	Learning rate	α	β	γ	δ	Threshold
NEWS	GA	9.2 epochs	1×10^{-5}	1.0	–	–	–	–
	NPO	0.6 epochs	7×10^{-5}	1.0	0.01	–	–	–
	SimNPO	1epoch	$5 \times 10^{-}5$	1.0	4.5	0.125	0	–
	GA+SURE	1 epoch	5×10^{-5}	1.5	–	–	–	99
	NPO+SURE	1 epoch	1×10^{-4}	50	0.5	–	–	95
	SimNPO+SURE	1 epoch	1×10^{-4}	1.0	4.5	0.125	0	90
BOOKS	GA	3 epochs	5×10^{-6}	100	–	–	–	–
	NPO	12 epochs	1.5×10^{-5}	2.0	0.02	–	–	–
	SimNPO	12 epochs	5×10^{-6}	1.0	3.0	0.125	0	–
	GA+SURE	1 epoch	5×10^{-5}	375	–	–	–	99
	NPO+SURE	1 epoch	1.2×10^{-4}	300	0.05	–	–	90
	SimNPO+SURE	1 epoch	5×10^{-5}	1.0	4.5	0.125	0	90

A.2 EXPERIMENTAL DETAILS

A.2.1 MODELS AND DATA

We run the unlearning baselines and the evaluation with the OpenUnlearning framework (Dorna et al., 2025). On MUSE, we use the target and retrained models released by Shi et al. (2025) for NEWS and BOOKS. The target model of each dataset is fine-tuned on the forget and retain sets, and the retrained model is fine-tuned on the retain set only. On TOFU, we use the Llama-2-7B-Chat and Llama-3.2-1B-Instruct models released with OpenUnlearning. The target models are fine-tuned on the full TOFU data, and the retrained models for forget01 and forget05 are fine-tuned on the retain99 and retain95 splits, respectively. We draw the calibration sequences from the forget and retain sets of each setting, with 64 sequences from each set.

A.2.2 QUANTIZATION CONFIGURATION

We follow the quantization setting of Zhang et al. (2025) for the unlearned baselines and the retrained model. Our method quantizes the target model at the same bit width with block size $B = 128$ and damping ratio $\lambda = 10^{-4}$.

A.2.3 BASELINES

We train all baselines from the target models with OpenUnlearning (Dorna et al., 2025). The retain loss is the negative log-likelihood on retain data, weighted by α. On MUSE, training uses the paged AdamW optimizer with 32-bit states, a constant learning rate, no weight decay, and bf16 precision, with a per-device batch size of 2 and two gradient accumulation steps. We select the hyperparameters and the checkpoint of each baseline that minimize the normalized root mean squared error of VerbMem and KnowMem with respect to the retrained model, without a constraint on MU. For the SURE variants, we train for one epoch and search the learning rate over $\{5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}$. Table 5 lists the selected hyperparameters, whose symbols follow the hyperparameter names of the OpenUnlearning implementations. For SURE, the threshold is the percentile of the module saliency scores above which the modules are updated (Zhang et al., 2025).

On TOFU, we set $\alpha = 1$ in all settings. For Llama-2-7B-Chat, we use the AdamW optimizer with a warmup of one epoch and gradient checkpointing. Table 6 lists the other hyperparameters, where Batch denotes the per-device batch size times the number of gradient accumulation steps.

A.2.4 HYPERPARAMETERS OF OUR METHOD

Table 7 lists the intervention strength κ for each setting.

Table 6: Hyperparameters of the baselines on TOFU. The symbols follow Table 5.

Model	Split	Method	Training length	Batch	Learning rate	β	γ	δ
Llama-2-7B-Chat	forget01	NPO	10 epochs	1×2	1×10^{-5}	0.1	–	–
		SimNPO	5 epochs	1×4	5×10^{-5}	3.0	0.125	0
	forget05	NPO	10 epochs	1×2	1×10^{-5}	0.1	–	–
		SimNPO	10 epochs	1×2	5×10^{-5}	3.0	0.125	0
Llama-3.2-1B-Instruct	forget01	NPO	8 epochs	4×5	2.611×10^{-5}	0.05	0.21	–
		SimNPO	8 epochs	4×5	3×10^{-5}	4.5	24.0	0
	forget05	NPO	9 epochs	4×5	2.5×10^{-5}	0.1414	2.181	–
		SimNPO	9 epochs	4×5	2×10^{-5}	2.0	4.0	0

Table 7: Intervention strength κ of our method.

Benchmark	Model	Setting	κ
MUSE	Llama-2-7B-Chat	NEWS	6.0156
		BOOKS	26.5625
TOFU	Llama-2-7B-Chat	forget01	3.75
		forget05	5.1875
	Llama-3.2-1B-Instruct	forget01	0.4375
		forget05	1.2812

A.2.5 EVALUATION METRICS AND NORMALIZED RETRAIN DISTANCE

For a method m and a metric c in the set $\mathbb{C}$ of reported metrics, let $x_{m,c}$ denote the value of metric c for method m and r_c the value for the full-precision retrained model in the same setting. We define the normalized retrain distance as

$$\mathrm{NRD}_m = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} \frac{|x_{m,c} - r_c|}{R_c}, \tag{29}$$

where R_c is the width of the value range of metric c. NRD weights all reported metrics equally, and the full-precision retrained model has zero NRD by definition. Table 8 lists R_c for each metric.

MUSE. KnowMem, VerbMem, and MU are ROUGE scores reported as percentages (Shi et al., 2025), so their values lie in $[0, 100]$ and $R_c = 100$. PrivLeak compares the AUC of the Min-K% Prob attack in distinguishing forget data from holdout data with the corresponding AUC of the retrained model (Shi et al., 2025):

$$\mathrm{PrivLeak} = \frac{\mathrm{AUC} - \mathrm{AUC_r}}{\mathrm{AUC_r}} \times 100, \tag{30}$$

where AUC and $\mathrm{AUC_r}$ are the AUCs of the evaluated model and the retrained model. We use $K = 40$ for the attack, following the implementation of OpenUnlearning (Dorna et al., 2025). The AUC lies in $[0, 1]$, so PrivLeak lies in $[-100,\ 100(1 - \mathrm{AUC_r})/\mathrm{AUC_r}]$ and has the range width $R_c = 100/\mathrm{AUC_r}$. The retrained model has $\mathrm{AUC_r} = 0.4764$ on NEWS and $\mathrm{AUC_r} = 0.5408$ on BOOKS, which gives $R_c = 209.91$ and $R_c = 184.91$, respectively.

TOFU. The truth ratio R of a question is the ratio of the length-normalized probability of the perturbed answers to that of the paraphrased answer (Maini et al., 2024). FQ is the p-value of a two-sample Kolmogorov–Smirnov test between the truth ratio distributions of the evaluated model and the retrained model on the forget set (Maini et al., 2024). MU is the harmonic mean of nine scores in $[0, 1]$, namely the answer probability, ROUGE-L recall, and truth ratio on the retain set, real authors, and world facts (Maini et al., 2024). TR is the mean of $\min(R, 1/R)$ over the forget questions (Dorna et al., 2025). All three TOFU metrics lie in $[0, 1]$, so they use $R_c = 1$.

Table 8: Range width R_c of each metric in NRD.

Benchmark	Metric	R_c
MUSE	KnowMem, VerbMem, MU	100
	PrivLeak (NEWS)	209.91
	PrivLeak (BOOKS)	184.91
TOFU	FQ, MU, TR	1

Table 9: Components that each ablation variant keeps or replaces.

Variant	Positions	Magnitudes	Compensation	κ
Ours	Score	$\lfloor \rho_{i,j}\delta_{i,j} \rceil$	Modified direction	Selected
w/o selection	Random	Random	Modified direction	Not used
w/o κ	Score	Random	Modified direction	Same as ours
w/o compensation	Score	$\lfloor \rho_{i,j}\delta_{i,j} \rceil$	GPTQ	Reselected

A.2.6 ABLATION VARIANTS

Each variant in Table 3 replaces one component of our method, and Table 9 summarizes the components that each variant keeps or replaces. All variants keep the sign of each displacement equal to the sign of $\delta_{i,j}$.

w/o selection. This variant replaces the score-guided choice of the displaced codes with random selection. In each layer, each code is displaced independently with probability equal to the fraction of codes that our method displaces in the layer. Each displacement magnitude $|k_{i,j}|$ is drawn independently from the distribution of displacement magnitudes of our method in the same layer. The intervention score and κ do not enter the variant. The variant therefore matches the expected number of displacements and their magnitude distribution in each layer, while the positions and magnitudes no longer depend on the intervention score.

w/o κ. This variant determines the positions and signs of the displaced codes by Equation 13 with the value of κ of our method. Each nonzero magnitude $|k_{i,j}|$ is then replaced with a random draw from the distribution of displacement magnitudes of our method in the same layer. The random magnitudes change the subsequent compensation and residuals, so the number of displaced codes differs from that of our method by up to 14%.

w/o compensation. This variant sets $\ell_j^F = \ell_j$, so the code displacement error propagates along the error-propagation direction of GPTQ, and keeps the score-guided code assignment. Replacing the compensation changes the effect of a given κ, so we select κ for this variant by the same sweep as for our method.

A.2.7 RUNTIME

We measure the runtime on an NVIDIA H200 GPU. The runtime of our method covers one run of the procedure, including the computation of activation statistics and the quantization. For each baseline, we report the time of 10 training epochs, following the number of epochs in the reproduction setting of OpenUnlearning (Dorna et al., 2025). We obtain this time as ten times the measured time of one epoch. The baseline times cover only the unlearning stage, and the corresponding sequential pipelines add the quantization time to them.

Comparison. Our method takes less time than 10 training epochs of NPO and SimNPO in all six settings of Table 10, and less time than 10 training epochs of GA in all settings except Llama-3.2-1B-Instruct forget01. On MUSE, our method takes 120 s on NEWS and 126 s on BOOKS, whereas one training epoch of the baselines takes from 138 s to 383 s. Our single-stage procedure, which includes the quantization, therefore takes less time than the unlearning stage alone of each evaluated sequential pipeline.

Table 10: Runtime of one run of our method, which includes the quantization, and of 10 training epochs of each baseline, which excludes the quantization. The baseline times are ten times the measured time of one epoch. On TOFU, we report the runtime of GA for reference only, since we do not evaluate GA on TOFU.

Benchmark	Setting	Ours	GA	NPO	SimNPO
MUSE	NEWS	120 s	1,380 s	2,880 s	2,540 s
	BOOKS	126 s	1,840 s	3,830 s	3,380 s
TOFU	Llama-2-7B-Chat, forget01	97 s	165 s	150 s	144 s
	Llama-2-7B-Chat, forget05	93 s	177 s	258 s	226 s
	Llama-3.2-1B-Instruct, forget01	28 s	24 s	41 s	33 s
	Llama-3.2-1B-Instruct, forget05	30 s	43 s	138 s	101 s

Dependence on model size. On TOFU, the runtime of our method grows with the model size and changes by at most 4 s between the two sizes of the forget set. Our method takes 28 s and 30 s on forget01 and forget05 with Llama-3.2-1B-Instruct, and 97 s and 93 s with Llama-2-7B-Chat. A training epoch of the baselines grows from 1 to 6 steps between the two splits, whereas our method uses at most 64 calibration sequences from each set in every split.