RAMP: RESIDUAL-GUIDED ROUTE WEIGHTING FOR VISION-LANGUAGE UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Suppressing a target answer does not necessarily prevent a vision-language model from revealing the same fact through alternative wording or visual evidence. We introduce RAMP (Residual-Adaptive Multimodal Penalties), a two-stage unlearning method that uses residual answer scores to determine both when and where to apply further forgetting updates. Stage I suppresses target answer spans. Stage II scores verified alternative answers, activates representation penalties when those scores remain high, and allocates their strength across text, image, and fusion representations using gradient sensitivity. Alias suppression and preservation losses jointly constrain the updates. On CLEAR, Stage II reduces held-out paraphrase recovery from 0.181 to 0.072 and indirect recovery from 0.174 to 0.089. On UMU-Bench, RAMP achieves lower remaining target performance and higher utility than the reported unlearning baselines at all three forget rates. Fixed-budget routing controls and baselines with shared supervision distinguish the contribution of adaptive weighting from that of additional data. Unseen attacks, output audits, and public-data substitution further characterize the method's effectiveness and practical requirements. Together, these results support using residual recovery to guide the allocation of unlearning updates.

1 INTRODUCTION

Machine unlearning seeks to remove designated knowledge while preserving a model's useful behavior. For a vision-language model (VLM), this requires selectivity across facts and input modalities. Forgetting a person's identity should prevent identification from a question, a photograph, or a combination of clues, while preserving the ability to describe unrelated attributes such as clothing. The challenge is therefore to change the model's access to a target fact without broadly degrading the representations that support retained tasks.

Most unlearning objectives act on specified answers. Gradient ascent and Negative Preference Optimization (NPO), for example, reduce the likelihood of target text (Jang et al., 2023; Zhang et al., 2024). Yet a fact can survive suppression of its original wording: the model may express it differently or recover it from another input modality. CLEAR and UMU-Bench expose the difficulty of balancing forgetting with preservation across text and images (Dontsov et al., 2025; Wang et al., 2025). Direct-answer accuracy therefore captures only part of the unlearning problem.

We begin with a gap between direct and semantic recovery. On CLEAR, training against target answer spans reduces direct recovery to 0.084, while recovery under held-out paraphrased and indirect questions remains 0.181 and 0.174. We call this remaining ability to express the target fact a *semantic residual*. This gap motivates a training principle: the answers that remain accessible after direct suppression should determine the focus of subsequent updates. A fixed allocation across modalities cannot respond to how residual recovery differs across targets or changes during training.

We introduce **RAMP** (**R**esidual-**A**daptive **M**ultimodal **P**enalties). The method first suppresses target spans, then scores verified alternative answers that express the same fact. When these scores remain high, their gradient sensitivities determine the relative weights of three representation penalties: one on question representations, one on visual features, and one on answer–image interactions. We refer to these intervention sites as the *text*, *image*, and *fusion* routes. This separates two decisions: whether an example needs additional representation changes, and how to distribute those changes across routes. Alternative-answer suppression remains active alongside the weighted penalties; retained examples, harmless same-image questions, and utility distillation constrain both training stages.

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.

RAMP's central contribution is the connection between a residual output score and the allocation of representation penalties. The score determines whether further representation changes are warranted, and its gradient sensitivities determine their relative strength. All parameters remain trainable; the weights specify how to combine losses, without assuming independent knowledge stores for the three routes. We isolate this design through adaptive, equal, shuffled, and single-route controls with fixed initialization, data, losses, and training length. Baselines supplied with the same auxiliary supervision separately assess the contribution of data construction.

Experiments on CLEAR and UMU-Bench support a favorable forgetting–preservation trade-off. On CLEAR, paraphrase and indirect recovery fall to 0.072 and 0.089. Tests with unseen prompts, reserved images, and compositional or multi-turn attacks assess whether the benefit extends beyond the finite training aliases. Output audits distinguish reduced disclosure from refusal or generation failure, while public-data substitution and cost measurements establish the requirements of the evaluated implementation.

Our contributions are:

- **Method:** A two-stage VLM unlearning method that uses residual answer scores to activate and adaptively weight text, image, and fusion penalties.
- **Mechanistic Analysis:** Controlled comparisons that isolate target-dependent weighting and distinguish its benefit from additional target and preservation supervision.
- **Comprehensive Evaluation:** An evaluation spanning held-out factual recovery, cross-modal leakage, unseen attacks, preservation, data requirements, and computational cost.

2 Related Work

Unlearning in language models. Gradient ascent reduces the likelihood of text to be forgotten (Jang et al., 2023), while NPO stabilizes forgetting updates (Zhang et al., 2024). Alternative-label training aims to approximate predictions without the target corpus (Eldan & Russinovich, 2023). These approaches provide the answer-level objectives on which RAMP builds. Our focus is the residual factual recovery that remains after such suppression, particularly when alternative wording and visual evidence provide additional access to the target.

Unlearning in vision-language models. Textual unlearning can improve robustness to both text-only and image–text attacks (Chakraborty et al., 2024). SIU constructs examples from a single image to remove a visual concept (Li et al., 2024). MMUnlearner restricts updates using parameter saliency to remove visual patterns while preserving associated textual knowledge (Huo et al., 2025). SAUCE intervenes on concept-related sparse-autoencoder features at inference (Geng & Li, 2025), and MANU prunes neurons according to their importance on forget and retain data across modalities (Liu et al., 2025). RAMP uses residual answer scores to adapt the strength of representation penalties during full-parameter training. Its intervention is thus conditioned on the model's remaining ability to express the target after direct-answer suppression, with separate penalties for textual, visual, and joint representations.

Distributed and shared representations. Different neuron groups can support the same fact (Chen et al., 2024), while superposition allows multiple concepts to share representational dimensions (Elhage et al., 2022). These observations motivate checking both residual recovery and collateral effects. RAMP treats text, image, and fusion representations as intervention sites, without assuming that they store independent components of a fact. Uniform, shuffled, and single-route controls test the value of its weighting rule.

Evaluating unlearning. TOFU measures factual forgetting and preservation in language models (Maini et al., 2024); CLEAR extends this evaluation to text and images (Dontsov et al., 2025). UMU-Bench measures consistency across modalities (Wang et al., 2025), and FIUBench tests whether forgotten identities remain recoverable under privacy attacks (Ma et al., 2025). We complement benchmark scores with held-out rewordings, images, and compositional or multi-turn prompts, and audit output behavior to distinguish reduced disclosure from refusal or generation failure.

3 Method

RAMP combines direct-answer suppression with representation updates guided by residual recovery (Figure 1). Let $x = (q, I, y^\star)$ denote a question, image, and answer, and let $\mathcal{F}$ and $\mathcal{R}$ be the forget

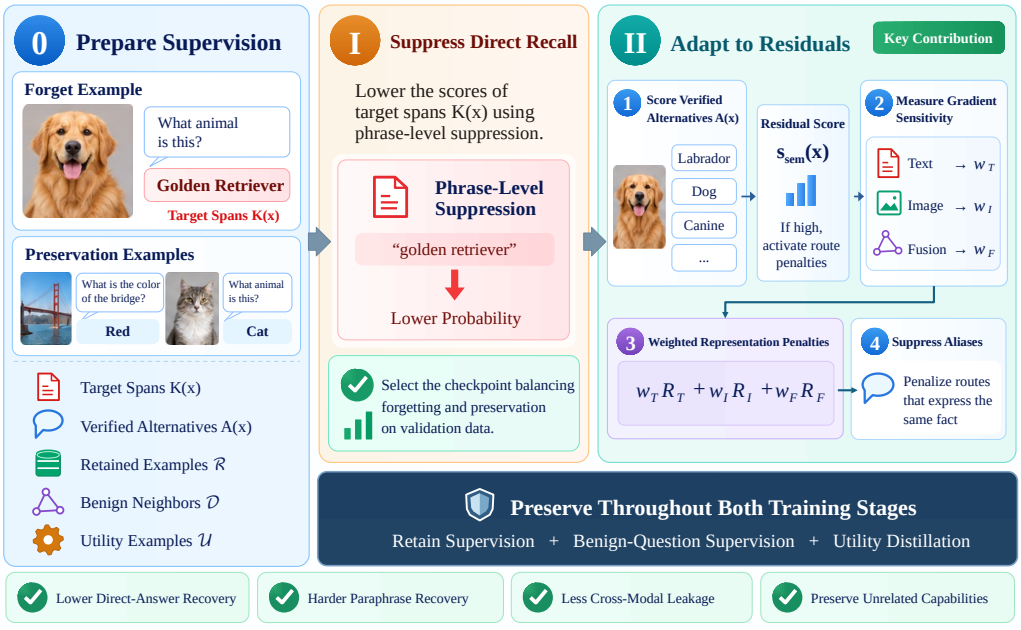


Figure 1: **RAMP overview.** Offline preparation defines what to forget and what to preserve. Stage I suppresses direct answer spans. Stage II uses scores of alternative answers to activate and weight representation penalties, while continuing answer-level suppression. All model parameters remain trainable.

and retain sets. We initialize p_θ from the original VLM p_{θ_0} and update all parameters. A frozen copy of p_{θ_0} provides reference predictions and features.

3.1 Stage 0: Construct Target and Preservation Data

For each forget example, an external language model extracts short answer spans and checks whether removing a span changes or obscures the target fact. Accepted *target spans* form $\mathcal{K}(x)$, termed indispensable knowledge units in the annotation protocol. The model also generates and verifies alternative answers that express the same fact without adding information. These *aliases* form $\mathcal{A}(x)$, a finite training set of equivalent expressions.

Preservation uses three sets: retained examples $\mathcal{R}$, verified same-image questions that do not reveal the target $\mathcal{D}$ (benign neighbors), and a broader utility set $\mathcal{U}$. By default, $\mathcal{U}$ follows the source proportions of the original instruction-tuning mixture; we also evaluate filtered public data. Target-related examples are removed from preservation data. Examples without valid spans are excluded from forget training, and those without valid aliases receive Stage I only; evaluation retains the original target pool. Construction uses `gpt-4o-2024-08-06`. Appendix A specifies prompts, verification, filtering, and fallback rules; Section 4.4 examines their contribution.

3.2 Stage I: Suppress Target Answer Spans

For a phrase $\phi = (t_1, \ldots, t_m)$, let $c_\phi(x)$ be the formatted context preceding the phrase and $t_{<j}$ its token prefix. We score the phrase under teacher forcing and apply an unlikelihood penalty:

$$s_\theta(\phi \mid x) = \frac{1}{m}\sum_{j=1}^{m} \log p_\theta(t_j \mid c_\phi(x), t_{<j}), \qquad u_\theta(\phi, x) = -\log\left(1 - e^{s_\theta(\phi|x)}\right). \tag{1}$$

Here e^{s_θ} is the geometric mean of token probabilities, rather than a sequence probability. Length normalization avoids the automatic decay of a product of token probabilities. Minimizing u_θ reduces the score of the target wording. Stage I averages this penalty over target spans and forget examples:

$$L_{\text{F-kw}} = \mathbb{E}_{x\sim\mathcal{F}} \frac{1}{|\mathcal{K}(x)|} \sum_{k\in\mathcal{K}(x)} u_\theta(k, x). \tag{2}$$

Concentrating the loss on verified spans limits direct suppression to the content selected for removal, while the preservation terms constrain changes to retained behavior.

Both stages combine a forget objective with preservation losses. $L_{\text{R-wCE}}$ is answer cross-entropy on $\mathcal{R}$, with increased weight on verified factual spans. L_{decouple} is answer cross-entropy on $\mathcal{D}$. L_{util} distills the frozen model's next-token distribution on $\mathcal{U}$ using KL divergence at matched response prefixes. With fixed nonnegative coefficients,

$$L^{(1)} = \lambda_1 L_{\text{F-kw}} + \lambda_2 L_{\text{R-wCE}} + \lambda_3 L_{\text{util}} + \lambda_4 L_{\text{decouple}}. \tag{3}$$

Appendix B gives the token-level definitions and averaging rules.

3.3 Stage II: Reduce residual recovery

Stage II separates activation from allocation. A scalar residual score determines whether an example receives representation penalties; sensitivities of that same score determine the relative weights of the text, image, and fusion penalties. This links both decisions to the current model rather than fixing their allocation before training. Answer-level suppression continues throughout, and preservation losses apply independently of the residual score.

Residual score and activation. Stage II starts from the Stage-I checkpoint selected on separate forget and preservation validation examples (Appendix D.3). For $\mathcal{F}_A = \{x \in \mathcal{F} : \mathcal{A}(x) \neq \varnothing\}$, a temperature-controlled log-mean-exp summarizes the current alias scores:

$$s_{\text{sem}}(x) = \tau_p \log \left[\frac{1}{|\mathcal{A}(x)|} \sum_{a \in \mathcal{A}(x)} \exp\left(\frac{s_\theta(a \mid x)}{\tau_p} \right) \right]. \tag{4}$$

Higher-scoring aliases contribute more; smaller $\tau_p > 0$ increases this emphasis. This score provides a training proxy for recovery over the finite alias set; output-based recovery is evaluated separately. We activate representation penalties when $h(x) = \mathbf{1}[s_{\text{sem}}(x) > \delta]$, with $\delta = \log(0.02)$. Alias suppression remains active at scale $\gamma(x) = 0.2 + 0.8h(x)$, including one-fifth strength below the threshold. The gate and scale are held fixed during each update.

Adaptive route weights. For active examples, we compute sensitivities under the full question–image input. E_Q denotes question embeddings and $Z(I)$ denotes projected image tokens. The highest-scoring alias, $a^\dagger = \arg\max_{a \in \mathcal{A}(x)} s_\theta(a \mid x)$, fixes the answer positions used for inspection. At selected decoder layers $\mathcal{L}_F$, B_ℓ contains answer-to-image attention weights. Using entrywise ℓ_1 norms and elementwise multiplication $\odot$, we define

$$g_T = \left\| \frac{\partial s_{\text{sem}}}{\partial E_Q} \right\|_1, \qquad g_I = \left\| \frac{\partial s_{\text{sem}}}{\partial Z(I)} \right\|_1, \qquad g_F = \sum_{\ell \in \mathcal{L}_F} \left\| B_\ell \odot \frac{\partial s_{\text{sem}}}{\partial B_\ell} \right\|_1. \tag{5}$$

The derivatives are taken through the aggregate score, with the discrete alias selection fixed. A softmax with temperature $\tau_a > 0$ converts the sensitivities into weights:

$$\widetilde{w}_r = \frac{\exp(g_r/\tau_a)}{\sum_{j \in \{T,I,F\}} \exp(g_j/\tau_a)}, \qquad r \in \{T, I, F\}. \tag{6}$$

For each example, the first active visit initializes $w_r = \widetilde{w}_r$; subsequent active visits use $w_r \leftarrow 0.7w_r + 0.3\widetilde{w}_r$. Smoothing limits changes between successive active visits. The weights are detached from the parameter update, so optimization acts on the weighted penalties without differentiating through the allocation rule. All parameters remain trainable. Because gradient magnitudes depend on the chosen representations, the weights express local sensitivity rather than a causal decomposition of stored knowledge. Section 4.4 tests their utility under fixed data and training budgets.

Route-specific representation penalties. For corresponding current and frozen feature summaries v^θ, v^{θ_0}, define the cosine-margin penalty

$$d_m(v) = \max\{0, \cos(v^\theta, v^{\theta_0}) - m\}. \tag{7}$$

Minimization reduces similarity only while it exceeds margin m. Comparisons use identical inputs and answer prefixes in both models. The three penalties act on distinct feature summaries.

Text. With the image removed, $\bar{H}_{\ell,Q}$ averages question-token hidden states at selected layers $\mathcal{L}_T$:

$$R_T = \frac{1}{|\mathcal{L}_T|} \sum_{\ell \in \mathcal{L}_T} d_{m_T}(\bar{H}_{\ell,Q}). \tag{8a}$$

This penalty acts on image-free question representations, although its weight is computed from the multimodal input.

Image. $\bar{V}$ averages patches within each of the top two vision blocks and then averages the two vectors; $\bar{Z}$ averages projected image tokens. We penalize both encoder and projected features:

$$R_I = d_{m_V}(\bar{V}) + \eta_Z d_{m_Z}(\bar{Z}). \tag{8b}$$

Fusion. Under the full input and selected alias $a^\dagger$, $\bar{H}_{\ell,A}$ averages answer-token hidden states. $\bar{B}_\ell$ averages answer-to-image attention over heads and answer positions, retaining one entry per image token:

$$R_F = \frac{1}{|\mathcal{L}_F|} \sum_{\ell \in \mathcal{L}_F} \left[d_{m_H}(\bar{H}_{\ell,A}) + \eta_A d_{m_A}(\bar{B}_\ell) \right]. \tag{8c}$$

The margins m_T, m_V, m_Z, m_H, m_A set where each penalty becomes inactive; $\eta_Z, \eta_A > 0$ set relative weights. These penalties move target-related features away from the frozen model; answer-level suppression links that change to the unlearning objective.

Combined objective. We suppress every verified alias using $\ell_{\text{sem}}(x) = |\mathcal{A}(x)|^{-1} \sum_{a \in \mathcal{A}(x)} u_\theta(a, x)$. The per-example forget loss is

$$\ell_{\text{F}}^{(2)}(x) = \lambda_5 \gamma(x) \ell_{\text{sem}}(x) + \lambda_6 h(x) \sum_{r \in \{T,I,F\}} w_r(x) R_r(x). \tag{9}$$

When $h = 0$, we skip route calculations, leave the stored weights unchanged, and apply only the reduced alias loss. Stage II retains the preservation objectives and adds next-token distillation on $\mathcal{R}$, denoted $L_{\text{R-KD}}$:

$$L^{(2)} = \mathbb{E}_{x \sim \mathcal{F}_A} \ell_{\text{F}}^{(2)}(x) + \lambda_7 \big(L_{\text{R-wCE}} + \beta L_{\text{R-KD}} \big) + \lambda_8 L_{\text{util}} + \lambda_9 L_{\text{decouple}}. \tag{10}$$

All coefficients are fixed and nonnegative. Preservation remains active regardless of the gate. Appendix C.2 defines feature pooling; Appendices C.3–C.4 specify gating, weight smoothing, and update order; Appendix D lists hyperparameters.

4 Experiments

Our evaluation examines three aspects of RAMP: the forgetting–preservation trade-off, the contribution of residual-guided updates, and recovery beyond the training answers. We first report benchmark performance, then compare training stages, route weights, and auxiliary supervision under explicit controls. Unseen attacks, output audits, utility-data substitution, and cost measurements assess the scope and practical requirements of the resulting behavior.

4.1 Benchmarks and evaluation

Model and benchmarks. We update all parameters of LLaVA-1.5-7B (Liu et al., 2024). CLEAR's 10% forget setting (Dontsov et al., 2025) reports target recovery (*Forget*), retained knowledge (*Retain*), unrelated-task performance (*Real*), and Forget Quality (FQ). UMU-Bench (Wang et al., 2025) covers classification, cloze, and generation at 5%, 10%, and 15% forget rates. Its Unlearning Completeness (UC) measures remaining forget-task performance, so lower is better; Model Utility (UT) is higher-is-better. Tables 1 and 2 use the released evaluators and published baseline values.

Table 1: **CLEAR, 10% forget.** Selected published baselines; arrows show preferred directions. Gold is CLEAR's counterfactual reference, and Base is the pre-unlearning model. **Bold** marks column-best unlearning results, including ties; reference rows are excluded.

Method	Real ↑	Retain ↑	Forget ↓	FQ ↑
Gold	0.50	0.51	0.19	1.00
Base	0.48	0.51	0.35	0.85
DPO	0.46	0.48	0.22	0.84
GA	0.27	0.00	**0.00**	0.67
IDK	0.48	0.51	0.33	0.84
LLMU	0.47	0.51	0.25	0.84
NPO	0.46	0.14	0.11	0.76
Retain FT	0.49	0.51	0.37	0.85
RMU	0.24	0.00	**0.00**	0.75
SCRUB	0.49	**0.52**	0.36	0.85
RAMP	**0.50**	**0.52**	0.08	**0.89**

Held-out recovery. Before Stage 0, we reserve 25% of each forgotten identity's text question–answer pairs and 25% of its images. *Exact* measures target-string recovery on direct questions; *Paraphrase* uses unseen rewordings; *Indirect* uses clues such as roles or dates. The latter two accept factual recovery with either original or alternative wording. Scores average instances within each identity, then average identities; they do not mark an identity recovered whenever any prompt succeeds. These output-based tests are distinct from the teacher-forced training score s_{sem} (Appendix E.4).

Comparisons and uncertainty. Additional comparisons rerun named methods with matched inputs, formatting, decoding, and scoring. CLEAR statistical reruns and routing controls use three seeds. Confidence intervals resample identity blocks; paired permutation tests use Holm correction within each comparison family (Appendix E.2). *Native utility* evaluates fixed examples from the original instruction-tuning distribution and is distinct from CLEAR Real and UMU UT. Appendices E.1 and D give protocol and training details.

4.2 Forgetting and preservation

Table 1 shows that RAMP combines low target recovery with strong retained performance: Forget is 0.08, Retain is 0.52, and Real is 0.50. The baselines illustrate why these metrics must be considered jointly. GA reduces both Forget and Retain to zero, while SCRUB maintains Retain at 0.52 but leaves Forget at 0.36. NPO reaches Forget 0.11, with substantially lower Retain of 0.14. RAMP's advantage is therefore the measured combination of forgetting and preservation, rather than the minimum Forget score in isolation.

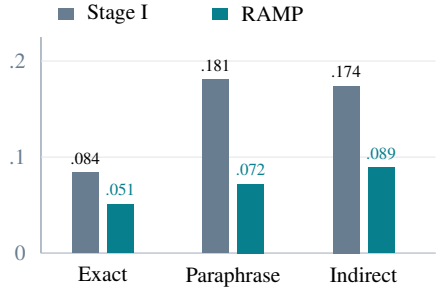
Table 2: **UMU-Bench.** Selected published baselines at 5%, 10%, and 15% forget rates; arrows show preferred directions. Origin is the pre-unlearning model. **Bold** marks column-best unlearning results, including ties; the reference row is excluded.

	5%		10%		15%	
Method	**UC ↓**	**UT ↑**	**UC ↓**	**UT ↑**	**UC ↓**	**UT ↑**
Origin	0.8873	0.5482	0.8632	0.5486	0.8501	0.5498
GA	0.7367	0.4558	0.6344	0.3944	0.6002	0.3880
GD	0.7300	0.4660	0.7113	0.4485	0.6110	0.3690
KL	0.7010	0.4358	0.6691	0.4099	0.5817	0.3949
NPO	0.7057	0.4196	0.6989	0.4250	0.6996	0.4497
PO	0.7149	0.3979	0.6222	0.3662	0.5832	0.3654
RAMP	**0.6089**	**0.4916**	**0.5474**	**0.4808**	**0.5056**	**0.4712**

Three-seed reruns give RAMP Forget 0.081 (95% CI: [0.045, 0.124]), compared with SCRUB's 0.359 (Holm-adjusted $p = 0.0006$). The Forget difference from NPO is not significant ($p = 0.444$), and neither are the Real, Retain, or FQ differences from SCRUB (Table 8). These tests support lower recovery relative to a baseline with similar retained performance, while leaving the smaller differences unresolved. We accordingly treat the benchmark table as a trade-off comparison rather than evidence of superiority on every metric.

On UMU-Bench (Table 2), RAMP has lower UC and higher UT than the reported unlearning baselines at all three forget rates. The task breakdown also exposes the limits of these averages: at 10% forgetting, RAMP's multimodal classification utility is 0.5269 versus NPO's 0.4357, whereas generation utility is lower, 0.1052 versus 0.1406 (Table 11). Thus the aggregate trade-off does not imply better preservation on every task.

(a) Recovery after direct suppression



(b) Recovery across input conditions

	Question only	Image + neutral prompt	Question + image
Equal weights	.050	.089	.112
Shuffled weights	.063	.111	.139
Text penalty only	.035	.145	.183
Image penalty only	.073	.056	.158
Fusion penalty only	.069	.123	.061
RAMP	.041	.068	.081

Figure 2: **Residual recovery and cross-modal leakage on CLEAR.** (a) Stage II reduces recovery on held-out direct, paraphrased, and indirect questions. (b) All variants share the Stage-I checkpoint, data, other losses, and 1200 Stage-II updates. Darker cells indicate more recovery. Single-route penalties leave substantial leakage through other input conditions; adaptive weighting gives a more balanced reduction. Input conditions are evaluation probes, not isolated internal routes. Values are point estimates.

4.3 Residual recovery after direct-answer suppression

Stage I reduces exact recovery to 0.084, yet paraphrase and indirect recovery remain 0.181 and 0.174. Stage II lowers these three scores to 0.051, 0.072, and 0.089 (Figure 2(a)). The reductions are larger for paraphrased and indirect questions than for exact recovery: 0.109 and 0.085, respectively, versus 0.033. This pattern is consistent with Stage II addressing expressions of the target that remain accessible after its original wording has been suppressed. RAMP also improves on NPO's paraphrase and indirect scores of 0.196 and 0.183 (Holm-adjusted $p = 0.0056$ and 0.0104).

Table 3: **Training recipes on CLEAR.** Recipes share prepared data. Utility uses the fixed native set. **Bold** marks column-best values.

Recipe	Exact ↓	Retain ↑	Utility ↑
Stage I only	0.084	0.515	0.487
Stage II only	0.173	0.498	0.474
Merged	0.093	0.487	0.463
RAMP	**0.051**	**0.521**	**0.501**

Table 3 compares complete training recipes using the same prepared data. Stage II alone leaves exact recovery at 0.173. Merging the objectives into one stage gives exact recovery 0.093 and utility 0.463, compared with 0.051 and 0.501 for RAMP. The sequential recipe provides both lower direct recovery and higher preservation among these alternatives. These comparisons assess complete training recipes; the next experiment isolates the route-weighting rule by fixing the starting checkpoint and the number of subsequent updates.

4.4 Adaptive routing and auxiliary supervision

Adaptive route weights. Routing variants share the Stage-0 data, selected Stage-I checkpoint, optimizer, other losses, and 1200 Stage-II updates. Equal weights set $w_T = w_I = w_F = 1/3$. Shuffled weights reassign adaptive vectors across identities, preserving their distribution while breaking the association with each target. Paraphrase recovery is 0.072 with adaptive weights, 0.094 with equal weights, and 0.113 with shuffled weights; indirect recovery is 0.089, 0.112, and 0.134. Native utility is 0.501, 0.488, and 0.477, respectively. Equal weighting controls for the presence of all three penalties, while shuffling controls for the marginal distribution of adaptive weights. Their higher recovery indicates that associating the weights with the corresponding target contributes to the observed improvement.

Consistency across input conditions. Figure 2(b) evaluates the same trained variants with a question alone, an image with a neutral prompt, and the original question–image pair. Each single-route penalty performs best under one input condition but leaves higher recovery elsewhere. The text-only penalty, for example, gives 0.035 without an image and 0.183 with the full input. The largest recovery across the three conditions is 0.183, 0.158, and 0.123 for the text-, image-, and fusion-only penalties, versus 0.081 for RAMP. The comparison reveals a trade-off hidden by any single access test: a penalty can suppress recovery in its strongest condition while leaving the target accessible through another. RAMP reduces the largest observed recovery across these conditions without requiring a different model for each input type. The UMU-Bench routing comparison gives a complementary task-level check. At 10% forgetting, adaptive rather than equal weights reduce UC from 0.5712 to 0.5474 and raise UT from 0.4690 to 0.4808. The absolute unimodal–multimodal target-score gaps fall from 0.069 to 0.054 for classification, 0.058 to 0.044 for cloze, and 0.095 to 0.079 for generation (Table 12). Smaller gaps alone could reflect equally high recovery in both conditions; the accompanying reduction in UC shows that agreement here improves alongside forgetting.

Table 4: **Shared supervision on CLEAR.** P adds preservation supervision; T adds target-span and alias losses. Baselines retain their original objectives, match total example presentations, and are not separately retuned. Utility uses the fixed native set. **Bold** marks column-best values.

Method	Para. ↓	Indirect ↓	Utility ↑
NPO + P	0.232	0.219	0.486
NPO + P + T	0.126	0.154	0.478
SCRUB + P	0.343	0.299	0.500
SCRUB + P + T	0.185	0.219	0.493
RAMP	**0.072**	**0.089**	**0.501**

Shared-supervision baselines. We give one-stage NPO and SCRUB benign-question supervision and utility distillation, then add target-span and alias suppression. These controls retain their original objectives, match RAMP's total example presentations, and are not separately retuned (Appendix E.6). Adding targets reduces paraphrase recovery from 0.232 to 0.126 for augmented NPO and from 0.343 to 0.185 for augmented SCRUB (Table 4). These targets improve both baselines, but do not close the gap to RAMP's 0.072 recovery.

Table 5: **Recovery under unseen attacks (↓).** Each family uses three prompts per identity across 20 identities and reserved images. The no-route control retains alias suppression and preservation. Scores average within, then across, identities. **Bold** marks the row minimum.

	Stage and route controls			Baselines	
Attack family	**Stage I**	**No route loss**	**RAMP**	**NPO**	**SCRUB**
Long-tail paraphrase	0.260	0.137	**0.091**	0.241	0.360
Visual clues	0.268	0.160	**0.098**	0.255	0.397
Two-hop relations	0.286	0.183	**0.111**	0.287	0.412
Candidate disambiguation	0.224	0.131	**0.084**	0.221	0.349
Two-turn dialogue	0.310	0.199	**0.124**	0.303	0.438

The same distinction appears under changed inputs. With all shared supervision, NPO and SCRUB still have question–image recovery of 0.226 and 0.309, compared with RAMP's 0.081. On UMU-Bench at 10% forgetting, augmented NPO reaches UC 0.6315 and UT 0.4510, versus 0.5474 and 0.4808 for RAMP (Table 18). The residual gap therefore extends beyond the CLEAR paraphrase test. These controls assess the complete training procedure; the fixed-budget comparisons above separately identify the contribution of route weighting.

Target construction. The construction ablation compares full-answer targets, a naive paraphrase bank, and the verified span-and-alias recipe. Exact recovery is 0.074, 0.056, and 0.051, respectively, while native utility is 0.452, 0.484, and 0.501. Benign-neighbor refusal also falls from 0.126 with full-answer targets to 0.057 with the naive bank and 0.041 with the verified recipe (Table 21). The comparison links target construction to collateral behavior as well as recovery: suppressing the entire answer affects harmless questions more often. These are complete recipes, so the result does not isolate verification from bank composition or training exposure.

Common training filter. A separate full-pool run starts with 779 examples: 11 are excluded from forget training and 16 receive Stage I only, but all remain in diagnostic evaluation. To test whether this filtering explains the gains, NPO and SCRUB exclude the same 11 training examples and are evaluated on the unchanged pool. RAMP's paraphrase recovery is 0.074 ± 0.041, compared with 0.206 ± 0.075 for NPO and 0.341 ± 0.087 for SCRUB; the reported uncertainties are 95% interval half-widths over three seeds. Indirect recovery is 0.090, 0.192, and 0.299, respectively (Table 23). The recovery advantage persists when training exclusions are shared and construction failures remain in the test. These full-pool diagnostics are separate from the held-out-input evaluation.

4.5 Unseen attacks and output behavior

Post-training attacks. A separate post-training suite tests 20 identities with three prompts from each of five families: long-tail paraphrases, visual clues, two-hop relations, candidate disambiguation, and two-turn dialogue. Prompts exclude exact gold-answer, target-span, and training-alias strings; image-based attacks use reserved images. None are used for training or checkpoint selection. These attacks test recovery beyond the finite training aliases.

Table 5 compares Stage I, full RAMP, and a no-route control that starts from the same Stage-I checkpoint and retains alias suppression. RAMP has lower recovery in every family than the no-route control, despite both variants receiving alias suppression. For two-hop questions, recovery is 0.111 with RAMP and 0.183 without the route loss; for two-turn dialogue, the corresponding values are 0.124 and 0.199. Two-turn dialogue remains the strongest tested family against RAMP, identifying a concrete source of residual leakage.

Output quality and selectivity. We partition CLEAR benchmark forget-query outputs into six mutually exclusive categories (Table 20). From Stage I to RAMP, semantic-only disclosure falls from 15.6% to 2.9%, while coherent non-target responses rise from 55.2% to 78.9%. Refusals decrease from 7.1% to 4.1%, and degenerate outputs from 0.5% to 0.3%. Thus the reduction in disclosure does not come from replacing answers with more refusals or generation failures. Semantic-only disclosure excludes exact-string hits; it is distinct from the Paraphrase and Indirect tests, which accept any wording that reveals the fact.

The benign-neighbor evaluation tests whether this change preserves harmless answers about the same images. From Stage I to RAMP, accuracy increases from 88.6% to 90.3% and target leakage falls

Table 6: **Utility-data substitution.** Only the utility-training set $\mathcal{U}$ changes. Native and public utility use separate, fixed evaluation sets. **Bold** marks column-best values.

	Held-out recovery ↓			Utility ↑		Benign
Training utility data	**Exact**	**Para.**	**Indirect**	**Native**	**Public**	**acc. ↑**
Original mixture	0.051	**0.072**	**0.089**	**0.501**	0.498	**0.903**
Filtered public data	0.055	0.080	0.098	0.493	**0.505**	0.892
Retain-set examples	0.052	0.093	0.113	0.452	0.466	0.876
No utility data	**0.046**	0.081	0.102	0.421	0.438	0.805

from 0.027 to 0.011. NPO gives 81.6% accuracy and 13.6% refusal on these questions, compared with RAMP's 4.1% refusal. A stratified 200-output audit gives Cohen's $\kappa = 0.884$; conservative relabeling changes each audited metric by at most 0.009. These results support selective suppression under the evaluated questions. They do not establish equivalence to Gold: 8.6% of RAMP's forget-query outputs contain wrong non-target facts, and a coherent non-target answer need not correctly answer the original identity question.

4.6 Data requirements, sensitivity, and cost

Utility-data source. We replace only $\mathcal{U}$ with 50,000 filtered public examples spanning multimodal instruction following, VQA, captioning, OCR, and grounded reasoning. All other data, losses, training steps, and checkpoint selection remain fixed. Native utility falls from 0.501 to 0.493 and indirect recovery rises from 0.089 to 0.098, while public utility improves from 0.498 to 0.505 (Table 6). Using only retain-set examples gives substantially lower native utility, 0.452. Removing utility supervision reduces exact recovery to 0.046 but lowers native utility to 0.421 and benign accuracy to 0.805. Retain and benign-neighbor losses alone do not recover the utility preserved by broader distillation. Filtered public data provide a practical alternative to the original mixture.

Sensitivity to loss weights. With other settings fixed, increasing the route-loss coefficient λ_6 from 0.4 to 1.2 reduces question–image recovery from 0.101 to 0.074, while native utility falls from 0.504 to 0.494. Stronger preservation weighting moves the trade-off in the opposite direction. These local sweeps characterize sensitivity around the default; several preservation and temperature settings use one seed and do not establish broad robustness (Appendix E.11).

Computational cost. On eight A100-80GB GPUs, Stage I takes 13.8 hours and both training stages take 22.6 hours, a 63.8% increase; offline preparation adds 1.845 hours. Attention extraction takes 0.70 hours within Stage II, and peak memory is 67.4 GB per GPU (Appendix E.12). The gate activates on 4,746 of 11,520 Stage-II forget visits (41.2%); attention extraction accounts for 7.9% of Stage-II time. Across UMU-Bench forget rates, active shares range from 37.8% to 42.69%, and attention extraction from 7.1% to 8.4% of Stage-II time. Residual gating therefore confines route computation to a subset of visits, although the second training stage still incurs substantial cost. These measurements characterize the evaluated settings, not scalability to substantially larger forget sets or other architectures.

5 Conclusion

RAMP uses residual answer scores to guide representation penalties after direct-answer suppression, allowing the model to focus additional unlearning updates on examples that remain recoverable after the first stage. It separates the decision of whether further intervention is needed from how that intervention is allocated across text, image, and fusion routes, making the update both target-dependent and modality-aware. On CLEAR and UMU-Bench, RAMP achieves a favorable forgetting–preservation trade-off, reducing target recovery while maintaining retained and unrelated-task performance. Controlled routing experiments and shared-supervision baselines further distinguish the contribution of adaptive residual-guided weighting from the effects of additional target and preservation data. Unseen attacks, held-out recovery tests, and public-data substitution additionally probe whether the observed behavior extends beyond training aliases and the original utility mixture.

These findings remain limited to the evaluated preservation distributions and attack families. They do not establish complete information erasure, robustness to all extraction strategies, or equivalence to retraining without the target data. Broader validation and lower preprocessing and training costs remain important future directions.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic data sets, design or provide feedback on research methodology or experiments, implement methods, assist with translation, and clean and reformat dataset, edit a research paper to improve readability. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, support qualitative and thematic data analysis, or interpret results.

Additionally, we used generative AI tools to create or modify scientific figures or images and create or edit software code. We have reviewed all AI-assisted work. All LLM-generated code was thoroughly tested and verified by three authors. Additionally, all formatted tables within the manuscript were manually cross-validated against the raw data by two authors to guarantee accuracy. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Trishna Chakraborty, Erfan Shayegani, Zikui Cai, Nael B. Abu-Ghazaleh, M. Salman Asif, Yue Dong, Amit Roy-Chowdhury, and Chengyu Song. Can textual unlearning solve cross-modality safety alignment? In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9830–9844, 2024.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Journey to the center of the knowledge neurons: Discoveries of language-independent knowledge neurons and degenerate knowledge neurons. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17817–17825, 2024. doi: 10.1609/aaai.v38i16.29735. URL https://ojs.aaai.org/index.php/AAAI/article/view/29735.

Alexey Dontsov, Dmitrii Korzh, Alexey Zhavoronkin, Boris Mikheev, Denis Bobkov, Aibek Alanov, Oleg Rogov, Ivan Oseledets, and Elena Tutubalina. CLEAR: Character unlearning in textual and visual modalities. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20582–20603, 2025.

Ronen Eldan and Mark Russinovich. Who's harry potter? approximate unlearning in LLMs, 2023.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://arxiv.org/abs/2209.10652.

Jiahui Geng and Qing Li. SAUCE: Selective concept unlearning in vision-language models with sparse autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3023–3033, October 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Geng_SAUCE_Selective_Concept_Unlearning_in_Vision-Language_Models_with_Sparse_Autoencoders_ICCV_2025_paper.html.

Jiahao Huo, Yibo Yan, Xu Zheng, Yuanhuiyi Lyu, Xin Zou, Zhihua Wei, and Xuming Hu. MMUnlearner: Reformulating multimodal machine unlearning in the era of multimodal large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7190–7206, 2025.

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, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.805. URL https://aclanthology.org/2023.acl-long.805/.

Jiaqi Li, Qianshan Wei, Chuanyi Zhang, Guilin Qi, Miaozeng Du, Yongrui Chen, Sheng Bi, and Fan Liu. Single image unlearning: Efficient machine unlearning in multimodal large language models. In *Advances in Neural Information Processing Systems*, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Zheyuan Liu, Guangyao Dou, Xiangchi Yuan, Chunhui Zhang, Zhaoxuan Tan, and Meng Jiang. Modality-aware neuron pruning for unlearning in multimodal large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5913–5933, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.295. URL https://aclanthology.org/2025.acl-long.295/.

Yingzi Ma, Jiongxiao Wang, Fei Wang, Siyuan Ma, Jiazhao Li, Jinsheng Pan, Xiujun Li, Furong Huang, Lichao Sun, Bo Li, Yejin Choi, Muhao Chen, and Chaowei Xiao. Benchmarking vision language model unlearning via fictitious facial identity dataset, 2025. URL https://openreview.net/forum?id=0y3hGn1wOk. OpenReview.

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

Chengye Wang, Yuyuan Li, XiaoHua Feng, Chaochao Chen, Xiaolin Zheng, and Jianwei Yin. Umubench: Closing the modality gap in multimodal unlearning evaluation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning, 2024.

A Constructing the forget and preservation data

Stage 0 is run before either training stage. All language-model calls use `gpt-4o-2024-08-06`, with JSON-schema checks and fixed acceptance rules. Candidate generation and verification are separate calls; "independently verified" does not mean that a different model family provides the verification. The resulting targets are fixed during unlearning. This separates deciding which facts to remove from updating the VLM: the target definition does not change as its answers become less likely. Questions and images reserved for held-out evaluation are excluded before construction. Separately generated attack prompts are never used as training targets, including the attacks generated after unlearning in the supplementary evaluation.

A.1 Answer spans, alternative expressions, and harmless neighbors

Answer spans. For $x = (q, I, y^\star)$, the extraction prompt proposes at most eight contiguous spans copied from $y^\star$. A separate verification call keeps a span only when removing or replacing it makes the answer incorrect, too vague, or identity-ambiguous. These accepted spans form $\mathcal{K}(x)$. We normalize whitespace and remove duplicate spans. For nested spans, we keep the shorter one when both identify the same fact, but keep the longer one when the extra words distinguish the identity or object, as in "iPhone 13 Pro" versus "iPhone." Spans shorter than two non-punctuation characters are dropped unless they are numbers, dates, or atomic entities.

Alternative expressions. The generator receives $(q, I, y^\star, \mathcal{K}(x))$ and proposes six to ten ways to express the same fact. The verification call checks semantic equivalence and rejects candidates that add or change facts. We also reject exact copies of the gold answer and trivial substring variants. At least half of the generated candidates are requested to avoid exact reuse of the indispensable spans. After filtering, we keep at most eight aliases; when more survive, selection favors diversity under edit distance and sentence-embedding similarity. The resulting $\mathcal{A}(x)$ contains alternative *answers*, not an exhaustive bank of attack questions. These answers serve two purposes: their scores reveal which expressions remain easy for the current model, and their unlikelihood losses teach it to avoid those expressions.

Harmless same-image questions. The neighbor generator proposes three question–answer pairs about the image that do not ask for, or reveal, the target. A verifier checks that each question is answerable from that image and that its answer does not reveal the target through a span, alias, or identifying relation. The frozen original model must also assign the answer a mean token log-probability above the 20th percentile of verified neighbor candidates in the same batch. At most

three accepted pairs form $\mathcal{D}(x)$, and $\mathcal{D} = \bigcup_{x \in \mathcal{F}} \mathcal{D}(x)$. For identity forgetting, clothing color is an example of an attribute that can remain answerable without naming the person. Keeping the image but changing the question gives direct supervision for the distinction we want: suppress the identity answer without making every question about that image unanswerable.

Retain-critical spans. Retain answers receive the same extraction and necessity-verification procedure. Accepted spans identify the factual content that should remain correct, rather than content to suppress. Their tokens receive the larger cross-entropy weight in Appendix B. We remove conflicting retain, neighbor, and utility examples using the same direct-overlap and semantic-overlap tests, so a preservation objective does not train the model to reproduce a forget target.

A.2 Building the utility set

Native-data construction. Our default utility pool is the original instruction-tuning mixture. A utility example must not teach back a forget target. We reject it if it directly contains an indispensable span or verified alias, or if it overlaps with the retain set. We also reject it when sentence-embedding cosine similarity to a forget question or alias exceeds 0.75, or when image-caption or instruction similarity to a target description exceeds 0.75.

Embedding models. We measure sentence-level overlap with `sentence-transformers/all-mpnet-base-v2`. For image-description and target-description overlap, we use the text tower of `openai/clip-vit-large-patch14-336`. Both encoders are frozen and kept in evaluation mode. For MPNet, we average token states using the attention mask, so padding does not enter the mean. For CLIP, we use the checkpoint's native tokenizer and maximum text length. We L2-normalize every resulting vector before computing cosine similarity. The CLIP comparison is between the textual descriptions, not between a text embedding and an image embedding. Both checkpoints and the threshold 0.75 are fixed across datasets, seeds, targets, and ablations. A semantic-overlap test rejects a candidate only when its similarity exceeds this threshold.

Coverage ranking and sample quotas. After filtering, we allocate a quota to each source in the original mixture's proportions. Within each source, we stratify by prompt length, answer length, task/modality tag, and image format or count when available, and allocate the source quota across these strata. To cover different examples within a stratum, we use the original model's visual and question features. Let L be the number of decoder layers, let norm denote unit-vector normalization, and let $\|$ denote vector concatenation. With the pooled features defined in Appendix C.2, the coverage vector is

$$e(x) = \left[\mathrm{norm}(\bar{Z}^{\theta_0}(I)) \parallel \mathrm{norm}(\bar{H}^{\theta_0}_{L-1,Q}(q, I))\right]. \tag{11}$$

The first half of $e(x)$ describes the image through the original model; the second describes the question in its multimodal context. Normalizing the two halves prevents their raw vector lengths from deciding which half dominates the distance. Within each source and stratum, k-means groups similar candidates, and k-center greedy selection ranks examples to cover different parts of those groups. This is a data-selection step, not an additional loss on the model. We retain this ranking for selection and quota refill. The prescribed utility-set size is

$$|\mathcal{U}| = \min\{120{,}000, \max(50{,}000, 8|\mathcal{F}|)\}. \tag{12}$$

The floor supplies at least 50,000 preservation examples, the middle range grows with the forget set, and the cap limits the pool to 120,000. Thus larger forget requests receive broader preservation coverage without letting the utility pool grow without bound.

Confidence trimming and quota refill. Within each stratum, we mark examples below the 10th or above the 90th percentile of teacher sequence log-likelihood as ineligible. We then scan the same coverage ranking from the top, skip ineligible examples, and continue until the stratum quota is filled. If a stratum has too few eligible candidates, its deficit is redistributed across other strata of the same source in proportion to their remaining eligible capacity. These strata continue down their existing rankings; we do not introduce another sampling rule or transfer quota between sources. Thus refill changes the allocation within a source, not the source-level mixture proportions. The completed utility set is kept fixed across stages, seeds, and ablations unless the utility source itself is under study. Each selected example has a stable target response, so the current and frozen models can be compared after the same answer prefixes.

When the original mixture is unavailable. The public-utility variant replaces only $\mathcal{U}$; $\mathcal{K}$, $\mathcal{A}$, $\mathcal{D}$, loss weights, overlap thresholds, training steps, and selection rules remain unchanged. It mixes public

instruction, visual question answering, captioning, OCR, and grounding/reasoning examples. Stable provided answers supply the response prefixes; otherwise, the original model supplies pseudo-targets or cached teacher outputs. The completed public-data construction uses the following sources and proportions.

Source/task family	Candidates	After filtering	Selected
LLaVA-Instruct / general instruction	52,500	51,486	17,500
VQAv2 / visual QA	45,000	44,394	15,000
COCO caption / image description	22,500	22,046	7,500
OCR-VQA + TextVQA	15,000	14,778	5,000
GQA + grounding/reasoning	15,000	14,757	5,000
Total	150,000	147,461	50,000

The selected proportions are $35\%, 30\%, 15\%, 10\%, 10\%$. Filtering removes 137 candidates by direct span/alias match, 1,421 by text-semantic overlap, and 981 by image-description overlap; no candidate is removed for retain-set overlap in this construction. Public-data replacement changes the source of preservation examples, not the forget objective or the attribution rule.

A.3 Failed construction and low-content answers

For a yes/no answer, we seek the minimal supporting proposition rather than treating "yes" or "no" alone as the fact to forget. Without a stable supporting proposition, the example is excluded from forget training but remains in evaluation. Numbers, dates, and atomic entities can themselves serve as indispensable units. If alias verification fails, we try at most three templated descriptive rewrites; an example with no valid alias receives Stage-I forgetting but no Stage-II alias-based update. If two extraction attempts and the deterministic fallback still yield no valid span, the example is excluded from forget training and retained in diagnostic evaluation. Thus losses requiring $\mathcal{K}(x)$ or $\mathcal{A}(x)$ are evaluated only on examples for which the required set is nonempty; exclusion from training does not filter the evaluation set.

A call that fails JSON validation twice falls back to deterministic processing. For spans, we extract noun phrases, named entities, model numbers, dates, and quantities and apply the same nesting rules. For aliases, we use a curated synonym/hypernym dictionary. For neighbors, fixed templates ask about visible color, count, location, OCR, or action, and candidates still must pass teacher verification.

A.4 Annotation prompts

The six prompts below turn the construction rules into explicit calls: propose spans, verify their necessity, propose equivalent answers, verify equivalence, then propose and check harmless same-image questions. A generated candidate is therefore not accepted merely because the generator produced it. Placeholders are filled with the sample's question, image, answer, and previously verified targets.

Candidate extraction.

```
System. You are a precise multi-modal annotation engine. Return valid
↪  JSON only. No explanations.

User. You are given (1) an image, (2) a question, and (3) a gold answer.

Task. Extract only the minimal contiguous spans from the gold answer that
↪  are indispensable to the core factual proposition of the answer for
↪  this image and question.

Rules.
- Return at most 8 spans.
- Each span must be copied verbatim from the gold answer.
- Keep entities, model names, relations, quantities, dates, and
↪  indispensable attributes only if removing them would make the answer
↪  wrong, incomplete, or identity-ambiguous.
- Do not keep style words, discourse words, or removable modifiers.
- Prefer shorter spans when two spans are equivalent.
- If the answer is short, return only the truly indispensable part.
```

```
Output schema.

{
  "core_units": [
    {
      "span": "...",
      "type": "entity|relation|number|date|attribute|other"
    }
  ]
}

Inputs.
Question: <QUESTION>
Gold answer: <GOLD_ANSWER>
Image: <IMAGE>
```

Necessity verification.

```
System. You are a strict verifier. Return valid JSON only.

User. You are given:
(1) an image,
(2) a question,
(3) a gold answer,
(4) one candidate span copied from the gold answer.

Task. Decide whether this candidate span is indispensable to the core
↪  factual proposition of the gold answer for this image and question.

Definition of indispensable. Removing or replacing this span would make
↪  the answer factually incorrect, too underspecified, or
↪  identity-ambiguous.

Rules.
- Be conservative.
- KEEP only if the span is clearly necessary.
- DROP anything redundant, stylistic, or replaceable without changing the
↪  core fact.

JSON schema.

{
  "decision": "KEEP|DROP",
  "reason_type":
  ↪  "identity|relation|quantity|time|attribute|redundant|style|other"
}

Inputs. Question: <QUESTION>
Gold answer: <GOLD_ANSWER>
Candidate span: <CANDIDATE_SPAN>
Image: <IMAGE>
```

Alias generation.

```
System. You are a careful multimodal paraphrase generator. Return valid
↪  JSON only.

User. You are given:
(1) an image,
(2) a question,
(3) a gold answer,
(4) the indispensable knowledge units.

Task. Generate 6 to 10 semantically equivalent alternate expressions that
↪  preserve the same target fact but avoid trivial lexical copying.
```

```
Rules.
- Each alias must preserve the same target fact as the gold answer.
- Vary surface form using aliases, descriptive rewrites, definitional
↪  rewrites, or relation-preserving paraphrases.
- Do NOT introduce new facts.
- Do NOT generate refusals, uncertainty, or generic descriptions.
- At least half of the aliases must avoid exact string reuse of the
↪  indispensable spans.

JSON schema.

{
  "aliases": [
    {"text": "..."}
  ]
}

Inputs. Question: <QUESTION>
Gold answer: <GOLD_ANSWER>
Indispensable units: <CORE_UNITS_JSON>
Image: <IMAGE>
```

Alias verification.

```
System. You are a semantic equivalence verifier. Return valid JSON only.

User. You are given:
(1) an image,
(2) a question,
(3) a gold answer,
(4) one candidate alias.

Task. Decide whether the alias expresses the same target fact as the gold
↪  answer for this image and question, without adding or altering facts.

JSON schema.

{
  "decision": "KEEP|DROP",
  "same_fact": true,
  "adds_new_fact": false
}

Inputs. Question: <QUESTION>
Gold answer: <GOLD_ANSWER>
Candidate alias: <ALIAS>
Image: <IMAGE>
```

Neighbor generation.

```
System. You are a careful multimodal dataset constructor. Return valid
↪  JSON only.

User. You are given:
(1) an image,
(2) a forget question,
(3) a gold answer,
(4) the indispensable units and verified aliases.

Task. Construct 3 benign neighbor questions about the same image that
↪  remain answerable but do NOT require the forgotten target.

Rules.
- Each question must be answerable from the same image.
- The question must not request the forgotten identity, its aliases, or
↪  any relation that directly reveals it.
```

```
- Prefer visible attributes, counting, color, location, OCR, harmless
↪  relations, or scene layout.
- Return one concise gold answer for each.
- Do NOT create neighbors whose answers contain any indispensable unit or
↪  alias.

JSON schema.

{
  "neighbors": [
    {"question": "...", "answer": "..."},
    {"question": "...", "answer": "..."},
    {"question": "...", "answer": "..."}
  ]
}

Inputs. Forget question: <QUESTION>
Gold answer: <GOLD_ANSWER>
Indispensable units: <CORE_UNITS_JSON>
Aliases: <ALIASES_JSON>
Image: <IMAGE>
```

Neighbor verification.

```
System. You are a strict multimodal verifier. Return valid JSON only.

User. You are given:
(1) an image,
(2) a candidate benign neighbor question,
(3) its candidate answer,
(4) the forget target units and aliases.

Task. Decide whether this neighbor is:
- answerable from the same image,
- harmless with respect to the forget target,
- non-revealing of the target.

JSON schema.

{
  "decision": "KEEP|DROP",
  "answerable": true,
  "harmless": true,
  "reveals_target": false
}

Inputs. Question: <CANDIDATE_QUESTION>
Answer: <CANDIDATE_ANSWER>
Forget units: <CORE_UNITS_JSON>
Aliases: <ALIASES_JSON>
Image: <IMAGE>
```

B SCORING PHRASES AND PRESERVING OTHER ANSWERS

B.1 THE CONTEXT USED TO SCORE A TARGET PHRASE

All scores use the underlying checkpoint's chat template, image-token formatting, and assistant prefix. For a span inside $y^\star$, $c_\phi(x)$ includes the answer tokens before that span, but not tokens at or after the span. For an alias scored as an alternative answer, it includes the formatted question–image input and assistant prefix. The preceding phrase tokens are supplied as fixed prefixes when scoring each next token. Only the target span contributes to $s_\theta(\phi \mid x)$; system tokens, the user question, image placeholders, padding, and any answer prefix are context, not targets of this score. This fixed-prefix evaluation is teacher forcing: it tests each target token after the preceding reference tokens instead of allowing an earlier generated mistake to change what is being scored. The model is not prompted with the full gold answer and then asked to repeat it.

A phrase may occupy several subword tokens. Its score averages their log-probabilities, so e^{s_θ} is a geometric mean rather than the joint sequence probability. Before evaluating the logarithm in u_θ, we clip this scalar to $[\epsilon, 1-\epsilon]$ with $\epsilon = 10^{-6}$. A high phrase score gives a larger unlikelihood penalty; minimizing that penalty discourages the selected wording. Clipping keeps the logarithm away from its singular endpoint rather than introducing a new forgetting target. The loss is evaluated over spans or aliases, then averaged over eligible forget examples. All teacher features and distributions come from the original frozen model in evaluation mode, with teacher gradients disabled.

B.2 Retain and benign-neighbor supervision

For a retained example $x = (q, I, y)$, y_t is its tth answer token and $y_{<t}$ its fixed answer prefix. Let $\omega_t = 1.5$ on verified retain-critical spans and $\omega_t = 1$ elsewhere. The retain loss is

$$L_{\text{R-wCE}} = -\mathbb{E}_{(q,I,y)\sim\mathcal{R}} \sum_{t=1}^{|y|} \omega_t \log p_\theta(y_t \mid q, I, y_{<t}). \tag{13}$$

A correct token becoming less likely increases this loss. The factor 1.5 makes that increase larger on verified factual tokens, so the retain objective protects the answer's factual content as well as its other words. For a harmless neighbor $(q', I, y') \in \mathcal{D}$, all answer tokens have unit weight:

$$L_{\text{decouple}} = -\mathbb{E}_{(q',I,y')\sim\mathcal{D}} \sum_{t=1}^{|y'|} \log p_\theta(y'_t \mid q', I, y'_{<t}). \tag{14}$$

This loss raises the probability of the harmless answer despite changes made for the forget target on the same image. The name L_{decouple} refers to separating the desired behaviors, such as answering a color question but not an identity question; the operation is ordinary answer cross-entropy, not another feature penalty.

B.3 Matching the original model's predictions

Correct-answer supervision protects the supplied answer tokens; matching the original model's predictions also protects its preferences among alternative next tokens. This prediction matching, called distillation, supplies the utility loss and the additional Stage-II retain loss. Both compare the current and frozen models at identical response prefixes, using the same frozen VLM already defined in the Method, not another learned model. For an example $x = (q, I, y)$, let Y_x contain its valid response-token positions. Prompt, image, and padding positions do not contribute to either loss. At each $t \in Y_x$, both models receive the same formatted input and the reference response tokens preceding t. Thus conditioning on x below denotes this fixed prefix, not access to the entire response.

Temperature and vocabulary. Let $\mathcal{V}$ be the full vocabulary, and let $z_{\vartheta,t}(v \mid x)$ be the unnormalized next-token score (logit) for token v at position t under model $\vartheta \in \{\theta_0, \theta\}$. With temperature $T = 2$, its full-vocabulary next-token distribution is

$$p^{(T)}_{\vartheta,t}(v \mid x) = \frac{\exp(z_{\vartheta,t}(v \mid x)/T)}{\sum_{v'\in\mathcal{V}} \exp(z_{\vartheta,t}(v' \mid x)/T)}, \qquad v \in \mathcal{V}. \tag{15}$$

The denominator makes these token probabilities sum to one. Using $T = 2$ softens both distributions, so matching them considers alternatives to the single highest-scoring token.

Utility: compare on the teacher's top-256 support. For each utility response position, we cache the frozen teacher's top $K = 256$ token logits. Let $\mathcal{V}_K(x, t)$ be this teacher-selected token set. Both teacher and student are renormalized on that same set:

$$p^{(K,T)}_{\vartheta,t}(v \mid x) = \frac{\exp(z_{\vartheta,t}(v \mid x)/T)}{\sum_{v'\in\mathcal{V}_K(x,t)} \exp(z_{\vartheta,t}(v' \mid x)/T)}, \qquad v \in \mathcal{V}_K(x, t). \tag{16}$$

Here *support* means the tokens included in the comparison. The student does not choose a separate top-K set: both models are compared on the teacher's chosen 256 alternatives. Renormalization compares their relative preferences within that set; it does not constrain probability mass outside it. We compute the KL at each valid response position, average over these positions, and then average over utility examples:

$$L_{\text{util}} = \mathbb{E}_{x\sim\mathcal{U}} \left[\frac{1}{|Y_x|} \sum_{t\in Y_x} T^2 \,\mathrm{KL}\Big(p^{(K,T)}_{\theta_0,t}(\cdot \mid x) \,\|\, p^{(K,T)}_{\theta,t}(\cdot \mid x)\Big) \right]. \tag{17}$$

The KL is zero when the two distributions agree and increases when their preferences differ. Minimizing it penalizes changes in next-token preferences on the utility examples. At $T = 2$, the factor $T^2 = 4$ rescales this averaged KL before the stage-level utility coefficient is applied. The outer average is over the utility examples in the current minibatch. This gives each example one contribution instead of weighting it by its number of response tokens. The same teacher-support approximation is used across utility-source comparisons and other ablations.

Retain: compare over the full vocabulary. For retain-side distillation, we use the full-vocabulary distributions from Eq. equation 15, not the top-256 approximation. The temperature, teacher-forced prefixes, and two levels of averaging remain the same:

$$L_{\text{R-KD}} = \mathbb{E}_{x\sim\mathcal{R}}\left[\frac{1}{|Y_x|}\sum_{t\in Y_x} T^2\,\mathrm{KL}\Big(p^{(T)}_{\theta_0,t}(\cdot \mid x)\;\|\;p^{(T)}_{\theta,t}(\cdot \mid x)\Big)\right]. \tag{18}$$

Unlike cross-entropy, this term checks the whole next-token distribution at each retained-answer position, not only the probability of its reference token. It therefore complements factual-token supervision while Stage II changes internal features. Stage II weights this retain-side term by $\beta = 0.5$ inside the retain objective in Eq. equation 10. The response-token means above apply to the distillation terms; the cross-entropy losses retain the token-sum definitions in Appendix B.2. Neither distillation term is scaled by the forget-example gate.

Mixing losses. Each minibatch is divided into forget, retain, neighbor, and utility subsets. Each preservation component is averaged over its own source examples before applying its loss weight. In Stage II, the gates are applied within each forget example's contribution before averaging over the eligible forget subset, as specified in Appendix C.4. This keeps a larger number of examples from one source from silently replacing the configured relative loss weights. It does not turn the cross-entropy token sums into token averages; their definitions remain as given above. The source proportions and coefficients are listed in Appendix D.

C Computing routing weights and comparing features

C.1 What is differentiated

The alias score answers whether another expression still needs suppression. Its derivatives answer a different question: which internal quantities change that score most readily under small local changes? The current model scores all verified aliases under teacher forcing. If the combined score is at or below δ, no route sensitivity is computed. For an active example, the scalar differentiated for attribution is always $s_{\text{sem}}(x)$ from Eq. equation 4. The highest-scoring alias $a^\dagger$ determines the answer positions at which fusion is examined and compared with the original model. Its discrete selection is not differentiated.

The smooth aggregation explains why an expression that remains easy to produce has more influence than one already suppressed. Writing α_a for the derivative of the combined score with respect to alias score $s_\theta(a \mid x)$ gives

$$\alpha_a = \frac{\exp(s_\theta(a \mid x)/\tau_p)}{\sum_{b\in\mathcal{A}(x)}\exp(s_\theta(b \mid x)/\tau_p)}. \tag{19}$$

These coefficients are positive and sum to one; the higher an alias score, the greater its coefficient. For any tensor z in the scoring computation, the chain rule gives

$$\frac{\partial s_{\text{sem}}}{\partial z} = \sum_{a\in\mathcal{A}(x)} \alpha_a \frac{\partial s_\theta(a \mid x)}{\partial z}. \tag{20}$$

The gradient is a weighted combination of the aliases' gradients. It therefore emphasizes the remaining easy expressions without replacing the bank by a single winning alias. A smaller τ_p concentrates these coefficients more strongly on high-scoring aliases. When z belongs only to one teacher-forced alias branch, only that branch contributes to its derivative. Each alias is scored under its own teacher-forced response prefix. The semantic forgetting loss is different from this probe: $\ell_{\text{sem}}(x)$ averages the unlikelihood penalties of *all* verified aliases. It is neither the highest alias score nor the smooth probe itself.

Attribution inputs. E_Q contains the question-token embeddings, excluding template and image tokens. $Z(I)$ contains projected image features supplied to the language decoder. All sensitivities are

evaluated under the full question–image input. The fusion input B_ℓ is the selected answer-to-image portion of the decoder attention tensor, retaining its head dimension when computing the entrywise gradient product in Eq. equation 5. The text and image sensitivities sum absolute gradient entries; fusion sums absolute attention-times-gradient entries across the selected layers. The absolute values measure sensitivity magnitude rather than allowing positive and negative entries to cancel. For fusion, multiplication by the existing attention weights emphasizes links that are both used and locally influential to the alias score. These are local sensitivity scores used to assign update coefficients, not a partition of the fact into independent knowledge stores. No dimension normalization or alternative saliency rule is added to those definitions.

C.2 Token positions, pooling, and layer selection

Routing weights decide how much pressure to apply; the following feature summaries define what each penalty compares. Let Q denote question-token positions, A the answer positions of $a^\dagger$, and M the projected image-patch positions after image-placeholder expansion. M excludes BOS/EOS tokens, separators, padding, system/user template tokens, and non-patch multimodal special tokens. In a single-image sequence, it is the image's projected-token span. For multi-image or any-resolution formatting, M is the union of the inserted projected patch-token spans; the boundaries of each span are retained. This rule defines the index mask, not a claim of additional architecture evaluations.

Text states. If $H^\vartheta_{\ell,t}$ is the hidden state at layer ℓ and token t for model ϑ, then

$$\bar{H}^\vartheta_{\ell,Q} = \frac{1}{|Q|}\sum_{t\in Q} H^\vartheta_{\ell,t}, \qquad \bar{H}^\vartheta_{\ell,A} = \frac{1}{|A|}\sum_{t\in A} H^\vartheta_{\ell,t}. \tag{21}$$

The first quantity is evaluated without the image for R_T. The second is evaluated with the full question–image input and the selected alias prefix for R_F. Removing the image in R_T targets the question processing that remains available without visual input. Keeping it in R_F lets the answer-state comparison reflect the combined input. The earlier gradients decide the pressure using both inputs; these penalties then apply it to the specified text-only or joint features.

Visual states: average patches first, then blocks. Let $\mathcal{L}_V$ contain the two highest blocks of the vision encoder. For model $\vartheta \in \{\theta, \theta_0\}$, $V^\vartheta_{\ell,p}$ is the hidden vector of spatial patch token p in block ℓ, and P_ℓ is the number of those patch tokens. We first summarize each block, then give the two summaries equal weight:

$$\bar{V}^\vartheta_\ell = \frac{1}{P_\ell}\sum_{p=1}^{P_\ell} V^\vartheta_{\ell,p}, \qquad \bar{V}^\vartheta = \frac{1}{|\mathcal{L}_V|}\sum_{\ell\in\mathcal{L}_V} \bar{V}^\vartheta_\ell, \qquad |\mathcal{L}_V| = 2. \tag{22}$$

Only spatial patch tokens enter the first mean; padding and non-patch special tokens do not. The second mean is taken over feature vectors, not over separate blockwise cosine penalties. We apply the cosine comparison only after these two means. The first mean summarizes the spatial image features in one block; the second gives each selected block equal weight in the visual summary. We do not normalize the block summaries before averaging them or concatenate them in place of the second mean.

The projector maps visual features into the language model's input space. For its projected patch-token vectors Z^ϑ_m, we separately compute

$$\bar{Z}^\vartheta = \frac{1}{|M|}\sum_{m\in M} Z^\vartheta_m. \tag{23}$$

The image penalty therefore remains

$$R_I = \max\{0, \cos(\bar{V}^\theta, \bar{V}^{\theta_0}) - m_V\} + \eta_Z \max\{0, \cos(\bar{Z}^\theta, \bar{Z}^{\theta_0}) - m_Z\}. \tag{24}$$

The first term encourages the forget image's encoder features to differ from the original model's; the second acts on the visual tokens delivered to the language decoder. Each current-model vector is compared with its frozen-model counterpart, never with the vector from the other representation space. The two terms use different thresholds because they act on different representations. The coefficient η_Z sets their relative strength, and each term becomes zero at its own similarity cutoff. Thus the image route changes both visual processing and its input to the decoder, without requiring either representation to keep moving once its cutoff is reached. Teacher and student use the same two vision blocks, image preprocessing, patch masks, and projected-token layout.

Attention patterns. $B_{\ell,h,a,m}$ is the attention assigned to image position m by answer position a in attention head h at layer ℓ. It records which visual tokens that answer position consults. Here a indexes an answer position, not an alias string. If there are H heads, the pooled pattern is

$$\bar{B}_{\ell,m} = \frac{1}{H|A|} \sum_{h=1}^{H} \sum_{a \in A} B_{\ell,h,a,m}, \qquad m \in M. \tag{25}$$

It retains one entry per image token rather than collapsing all image positions to a scalar. The result summarizes the relative pattern of visual attention across the answer, so its comparison checks how the image is consulted, not just the total attention value. The cosine comparison in R_F uses this vector; the gradient-product sensitivity uses the unpooled tensor. Together, the two fusion terms compare the features formed while producing the alias and the distribution of attention over its image tokens. These complement R_I, which compares the visual representations themselves. Both model passes use the same $a^\dagger$ selected by the current model, not separate teacher- and student-selected answers.

Layer indices. All layer indices are zero-based. For the 32-layer decoder, we use the fixed sets

$$\mathcal{L}_T = \{8, 16\}, \qquad \mathcal{L}_F = \{20, 24, 28, 31\}. \tag{26}$$

We do not recompute these sets using a relative-depth formula. For other decoder depths L, let $\rho \in [0, 1]$ denote a relative depth. We map ρ to the nearest valid zero-based layer index, rounding a half-integer upward:

$$\mathrm{idx}(\rho, L) = \min\big\{L - 1,\ \max\big\{0,\ \big\lfloor \rho(L-1) + \tfrac{1}{2} \big\rfloor\big\}\big\}. \tag{27}$$

Here $\lfloor \cdot \rfloor$ rounds down after adding $1/2$; the outer bounds keep the result between 0 and $L - 1$. The rule preserves approximately where a probe lies in the decoder depth without assuming different decoders have the same number of layers. The selected sets for $L \neq 32$ are

$$\begin{aligned} \mathcal{L}_T &= \{\mathrm{idx}(0.25, L), \mathrm{idx}(0.50, L)\}, \\ \mathcal{L}_F &= \{\mathrm{idx}(0.60, L), \mathrm{idx}(0.75, L), \mathrm{idx}(0.90, L), L - 1\}. \end{aligned} \tag{28}$$

Duplicate indices are removed, and each set is stored in ascending order. The averages in the route penalties use the number of distinct selected layers. For the 32-layer model, utility-side cached layers $\mathcal{L}_U = \{8, 16, 24, 31\}$ do not add another forgetting loss.

C.3 When to run routing and how to update its weights

Score every eligible example before branching. Let $\mathcal{F}_A$ contain the forget examples whose alias bank is nonempty. At each Stage-II visit to $x \in \mathcal{F}_A$, we rescore every alias with the current model and compute $s_{\mathrm{sem}}(x)$. The gate $h(x)$ is one when the score exceeds δ and zero otherwise; $\gamma(x)$ sets the strength of direct alias suppression:

$$h(x) = \mathbf{1}[s_{\mathrm{sem}}(x) > \delta], \qquad \gamma(x) = 0.2 + 0.8h(x), \qquad \delta = \log(0.02). \tag{29}$$

The equality case belongs to the low-residual branch. The threshold is on the length-normalized, teacher-forced probe; it is not a measured 2% rate of semantic recovery. The gate is a fixed decision for the current update, not a differentiable loss term. Whenever training encounters the same example again, it uses fresh scores; the example can therefore return to the high-residual branch. This gate distinguishes how much intervention is currently needed, rather than permanently marking an example as forgotten.

High residual: compute the route intervention. When $h(x) = 1$, we select $a^\dagger$, compute the three sensitivities, and apply the softmax in Eq. equation 6. The resulting current weight vector $\widetilde{w}(x)$ is nonnegative and sums to one. All three penalties remain in the weighted sum; the largest sensitivity does not select a single winning route. A larger sensitivity receives a larger current coefficient. Smaller τ_a concentrates the coefficients more strongly, while larger τ_a makes them closer to equal weights.

We store a separate moving-average weight vector for each forget example. Let n count that example's visits to the high-residual branch, and let $\widetilde{w}_n(x)$ be the current vector at its nth such visit. The operator sg means that a value is used as a constant during differentiation. On the first training occurrence with a score above the threshold, there is no previous weight vector to average. On later such occurrences of the same example, we combine its new vector with its own stored vector:

$$w_1(x) = \mathrm{sg}[\widetilde{w}_1(x)], \qquad w_n(x) = 0.7w_{n-1}(x) + 0.3\,\mathrm{sg}[\widetilde{w}_n(x)], \quad n \geq 2. \tag{30}$$

This is an exponential moving average: 70% of the previous value and 30% of the new value. For example, a weight of 0.6 followed by a new estimate of 0.2 becomes $0.7(0.6) + 0.3(0.2) = 0.48$, rather than abruptly dropping to 0.2. The rule dampens changes between repeated measurements of the same example; it does not mix different examples' weights. We do not initialize the state with uniform weights or share it across identities. The stored state is detached, and no gradient through the sensitivity or EMA calculation enters the parameter update. The route penalties themselves still differentiate through the current model's features and attention. Thus gradients change the model under an already chosen allocation of pressure, rather than changing the allocation to make the loss smaller.

Low residual: only weak semantic cleanup. When $h(x) = 0$, we do not compute g_T, g_I, g_F, form new routing weights, update the sample's EMA, or evaluate any route penalty. An existing EMA state is left unchanged, but is not used for this visit. We use only 20% of the alias-level unlikelihood to discourage the target expressions from returning. The factor 0.2 is the fixed reduced-strength setting, not an estimate that 20% of the knowledge remains. Keeping this small output-level penalty continues discouraging the aliases, while skipping feature repulsion avoids further explicit displacement for a low-score example. This branch still needs the ordinary alias-scoring forward passes; skipping routing does not mean skipping the model's image encoder or attention operations inside those passes.

The exact forget-side loss. For each $x \in \mathcal{F}_A$, the direct alias penalty is

$$\ell_{\text{sem}}(x) = \frac{1}{|\mathcal{A}(x)|} \sum_{a \in \mathcal{A}(x)} u_\theta(a, x). \tag{31}$$

This average directly penalizes every verified way of restating the target. It is distinct from s_{sem}, which gives higher-scoring aliases more influence when deciding whether and where to intervene. The complete forget-side contribution is

$$\ell_{\text{F}}^{(2)}(x) = \begin{cases} \lambda_5 \ell_{\text{sem}}(x) + \lambda_6 \displaystyle\sum_{r \in \{T,I,F\}} w_r(x) R_r(x), & s_{\text{sem}}(x) > \delta, \\ 0.2\lambda_5 \ell_{\text{sem}}(x), & s_{\text{sem}}(x) \le \delta. \end{cases} \tag{32}$$

The upper branch combines direct suppression of target expressions with weighted changes to internal features. The lower branch retains only weak direct suppression because the score no longer calls for explicit feature intervention. This is equivalent to Eq. equation 9. The expression $h \sum_r w_r R_r$ is defined to be zero in the low branch without evaluating that sum. It must not be implemented by computing the full route branch and multiplying its result by zero afterward. This distinction determines both the update and which computations are skipped.

Preservation stays on. The full objective is Eq. equation 10: the mean of these per-example forget contributions plus the retain, retain-distillation, benign-neighbor, and utility losses. Neither h nor γ multiplies a preservation term. All model parameters remain trainable in the low branch: alias suppression and preservation can still change internal features. What stops is the explicit route-repulsion loss, not every possible feature change.

There is also a separate stopping rule inside each active route penalty. A similarity term becomes zero when its own margin is reached, even if $h = 1$ and other route terms remain active. For example, the projected-token penalty can be zero while the encoder-state penalty is still positive. The sample-level gate and the individual similarity thresholds therefore control different things: whether to run the route branch, and which of its comparison terms still apply pressure. Neither is a certificate that every possible expression of the fact has been removed.

C.4 Stage-II update in execution order

The following procedure makes the skipped computations explicit. All references to an EMA state concern the current sample x, not the minibatch as a whole. The same alias-scoring pass supplies both a decision signal, s_{sem}, and an output-suppression loss, ℓ_{sem}; only high-score examples then require the route calculation.

1. Draw forget, retain, neighbor, and utility examples according to the Stage-II source mix. For the forget side, use examples in $\mathcal{F}_A$; cases without aliases follow Appendix A.3.
2. For each eligible forget example, score its aliases under the current model. Compute $s_{\text{sem}}(x)$ and $\ell_{\text{sem}}(x)$ from these scores.

3. If $s_{\text{sem}}(x) \leq \delta$, set its forget loss to $0.2\lambda_5 \ell_{\text{sem}}(x)$. Skip all route calculations, leave its stored EMA unchanged, and continue with the next forget example.

4. Otherwise, select the current $a^\dagger$ and compute g_T, g_I, g_F from s_{sem}. Form the detached current softmax weights. Initialize this example's EMA on its first active visit, or update it using Eq. equation 30.

5. For this active example, compare aligned current-model and frozen-model features to compute R_T, R_I, R_F. In R_I, first patch-average each of the top two vision blocks, then average their two vectors before computing cosine similarity. Add the weighted route loss to the full-strength alias loss.

6. Average the forget contributions over all eligible forget examples in the minibatch, including both high- and low-residual examples. Compute the preservation losses over their respective source subsets and add them at full configured strength. Update θ while keeping θ_0, gates, and routing coefficients fixed.

The averaging denominator is the eligible forget subset, not only the active route subset. Thus a low-residual example contributes its reduced semantic loss without causing the remaining active route terms to be renormalized upward. In particular, reducing the number of active examples does not implicitly rescale λ_6.

C.5 Attention extraction and frozen-model caching

When the main attention kernel does not expose attention maps, a selected high-residual forget minibatch is rerun with attention outputs enabled and gradient checkpointing retained. This provides the answer-to-image tensors needed for the high-residual route calculation. Low-residual examples do not trigger this attention-extraction rerun. The route penalties themselves also require differentiable student features and attention whenever they are evaluated; frozen teacher caches do not replace these student tensors.

The original model stays frozen throughout both stages. Selected teacher hidden states, visual states, projected tokens, and attention patterns can be cached for forget, retain, and neighbor examples. Utility caching stores top-256 teacher logits per response position as described in Appendix B.3. Cached features used in a comparison must match its question, image formatting, and answer prefix. A cache from a different alias cannot stand in for the teacher features of the currently selected $a^\dagger$.

D Training settings and checkpoint selection

D.1 Fixed coefficients

The loss weights, thresholds, temperatures, and learning-rate multipliers are selected on a pilot development split and then kept fixed across the reported datasets, seeds, and ablations unless that parameter is explicitly being ablated. The constants fix the strength of each operation: loss weights set forgetting versus preservation pressure, margins stop individual similarity penalties, and temperatures control how sharply scores affect weighting. The low-residual factor 0.2 and EMA coefficients $0.7/0.3$ retain their fixed default values; their exact values are not consequences of the loss equations. The reported defaults are

Quantity	Setting
Stage-I weights $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$	$(1.0, 1.0, 1.2, 0.6)$
Stage-II weights $(\lambda_5, \lambda_6, \lambda_7, \lambda_8, \lambda_9)$	$(1.0, 0.8, 0.8, 1.2, 0.5)$
Retain critical-token weight	1.5
Distillation temperature T; retain KD weight β	2; 0.5
Probe temperature τ_p; routing temperature τ_a	0.7; 2.0
Projected-token scale η_Z; attention scale η_A	1.5; 0.8
Thresholds $(m_T, m_V, m_Z, m_H, m_A)$	$(0.25, 0.55, 0.35, 0.20, 0.30)$
Routing activation threshold δ	$\log(0.02)$
EMA old/new weights; low-residual alias multiplier	$0.7/0.3$; 0.2
Low-residual route loss	0
EMA initialization	Current weights on the first active visit

D.2 Optimizer, batches, and schedule

Both stages train all parameters with AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.05, and gradient-norm clipping at 1.0. The reported main configuration uses bf16, gradient checkpointing, and eight A100-80GB GPUs, with maximum sequence length 2048. Stage I uses microbatch size two per GPU and effective global batch size 64; Stage II uses one per GPU and effective global batch size 32. The source proportions are

Stage	Forget	Retain	Neighbors	Utility
I	25%	20%	15%	40%
II	30%	15%	15%	40%

The base learning rates are 5×10^{-6} in Stage I and 2×10^{-6} in Stage II. Both use cosine decay, with warmup fractions 3% and 5%, respectively. The component-specific multipliers are

Component	Stage I	Stage II
Multimodal projector	2.0	2.5
Top third of language-model layers	1.0	1.0
Middle third	0.8	0.85
Bottom third	0.6	0.65
Top quarter of vision blocks	0.25	0.30
Remaining vision blocks	0.10	0.10

CLEAR 10% uses 2400 Stage-I steps and 1200 Stage-II steps. UMU-Bench 5%, 10%, and 15% use Stage-I/Stage-II step budgets of 2200/1000, 2400/1200, and 2800/1400. Stage II starts from the selected Stage-I checkpoint, not automatically the last optimization step.

D.3 Validation, transition, and stopping

Validation examples are used to assess saved models and select checkpoints, not to compute training gradients. Fast validation runs every 100 steps and full validation every 300 steps. Validation covers exact forgetting, semantic recovery, retain answers, held-out utility, and harmless-neighbor accuracy/refusal. Validation reformulations are constructed separately from the training alias bank and do not reuse its exact strings as attack prompts.

Balance exact and semantic forgetting. We first average the diagnostics within each validation slice. For model p_ϑ, let $\ell^{\mathrm{val}}_{\mathrm{exact}}(\vartheta)$ be the mean of exact-answer and indispensable-unit hit rates. Let $\ell^{\mathrm{val}}_{\mathrm{sem}}(\vartheta)$ be the mean of alias, held-out paraphrase, and indirect-description success rates. These validation rates are distinct from the per-example training loss $\ell_{\mathrm{sem}}(x)$. We give the two slices equal weight when measuring improvement over the original model:

$$\begin{aligned}\mathrm{ForgetGain}(\theta) = \frac{1}{2}\Big[&\ell^{\mathrm{val}}_{\mathrm{exact}}(\theta_0) - \ell^{\mathrm{val}}_{\mathrm{exact}}(\theta) \\ &+ \ell^{\mathrm{val}}_{\mathrm{sem}}(\theta_0) - \ell^{\mathrm{val}}_{\mathrm{sem}}(\theta)\Big].\end{aligned} \tag{33}$$

A decrease in either recovery slice increases ForgetGain; using both slices avoids selecting a model solely because it stopped producing the original wording. The differences are absolute changes in leakage rates, not percentages divided by base rates. Negative differences are retained when leakage increases. Each slice contributes half of ForgetGain, despite containing different numbers of diagnostics.

Select for forgetting and preservation together. Forgetting alone is not the selection criterion. Let $\Delta\mathrm{RetainLoss}$ and $\Delta\mathrm{UtilityLoss}$ be performance drops relative to the original model, and let $\mathrm{BenignRefusal}$ be the validation refusal rate on harmless-neighbor questions. We select checkpoints using

$$S(\theta) = \mathrm{ForgetGain}(\theta) - \mu_1 \Delta\mathrm{RetainLoss} - \mu_2 \Delta\mathrm{UtilityLoss} - \mu_3 \mathrm{BenignRefusal}, \tag{34}$$

with $(\mu_1, \mu_2, \mu_3) = (1.0, 1.2, 1.5)$. A higher S favors lower recovery, but subtracts the cost of poorer retained answers, lost utility, and harmless-question refusal. The checkpoint passed to Stage II is therefore chosen for forgetting and preservation together, not simply because it is the final Stage-I

update. Training stops when this score has not improved for four full validations, utility falls by more than two absolute points from the best checkpoint, or harmless-neighbor refusal rises for three full validations. The same selected checkpoint is used for all downstream diagnostics of a run.

D.4 Separating total training cost from routing cost

On CLEAR 10%, Stage I takes 13.8 hours and Stage II 8.8 hours: full training totals 22.6 hours, a 63.8% increase over Stage I alone. The attention-extraction rerun takes 0.70 hours, corresponding to 7.9% of Stage-II time. Stage 0 adds 1.845 hours for 584 forget examples after held-out reservation, and peak memory is 67.4 GB per GPU. These measurements apply to this configuration.

E Additional experiments and evaluation details

E.1 Shared evaluation protocol

Benchmark results and additional runs. The primary model is LLaVA-1.5-7B. CLEAR uses the official 10% forget setting; UMU-Bench uses the official 5%, 10%, and 15% settings. We report the benchmarks' official metrics using their released evaluators. Published baseline results in Table **??** remain unchanged. Gold is CLEAR's counterfactual reference, and Base/Origin is the model before unlearning. Our additional experiments rerun named methods or modify specific parts of TRACE. They use the same tokenizer, image preprocessing, chat template, input formatting, and scorer within each comparison. Baselines trained with TRACE's extra data are separate comparisons, not replacements for the published entries.

Training and decoding. The recorded main configuration uses eight A100-80GB GPUs, bf16, gradient checkpointing, maximum sequence length 2048, and effective batch sizes 64 and 32 for Stages I and II. CLEAR uses 2400 Stage-I and 1200 Stage-II steps. UMU's 5%, 10%, and 15% runs use Stage-I/Stage-II step budgets of 2200/1000, 2400/1200, and 2800/1400. Stage II starts from the selected Stage-I checkpoint. The additional generation tests use greedy decoding with temperature zero and top-p one, allowing 64 new tokens for short answers and 192 for descriptive or multi-turn responses. Within a run, the same selected checkpoint is evaluated across its downstream tests. The statistical comparison and every routing variant have three seeds; the sensitivity tables state their own seed counts. Post-training attack results are aggregate means.

Check behavior beyond the retained facts. We evaluate a fixed set of examples from the original instruction-tuning distribution to check whether other useful behavior survives unlearning. We call performance on this set *native utility*; tables shorten it to *Utility* when no other utility score is shown. Performance on a separate fixed evaluation set drawn from public data is *public utility*. Both are evaluation scores, not the training KL loss or the UMU benchmark score. Changing the training set $\mathcal{U}$ leaves these evaluation sets unchanged, so all training variants are tested on the same examples. We also measure accuracy on the evaluated benign-neighbor questions to check whether forgetting an identity prevents harmless answers about the same image. This measures preservation on those questions, not generalization to new benign questions.

E.2 Statistical reruns

TRACE, NPO, SCRUB, and LLMU each use three seeds on the same CLEAR 10% split, with 20 forgotten identities. We resample whole identity blocks 10,000 times to obtain 95% confidence intervals, rather than treating an identity's questions and images as independent observations. Real and native-utility examples use 20 fixed blocks keyed to the identity partition, shared by all methods and seeds. Paired permutation tests use at least 10,000 rounds. Holm correction is applied to each experiment's full comparison family; showing a subset of its tests does not change the adjusted values. Gold and Base remain reference point estimates.

Table 7 shows the joint advantage: TRACE retains knowledge at a level near SCRUB while lowering Forget, unlike NPO, whose low Forget accompanies much lower Retain.

Table 8 separates the primary and secondary paired tests: the Forget comparison with NPO is not significant, whereas the secondary comparison with SCRUB is.

Table 9 checks recovery beyond the original questions and utility on the same checkpoints. *Exact* checks target-string matches on direct questions. *Paraphrase* and *Indirect* test unseen rewordings and clue-based questions, counting recovery of the fact in any wording. Appendix E.4 gives the construction and scoring rules.

Table 7: CLEAR statistical reruns: means with 95% intervals. Gold and Base are reference point estimates; Real, Retain, and FQ are higher-is-better, and Forget is lower-is-better.

Method	Real	Retain	Forget	FQ
Gold	0.500	0.510	0.190	1.000
Base	0.480	0.510	0.350	0.850
NPO	0.461 [0.428, 0.494]	0.142 [0.097, 0.188]	0.111 [0.066, 0.161]	0.761 [0.715, 0.805]
SCRUB	0.491 [0.462, 0.521]	0.520 [0.496, 0.544]	0.359 [0.294, 0.424]	0.851 [0.821, 0.881]
LLMU	0.471 [0.440, 0.502]	0.511 [0.486, 0.535]	0.251 [0.199, 0.310]	0.842 [0.808, 0.874]
TRACE	0.501 [0.471, 0.531]	0.521 [0.497, 0.545]	0.081 [0.045, 0.124]	0.889 [0.856, 0.920]

Table 8: Primary comparisons (above the rule) and secondary comparisons (below it) retain their separate correction families.

Metric	Comparator	TRACE	Baseline	Holm p
Real	SCRUB	0.501	0.491	0.872
Retain	SCRUB	0.521	0.520	0.872
Forget	NPO	0.081	0.111	0.444
FQ	SCRUB	0.889	0.851	0.296
Forget	SCRUB	0.081	0.359	0.0006
Retain	NPO	0.521	0.142	0.0006
FQ	NPO	0.889	0.761	0.0021

Table 9: Held-out recovery and native utility in the statistical reruns: means with 95% intervals. Lower recovery and higher utility are preferred.

Method	Exact	Paraphrase	Indirect	Native utility
NPO	0.147 [0.081, 0.231]	0.196 [0.113, 0.302]	0.183 [0.103, 0.286]	0.455 [0.434, 0.476]
SCRUB	0.298 [0.188, 0.430]	0.331 [0.216, 0.465]	0.287 [0.177, 0.419]	0.494 [0.477, 0.511]
LLMU	0.218 [0.126, 0.331]	0.266 [0.160, 0.389]	0.238 [0.139, 0.357]	0.486 [0.467, 0.504]
TRACE	0.051 [0.020, 0.096]	0.072 [0.034, 0.128]	0.089 [0.043, 0.154]	0.501 [0.486, 0.516]

Table 10 tests those differences: the three recovery comparisons with NPO are significant, while the preservation comparisons with SCRUB are not.

Table 10: Selected auxiliary contrasts, retaining the adjusted values from the full auxiliary test family.

Metric	Comparator	TRACE	Baseline	Holm p
Exact	NPO	0.051	0.147	0.0077
Paraphrase	NPO	0.072	0.196	0.0056
Indirect	NPO	0.089	0.183	0.0104
Utility	SCRUB	0.501	0.494	0.332
Neighbor accuracy	SCRUB	0.903	0.872	0.166
Neighbor refusal	SCRUB	0.041	0.079	0.111

All three held-out recovery comparisons with NPO are significant; Real, Retain, and native utility do not establish superiority over SCRUB, and official Forget is not significantly lower than NPO's. The supported gain is therefore stronger forgetting under new wording alongside the observed retained performance, not tiny improvements on every preserved task.

E.3 Full UMU-Bench results and modality differences

UMU-Bench tests whether target facts are forgotten while other task performance is preserved. Its *Unlearning Completeness* (UC) score summarizes remaining performance on forget tasks; *Model Utility* (UT) summarizes performance to preserve. Lower UC and higher UT are better. The task-level measures are classification accuracy, cloze accuracy, and generation ROUGE-L; utility is evaluated under unimodal and multimodal inputs. Table 11 expands the main UC and UT averages into target, unimodal-utility, and multimodal-utility columns without changing the summary metrics.

TRACE lowers target recovery across classification, cloze, and generation at each forget rate (Table 11): the UC gain is not driven by one answer format. The utility breakdown also shows that higher average UT does not imply superiority on every retained subtask.

Table 11: Full UMU-Bench task results. Class. and Cloze are accuracies; Gen. is ROUGE-L. UC and UT retain their official definitions. All three forget rates and every reported method are shown.

	Target performance ↓				Unimodal utility ↑			Multimodal utility ↑			
Method	Class.	Cloze	Gen.	UC	Class.	Cloze	Gen.	Class.	Cloze	Gen.	UT
Forget 5%											
Origin	.8333	.9133	.9153	.8873	.7772	.8070	.7383	.5054	.2865	.1749	.5482
GA	.7333	.8333	.6435	.7367	.6649	.6884	.4922	.4662	.2876	.1352	.4558
GD	.7067	.7733	.7100	.7300	.6635	.7140	.5148	.4782	.2919	.1336	.4660
KL	.6933	.8533	.5565	.7010	.6474	.6796	.4166	.4641	.2974	.1094	.4358
NPO	.7467	.7600	.6103	.7057	.6733	.5358	.4494	.4597	.2789	.1206	.4196
PO	.6200	.8333	.6914	.7149	.6298	.7126	.2320	.4804	.2800	.0525	.3979
TRACE	.6000	.6767	.5499	.6089	.7065	.7421	.5634	.5342	.2901	.1133	.4916
Forget 10%											
Origin	.8233	.9100	.8564	.8632	.7752	.8078	.7415	.5054	.2865	.1749	.5486
GA	.6467	.6433	.6131	.6344	.6496	.5026	.3895	.4499	.2821	.0928	.3944
GD	.6800	.7467	.7072	.7113	.6615	.6011	.5490	.4499	.2800	.1497	.4485
KL	.6733	.7567	.5773	.6691	.6593	.6256	.3476	.4706	.2952	.0608	.4099
NPO	.6933	.7233	.6802	.6989	.6878	.5348	.4724	.4357	.2789	.1406	.4250
PO	.6500	.7000	.5165	.6222	.5785	.6237	.1967	.4575	.2854	.0552	.3662
TRACE	.5267	.6033	.5122	.5474	.6948	.7184	.5517	.5269	.2878	.1052	.4808
Forget 15%											
Origin	.7622	.9133	.8747	.8501	.7831	.8059	.7431	.5054	.2865	.1749	.5498
GA	.6022	.6111	.5872	.6002	.6784	.4569	.3590	.4815	.2821	.0723	.3880
GD	.5533	.6733	.6065	.6110	.5784	.4847	.3554	.3998	.2810	.1152	.3690
KL	.5933	.6556	.4962	.5817	.6722	.5384	.3133	.4815	.2985	.0656	.3949
NPO	.6600	.7111	.7276	.6996	.7251	.5008	.5573	.4847	.2778	.1526	.4497
PO	.5244	.6978	.5275	.5832	.5725	.6024	.2059	.4684	.2854	.0576	.3654
TRACE	.4800	.5578	.4790	.5056	.6812	.7001	.5368	.5204	.2862	.1025	.4712

Compare the two access conditions. For a task t, let v_t^{uni} and v_t^{multi} be its target-performance scores under unimodal and multimodal evaluation. The task-level gap is their absolute difference,

$$\Delta_t = \left| v_t^{\text{uni}} - v_t^{\text{multi}} \right|. \tag{35}$$

Classification and cloze use accuracy; generation uses ROUGE-L. We report the three task gaps separately and read them together with UC: agreement alone would not establish forgetting if both conditions still recovered the target.

Table 12 tests whether adaptive weighting reduces the difference between input modalities while also lowering target recovery.

Table 12: UMU-Bench 10% routing comparison. Each gap concerns target performance, not unrelated utility questions.

Routing	UC ↓	UT ↑	Class. gap	Cloze gap	Gen. gap
TRACE	0.5474	0.4808	0.054	0.044	0.079
Equal weights	0.5712	0.4690	0.069	0.058	0.095
Shuffled weights	0.5918	0.4607	0.085	0.074	0.119
Text route only	0.6124	0.4632	0.106	0.091	0.132
Image route only	0.5986	0.4567	0.089	0.079	0.126
Fusion route only	0.5843	0.4598	0.080	0.070	0.108

Adaptive weights give lower UC, higher UT, and smaller gaps than equal weights (Table 12). They therefore reduce target recovery while making forgetting less uneven across modalities. This supports distributing pressure according to the residual rather than weighting every route equally.

E.4 Held-out questions and training-recipe comparisons

Reserve inputs before construction. For each of the 20 forgotten identities, we reserve 25% of its text question–answer pairs and 25% of its images before Stage 0. Reserved inputs do not enter forget training, alias construction, or benign-neighbor construction. Evaluation uses direct questions (*Exact*), independently constructed rewordings (*Paraphrase*), and questions referring through attributes,

relations, roles, or dates (*Indirect*). Image-based tests pair these questions with reserved images. Training aliases and evaluation prompts are separate objects: the latter are not inserted into $\mathcal{A}(x)$.

Score the observed answers, not the training probe. Exact recovery uses deterministic target-string normalization: lowercase text, normalize whitespace and simple formatting variants, and strip outer punctuation and article-only differences. A match to the gold answer, a verified indispensable span, or a benchmark-equivalent target string counts as an *Exact* hit. Synonyms are not introduced at this step. Semantic checks extract candidate factual content and apply the same fixed target-equivalence verifier across compared methods. A paraphrase or indirect test succeeds when the answer reveals its target fact, including when it uses the original answer string. The labels refer to the question family, not mutually exclusive types of generated answer. The output-category analysis in Appendix E.8 uses a different, mutually exclusive partition.

Average test instances within identities. Let $N = 20$ be the number of identities in the CLEAR diagnostics and $\mathcal{E}_{i,c}$ the evaluated prompt–fact–image instances for identity i in test family c; image-free instances omit the image. For $e \in \mathcal{E}_{i,c}$, let $H_c(e) \in \{0, 1\}$ record whether the generated answer reveals the target according to that family's exact or semantic rule. Its recovery score is

$$r_{i,c} = \frac{1}{|\mathcal{E}_{i,c}|} \sum_{e \in \mathcal{E}_{i,c}} H_c(e), \qquad R_c = \frac{1}{N} \sum_{i=1}^{N} r_{i,c}. \tag{36}$$

For multi-seed runs we compute the score separately for each seed and then average seeds. Thus each identity contributes a recovery proportion, rather than a single label recording whether any of its prompts succeeded. Exact, Paraphrase, and Indirect use their respective test instances; none is the teacher-forced probe s_{sem}.

TRACE has the lowest direct, paraphrase, and indirect recovery among the named reruns in Table 13. Since these questions and images were excluded from construction and training, the benefit extends beyond the exact inputs used for unlearning.

Table 13: Named methods in the transparent held-out comparison. These are rerun diagnostic results, not copied benchmark scores.

Method	Exact ↓	Paraphrase ↓	Indirect ↓
Base	0.341	0.376	0.338
NPO	0.147	0.196	0.183
DPO	0.119	0.236	0.222
LLMU	0.216	0.255	0.238
SCRUB	0.298	0.331	0.287
Retain FT	0.318	0.356	0.309
TRACE	0.051	0.072	0.089

Compare the two stages. The held-out output-check experiment compares the selected Stage-I checkpoint with full TRACE under the same direct, paraphrase, and indirect test families. Table 14(a) shows which expressions still reveal the fact after Stage I; panel (b) compares the full sequence with omitted-stage and merged-stage alternatives.

Table 14: Two separate stage comparisons: (a) held-out recovery after Stage I versus full TRACE; (b) structural recipe ablations sharing Stage-0 objects. Recovery and refusal are lower-is-better; Retain and utility are higher-is-better.

(a) Held-out output checks

Method	Exact	Para.	Indirect
TRACE	0.051	0.072	0.089
Stage I only	0.084	0.181	0.174
SCRUB	0.298	0.331	0.287
NPO	0.147	0.196	0.183

(b) Training recipes

Recipe	Exact	Retain	Native utility	Neighbor refusal
Stage I only	.084	.515	.487	.071
Stage II only	.173	.498	.474	.093
One-stage merged	.093	.487	.463	.108
TRACE	.051	.521	.501	.041

Stage I leaves paraphrase and indirect recovery at 0.181 and 0.174, whereas the full method gives 0.072 and 0.089. Stage II therefore addresses recovery that remains after the original answer has been suppressed; it is not merely repeating the direct-answer objective. Conversely, Stage II alone leaves higher direct recovery, and the merged objective preserves less utility than the full sequence

(Table 14(b)). The two stages work better together in these tested recipes. This comparison supports the complete sequence, without attributing every difference between the recipes to order alone.

E.5 Changing the route weights

All six routing variants use the same Stage-0 objects, selected Stage-I checkpoint, optimizer, learning rate, scheduler, source mixture, other loss coefficients, and 1200 CLEAR Stage-II updates. No variant retrains Stage I. Each uses three seeds and the same validation cadence, selection rule, and evaluation protocol. The intervention weights are the changed variable.

Equal routing assigns $(w_T, w_I, w_F) = (1/3, 1/3, 1/3)$. The shuffled control computes adaptive vectors and uses a seed-specific shuffle to assign them to other identities; no identity keeps its own vector. It preserves the collection of weight vectors but removes their assignment to the corresponding target. Text-, image-, and fusion-only controls use $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$, respectively. They retain alias suppression and the preservation objectives.

Table 15 reports what changes when only the weights change. Adaptive weighting gives both lower recovery and higher native utility than equal, shuffled, or single-route weighting. Since the targets, other losses, initialization, and update budget are shared, these gains support the choice of weights rather than more supervision or longer training.

Table 15: Routing-only results for the same six variants: official CLEAR scores, held-out recovery, and preservation. Exact/Paraphrase/Indirect and refusal are lower-is-better; Real, Retain, FQ, utility, and neighbor accuracy are higher-is-better.

	(a) Official CLEAR				(b) Recovery ↓			(c) Preservation		
Routing	Real	Retain	Forget	FQ	Exact	Para.	Indirect	Native utility	Neighbor acc.	Neighbor refusal
TRACE	.500	.520	.080	.890	.051	.072	.089	.501	.903	.041
Equal weights	.492	.513	.100	.872	.058	.094	.112	.488	.889	.058
Shuffled weights	.486	.504	.120	.854	.066	.113	.134	.477	.872	.075
Text route only	.489	.509	.137	.844	.067	.137	.160	.486	.884	.064
Image route only	.482	.505	.126	.849	.063	.122	.145	.480	.876	.074
Fusion route only	.486	.507	.112	.859	.061	.110	.130	.483	.881	.069

Test how the fact can be elicited. The question-only test removes the image and asks the original question. The neutral-image test uses the image with a prompt such as "Who is shown in this image?" rather than the original question. The joint test supplies both the original question and image. These are output-level access conditions; the joint test does not isolate an independent internal fusion store. Each condition uses the same answer normalization, semantic verification, and identity-balanced aggregation as the other recovery tests.

Table 16 tests whether suppressing one way of recovering the fact leaves another way effective.

Table 16: Recovery under three access conditions. Lower is better.

Routing	Question only	Neutral image	Question + image
TRACE	0.041	0.068	0.081
Equal weights	0.050	0.089	0.112
Shuffled weights	0.063	0.111	0.139
Text route only	0.035	0.145	0.183
Image route only	0.073	0.056	0.158
Fusion route only	0.069	0.123	0.061

Each single-route control is strongest on its corresponding input condition: text-only gives 0.035 on question-only tests, image-only 0.056 on neutral-image tests, and fusion-only 0.061 on joint tests. But each leaves another condition more recoverable than TRACE. TRACE's largest condition-level score is 0.081, versus 0.183, 0.158, and 0.123 for those controls. Jointly penalizing the three routes keeps recovery low when the input changes, whereas one-route updates leave another condition substantially more recoverable. Equal and shuffled weights are also worse, showing why this pressure should follow each target's residual rather than a fixed assignment.

E.6 GIVING BASELINES THE SAME ADDITIONAL DATA

We compare two augmented versions of NPO and two of SCRUB. The preservation-only version keeps the original baseline objective and adds cross-entropy on the same benign-neighbor pool $\mathcal{D}$ and utility KL on the same filtered $\mathcal{U}$. The all-shared-data version additionally applies phrase suppression on $\mathcal{K}$ and alias unlikelihood on $\mathcal{A}$. The original baseline objective remains active, and no route loss is added. We do not add TRACE's full retain-loss combination beyond the baseline's existing objective. These are one-stage runs with total training-example presentations matched to TRACE's two stages; the augmented baselines are not separately retuned.

Table 17 separates two roles of the extra data: helping the model retain useful answers and teaching it which target expressions to suppress.

Table 17: Same-resource controls on CLEAR. Preservation adds benign-neighbor supervision and utility distillation; all shared data additionally supplies span and alias suppression. Utility and neighbor accuracy are higher-is-better; refusal is lower-is-better.

	Official CLEAR				Held-out recovery ↓			Preserved behavior		
Variant	Real	Retain	Forget	FQ	Exact	Para.	Indirect	Native utility	Neighbor acc.	Neighbor refusal
NPO	.460	.140	.110	.760	.147	.196	.183	.455	.816	.136
NPO + preservation	.483	.405	.162	.819	.169	.232	.219	.486	.878	.072
NPO + all shared data	.478	.392	.092	.838	.082	.126	.154	.478	.861	.085
TRACE	.500	.520	.080	.890	.051	.072	.089	.501	.903	.041
SCRUB	.490	.520	.360	.850	.298	.331	.287	.494	.872	.079
SCRUB + preservation	.500	.523	.352	.858	.309	.343	.299	.500	.892	.054
SCRUB + all shared data	.492	.504	.194	.873	.141	.185	.219	.493	.881	.066

Table 18 checks whether the same additions also prevent recovery when the input changes. Even with all four sets, NPO leaves question–image recovery of 0.226, versus 0.081 for TRACE; its UMU UC is also higher and UT lower. Providing the same target and preservation examples therefore does not by itself reproduce TRACE's control of multimodal recovery.

Table 18: Shared-data baselines under CLEAR access tests and UMU-Bench. All recovery scores and UC are lower-is-better; UT is higher-is-better. The panels are separate benchmark evaluations.

(a) CLEAR input conditions

Variant	Question only	Neutral image	Question + image
NPO	0.134	0.229	0.274
NPO + preservation	0.151	0.257	0.309
NPO + all shared data	0.083	0.175	0.226
TRACE	0.041	0.068	0.081
SCRUB	0.305	0.329	0.374
SCRUB + preservation	0.314	0.341	0.389
SCRUB + all shared data	0.144	0.254	0.309

(b) UMU-Bench, 10%

Variant	UC ↓	UT ↑
NPO	0.6989	0.4250
NPO + preservation	0.7190	0.4615
NPO + all shared data	0.6315	0.4510
TRACE	0.5474	0.4808

The additions help with the behaviors they target (Table 17): preservation data raise NPO's native utility from 0.455 to 0.486 and benign-neighbor accuracy from 0.816 to 0.878. Adding verified targets then lowers paraphrase recovery from 0.232 to 0.126 and indirect recovery from 0.219 to 0.154; SCRUB shows the same direction. However, both augmented methods still leak more than TRACE. The data thus supply useful supervision, but how that supervision is used also matters. This comparison supports the full training procedure; the routing-only experiment separately identifies adaptive weighting as one contributor.

E.7 ATTACKS CONSTRUCTED AFTER TRAINING

The post-training suite is separate from the held-out paraphrase and indirect tests above. For every forgotten identity, we construct five families with three prompts each after training has finished. No prompt contains the exact gold answer, a verified target span, or a training-alias string. Prompts must still concern the target fact and must not introduce false facts. Applicable attacks use the reserved images. The same resulting suite is used for all compared methods, without training or checkpoint selection on its prompts.

Five ways to seek the fact. Long-tail paraphrases use new definitions, roles, attributes, or relations. Indirect visual questions ask through clothing, scene, action, or other visible clues. Two-hop questions combine clues such as a role and a date. Candidate questions use anonymous Candidate A/B/C labels with an evaluation mapping to the target. Two-turn dialogues begin with a harmless question and then use its answer to seek the identity or fact. An evaluated dialogue reveals the target when either turn does so; this within-dialogue rule is separate from averaging evaluations within identities.

Comparison and scoring. Stage I tests what direct suppression leaves recoverable. The no-route-loss model in this suite starts from the same Stage-I checkpoint as its full-TRACE counterpart and sets $\lambda_6 = 0$, retaining aliases and preservation losses. It is the attack-suite ablation, not the separately trained sensitivity-sweep ablation. Semantic recovery counts answers that reveal the target fact, including those that use its original wording. We report this recovery rate, exact recovery, refusal, and wrong-fact rates, averaging attack outcomes within each identity and then across identities. Family-specific tables use the evaluated instances of each family rather than an identity-level logical OR over all its prompts. The tables report aggregate means for this suite.

Table 19 checks whether TRACE's advantage survives new ways of asking for the target, rather than only the finite expressions used in training.

Table 19: Post-training attacks: (a) aggregate recovery and output behavior, (b) recovery by attack family, and (c) recovery by input condition. Lower is better throughout. No route loss denotes this attack-suite ablation, not the dedicated sensitivity run.

Metric	Base	Stage I only	No route loss	TRACE	NPO	SCRUB
(a) Aggregate scores						
Exact	0.343	0.100	0.062	0.055	0.156	0.307
Semantic hit	0.436	0.236	0.132	0.095	0.229	0.376
Refusal	0.014	0.073	0.052	0.044	0.138	0.080
Hallucination	0.032	0.065	0.046	0.037	0.116	0.058
(b) Attack families						
Paraphrase	0.424	0.260	0.137	0.091	0.241	0.360
Visual clues	0.437	0.268	0.160	0.098	0.255	0.397
Two-hop	0.461	0.286	0.183	0.111	0.287	0.412
Candidates	0.386	0.224	0.131	0.084	0.221	0.349
Two-turn	0.489	0.310	0.199	0.124	0.303	0.438
(c) Input conditions						
Text only	0.424	0.260	0.137	0.091	0.241	0.360
Neutral image	0.437	0.268	0.160	0.098	0.255	0.397
Joint attack	0.480	0.304	0.198	0.122	0.302	0.430
Two-turn joint	0.489	0.310	0.199	0.124	0.303	0.438

Table 19(a) gives semantic recovery 0.095 for TRACE, versus 0.236 after Stage I and 0.132 without the route loss; TRACE also outperforms NPO and SCRUB. The no-route model still receives alias suppression, so its higher recovery shows that the internal penalties add resistance beyond lowering the probabilities of verified answers. That benefit extends to combining clues and using dialogue: two-hop and two-turn recovery are 0.111 and 0.124 for TRACE, versus 0.183 and 0.199 without the route loss (panel (b)). The alias bank therefore need not enumerate these future questions for the learned intervention to help on them. This is generalization to the tested attacks, not a guarantee for every possible prompt.

E.8 Outputs and benign-neighbor behavior

For CLEAR's benchmark forget queries, each output receives exactly one of six labels. Exact target recovery is checked first. Semantic-only recovery reveals the fact without the target string. The remaining categories are refusal, empty or degenerate text, wrong non-target facts, and coherent non-target answers. The semantic-only output category excludes exact recovery; it is not interchangeable with a semantic-hit test that accepts any recovered expression. All models share the generation and classification procedure.

Table 20 asks whether preventing identity disclosure also prevents harmless answers about the same image.

Table 20: Forget-query outputs and evaluated benign-neighbor behavior. Each model's six shares in (a) sum to one. In (b), accuracy is higher-is-better; refusal, wrong facts, and target leakage are lower-is-better. Length counts generated tokens.

	(a) Forget-query output shares						(b) Evaluated benign neighbors				
Method	Exact	Semantic only	Coherent	Wrong fact	Refusal	Degen-erate	Accuracy	Refusal	Wrong fact	Target leak	Length
Gold	.164	.030	.700	.083	.020	.003	.896	.026	.056	.018	15.4
Base	.332	.069	.408	.169	.018	.004	.884	.018	.073	.064	15.6
TRACE	.052	.029	.789	.086	.041	.003	.903	.041	.047	.011	15.8
Stage I only	.080	.156	.552	.136	.071	.005	.886	.071	.059	.027	15.2
SCRUB	.319	.083	.432	.130	.032	.004	.872	.079	.068	.061	14.9
NPO	.103	.076	.464	.194	.151	.012	.816	.136	.109	.033	11.8

Table 20(a) shows that coherent non-target answers account for 78.9% of TRACE's forget-query outputs; refusals and degenerate text account for 4.1% and 0.3%. Wrong non-target facts still occur in 8.6% of outputs. Panel (b) gives evaluated benign-neighbor accuracy of 90.3%, with target leakage 0.011 on those questions. TRACE thus usually avoids the target without refusing and still answers the evaluated harmless questions correctly. This supports the intended separation between the identity to remove and the visual behavior to preserve. It does not establish equality to Gold's output distribution, and a coherent non-target answer is not necessarily correct for the original identity question.

Human audit. A stratified audit covers 200 outputs across methods and categories, with rare refusal, empty, and semantic-recovery cases sampled more heavily. Cohen's κ is 0.884; conservative relabeling changes each audited metric by at most 0.009. Possible target disclosures are assigned to semantic recovery, possible abstentions to refusal, and wrong identities without target disclosure to wrong non-target facts. The small changes after relabeling support the output-based conclusion: low disclosure is not mainly refusal or degenerate generation. This audit checks category assignments; the benchmark significance tests are separate.

E.9 Construction choices and training exclusions

Compare complete construction choices. Full-answer forgetting replaces span-level targets with the entire answer. The naive-bank alternative replaces the verified aliases with its generated paraphrase bank. This comparison evaluates the full construction recipes, not verification in isolation.

Table 21 gives lower exact recovery and higher retain and utility scores with the full construction than with full-answer suppression or a naive alias bank. Careful target construction therefore improves both forgetting and preservation. This supports the complete construction choice, not verification isolated from bank composition or training exposure.

Table 21: Construction-recipe comparison. The span and verified-bank recipe has stronger preservation than the tested alternatives.

Construction	Exact ↓	Retain ↑	Native utility ↑	Neighbor refusal ↓
Full-answer targets	.074	.479	.452	.126
Naive paraphrase bank	.056	.511	.484	.057
Verified construction	.051	.521	.501	.041

Account for the full pool separately from the held-out run. The full-pool accounting experiment starts with 779 rows. Of these, 768 enter forget training and 11 are excluded: five unsupported yes/no cases and six extraction or fallback failures. All 11 excluded rows remain in the full-pool diagnostic evaluation, so construction failures are not hidden by omitting them from the test. There are 752 Stage-II-eligible rows and 16 rows that receive Stage I but skip Stage II. Table 22 separates the primary categories from overlapping processing flags; the five primary categories account for all 779 rows.

The held-out evaluation and compute experiment starts from the same 779-row pool but reserves approximately 25% before Stage 0, leaving 584 processed examples. Its fallback and Stage-II-skipped rates are 6.2% and 3.8%. They belong to that reserved-data run, not to the full-pool counts above.

Table 22: Full-pool accounting. The first five rows partition all 779 examples; the last two are overlapping flags.

Category	Rows	Share (%)
Ordinary construction	666	85.5
Yes/no with supporting proposition	18	2.3
Unsupported yes/no	5	0.6
Atomic answers	84	10.8
Extraction/fallback failure	6	0.8
Fallback aliases used (overlapping flag)	38	4.9
Stage II skipped (overlapping flag)	16	2.1

Give baselines the same training filter. The common-filter experiment excludes the same 11 rows from NPO and SCRUB forget training, while evaluating all methods on the unchanged 779-row pool. It does not remove the 16 Stage-II-skipped rows because those remain usable for Stage-I training. The results use three seeds and identity-level bootstrap intervals.

Table 23 checks recovery and preservation with the common training filter and unchanged evaluation.

Table 23: Common training-filter comparison with evaluation unchanged. Values give means and reported 95% interval half-widths. Recovery and neighbor refusal are lower-is-better; native utility is higher-is-better.

Method	Exact	Paraphrase	Indirect	Native utility	Neighbor refusal
NPO-common-filter	0.154 ± 0.062	0.206 ± 0.075	0.192 ± 0.072	0.459 ± 0.025	0.128 ± 0.055
SCRUB-common-filter	0.308 ± 0.083	0.341 ± 0.087	0.299 ± 0.083	0.493 ± 0.018	0.079 ± 0.038
TRACE-common-filter	0.052 ± 0.033	0.074 ± 0.041	0.090 ± 0.046	0.500 ± 0.017	0.042 ± 0.029

Under the shared filter, TRACE gives paraphrase recovery 0.074, versus 0.206 for NPO and 0.341 for SCRUB; indirect recovery is likewise lower. TRACE's advantage therefore survives giving the baselines the same exclusion rule. It is not explained by that training filter or by dropping construction failures from evaluation.

E.10 Replacing the source of utility data

We compare *Native-U*, which draws the training set $\mathcal{U}$ from the original instruction-tuning mixture, with *Public-U*, which replaces that set with filtered public examples. The target spans, alias banks, benign neighbors, core loss weights, thresholds, training steps, and selection rule remain unchanged. The public pool covers multimodal instruction following, visual QA, captions, OCR, and grounding/reasoning. The original model supplies the distillation predictions using provided response prefixes or teacher-generated targets. Two further controls use the retain set in place of $\mathcal{U}$ (*Retain-set U*) or omit utility supervision (*No U*), while keeping the other preservation losses. Table 24 gives the public-source counts at each selection step.

Table 24: Public utility-data sources before filtering, after filtering, and after selection. The selected total is 50,000.

Source	Candidates	After filtering	Selected
Public LLaVA-Instruct / general instruction	52,500	51,486	17,500
VQAv2 / visual QA	45,000	44,394	15,000
COCO caption / image description	22,500	22,046	7,500
OCR-VQA + TextVQA / OCR-oriented QA	15,000	14,778	5,000
GQA + grounding/reasoning mixed	15,000	14,757	5,000
Total	150,000	147,461	50,000

Filtering removes 137 examples by direct target matching, 1,421 by text-semantic similarity, and 981 by image-description similarity, leaving 147,461 candidates for the 50,000-example set (Table 24). These public sources supply a target-filtered preservation set without access to the original training mixture. The next comparison tests whether this replacement still preserves useful answers.

Table 25 separates the need for utility supervision from the need for the original training data.

Table 25: Utility-source replacements with targets and training settings fixed. Panels distinguish CLEAR metrics, held-out target recovery/native utility, public-slice and neighbor behavior, and UMU-Bench. Forget, recovery and refusal are lower-is-better; retained-performance and utility scores are higher-is-better.

	(a) CLEAR				(b) Held-out / native slice				(c) Public / neighbors			(d) UMU	
Source	Real	Retain	Forget	FQ	Exact	Para.	Indirect	Native utility	Public utility	Neighbor acc.	Neighbor refusal	UC	UT
Native-U	.500	.520	.080	.890	.051	.072	.089	.501	.498	.903	.041	.5474	.4808
Public-U	.496	.516	.086	.880	.055	.080	.098	.493	.505	.892	.050	.5549	.4746
Retain-set U	.488	.508	.092	.866	.052	.093	.113	.452	.466	.876	.066	.5634	.4552
No U	.468	.462	.065	.842	.046	.081	.102	.421	.438	.805	.143	.5387	.4301

Public-U gives native utility 0.493 versus Native-U's 0.501 and no-U's 0.421, while keeping CLEAR Retain at 0.516 and Forget at 0.086 (Table 25). Thus public examples can support the preservation term while the forget objective and routing rule remain unchanged. Removing that term gives an even lower raw Forget of 0.065, but Retain falls to 0.462 and benign-neighbor refusal rises to 0.143. Utility supervision is therefore preventing loss of other useful behavior, not merely weakening target suppression. The same distinction holds on UMU: Public-U gives UC 0.5549 and UT 0.4746, close to Native-U's 0.5474 and 0.4808, whereas no-U gives UT 0.4301 (panel (d)). The original mixture is helpful, but the preservation mechanism does not require it as its only data source.

E.11 Dedicated sensitivity sweep

We start from the same selected Stage-I checkpoint and keep the Stage-0 objects and 1200 Stage-II updates fixed. Each run changes one parameter group, without using the sweep to choose a new default. The route-loss coefficient λ_6 controls the strength of the weighted internal penalties. The preservation scale α_P multiplies λ_7, λ_8, and λ_9 together. The routing temperature τ_a controls how strongly the soft weights favor larger sensitivities. Default values are $\lambda_6 = 0.8$, $\alpha_P = 1$, and $\tau_a = 2$. The zero-route-loss row is the dedicated sensitivity run; the post-training attack ablation is evaluated separately in Appendix E.7.

Table 26 checks how forgetting and preservation change when their weights change.

Table 26: Dedicated CLEAR sensitivity sweep. The same setting and seed counts apply to recovery and preservation columns. Retain, utility and neighbor accuracy are higher-is-better; refusal is lower-is-better. One-seed rows are descriptive checks.

		Recovery ↓				Preserved behavior			
Setting	Seeds	Exact	Question only	Neutral image	Question + image	Retain	Native utility	Neighbor acc.	Refusal
TRACE (default)	3	.051	.041	.068	.081	.521	.501	.903	.041
$\lambda_6 = 0$ (sweep)	3	.064	.062	.119	.151	.524	.507	.910	.036
$\lambda_6 = 0.4$	3	.056	.046	.083	.101	.522	.504	.906	.039
$\lambda_6 = 1.2$	3	.049	.039	.063	.074	.516	.494	.889	.055
$\alpha_P = 0.5$	1	.045	.036	.058	.069	.503	.477	.858	.076
$\alpha_P = 2$	1	.060	.049	.087	.104	.527	.512	.919	.030
$\tau_a = 1$	1	.053	.044	.071	.090	.517	.496	.891	.054
$\tau_a = 4$	1	.057	.049	.086	.105	.515	.491	.884	.060

Table 27 repeats representative settings on UMU-Bench: stronger route pressure lowers UC but also lowers UT, matching the forgetting–preservation trade-off.

With $\lambda_6 = 0$, alias suppression remains but question–image recovery reaches 0.151, versus 0.081 at the default (Table 26). The route penalty therefore reduces recovery that direct alias suppression leaves behind; the no-route variant's higher utility, 0.507 versus 0.501, also shows why forgetting and preservation must be read together. Increasing λ_6 from 0.4 to 1.2 lowers question–image recovery from 0.101 to 0.074 but also lowers native utility from 0.504 to 0.494. Changing preservation strength reverses that balance: $\alpha_P = 0.5$ gives recovery 0.069 and utility 0.477, whereas $\alpha_P = 2$ gives 0.104 and 0.512. The two sets of losses therefore have distinguishable roles: stronger route penalties suppress more recovery, while stronger preservation retains more useful behavior. The tested sharper

Table 27: Representative UMU-Bench 10% sensitivity checks.

Setting	Seeds	UC ↓	UT ↑
TRACE (default)	3	0.5474	0.4808
$\lambda_6 = 0$ (sweep)	1	0.6020	0.4920
$\lambda_6 = 1.2$	1	0.5410	0.4720
$\alpha_P = 0.5$	1	0.5280	0.4590
$\alpha_P = 2$	1	0.5680	0.4910
$\tau_a = 1$	1	0.5540	0.4760
$\tau_a = 4$	1	0.5700	0.4740

and softer routing temperatures both give higher recovery and lower utility than the default. These observations support balancing the losses, not simply increasing forgetting pressure; one-seed rows remain descriptive checks.

E.12 PREPROCESSING, TRAINING, AND ATTENTION-EXTRACTION COST

We separate the recorded offline construction time from training time, and separate attention-extraction forward passes from the rest of Stage II. Measurements use eight A100-80GB GPUs with bf16, sequence length 2048, gradient checkpointing, and effective batch sizes 64 and 32 for Stages I and II. The CLEAR compute run processes 584 examples after the held-out reservation described in Appendix E.9. UMU counts forgotten profiles, each with six canonical Stage-0 facts: its 5%, 10%, and 15% settings process 150, 300, and 450 examples.

Table 28 separates training time from the extra forward passes used to extract attention and records how often the gate activates. Here a forget visit is one presentation of a forget example during Stage II; it is active when its residual score exceeds the routing threshold.

Table 28: Cost across forget settings: (a) offline construction, (b) optimizer updates and training time, (c) peak memory per GPU and training-time increase over Stage I, and (d) attention extraction. Stage-0 time is additional; attention time is already included in Stage II.

Metric	CLEAR 10%	UMU 5%	UMU 10%	UMU 15%
(a) Stage-0 preparation				
Processed examples	584	150	300	450
Stage-0 time (h)	1.845	0.410	0.830	1.260
Stage-0 h / 1,000	3.160	2.733	2.767	2.800
Aliases per example	6.34	6.58	6.56	6.52
Neighbors per example	2.71	2.84	2.83	2.81
(b) Training				
Stage-I updates	2400	2200	2400	2800
Stage-II updates	1200	1000	1200	1400
Stage-I time (h)	13.80	12.65	13.80	16.07
Stage-II time (h)	8.80	7.31	8.78	10.36
Total training time (h)	22.60	19.96	22.58	26.43
(c) Memory and training increase				
Stage-I-only peak / GPU	62.7 GB	62.3 GB	62.6 GB	62.9 GB
Full-training peak / GPU	67.4 GB	67.0 GB	67.3 GB	67.8 GB
Training-time increase	63.8%	57.8%	63.6%	64.5%
(d) Attention extraction				
Forget visits	11520	9600	11520	13440
Active visits	4746	3629	4562	5738
Active share	41.20%	37.80%	39.60%	42.69%
Attention forward time (h)	0.70	0.52	0.67	0.87
Share of Stage-II time	7.9%	7.1%	7.6%	8.4%

The gate restricts routing to fewer than half of these visits in every setting (Table 28(d)). On CLEAR, the extra attention-extraction pass takes 0.70 hours within the 8.8-hour Stage-II total. Across UMU rates, preparation cost per 1,000 examples and peak memory remain close (panels (a,c)). Selective attribution therefore limits routing work, but does not remove the cost of second-stage learning.