MODULE: LEARNING MODULAR EXPERT REPRESENTATIONS FOR LOCALIZED MACHINE UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning aims to remove the influence of specified training data while preserving performance on retained data. Existing methods typically modify models post hoc and often require broad parameter updates, resulting in substantial computational cost and collateral degradation. We study whether structured deletion requests can instead be made localizable to small expert subsets. We introduce *Modular Unlearning via Localized Mixture of Experts* (ModULE), a deletion-aware MoE framework that promotes *routing-space localizability*. During learning, ModULE combines sparse routing, balanced expert utilization, and centered kernel alignment to encourage sparse and differentiated expert pathways. Given a deletion request, it selects experts that provide high forget-route coverage with limited retain exposure and identifies retained examples whose routes intersect the selected experts. Unlearning then updates only the selected experts, while output and route distillation protect exposed retained examples and limit layer-wise routing shifts. Under hard routing and frozen shared parameters, retained inputs whose pathways avoid the updated experts remain unchanged. Experiments on structured class- and domain-level deletion, with random deletion as a stress test, show that ModULE improves the forgetting-utility trade-off while updating fewer parameters than matched dense and conventionally trained MoE baselines.

1 INTRODUCTION

Machine unlearning (MU) aims to remove the influence of designated training data from a learned model while preserving the utility associated with the remaining data (Li et al., 2025; Huang et al., 2025). This capability is increasingly relevant to the right to be forgotten (Nguyen et al., 2026b; Liu et al., 2022), copyrighted-data removal (Lee et al., 2026; Wang et al., 2026), and the removal of unsafe or prohibited knowledge (Dong et al., 2026; Zhong et al., 2026; Pawelczyk et al., 2025). Ideally, an unlearned model should behave similarly to a model retrained from the same initialization without the forget set. Achieving this objective is difficult for modern deep networks because the influence of a sample, class, user, or domain can be distributed across many parameters and intertwined with retained information (Khalil et al., 2026; Bourtoule et al., 2021). Consequently, removing the influence of even a small subset of training data may require global model updates or expensive retraining (Chen et al., 2023a). Such updates incur substantial computational cost and can unintentionally damage retained capabilities, leading to collateral forgetting and unstable forgetting-utility trade-offs (Choi & Na, 2025; Nguyen et al., 2026a).

Most existing MU methods treat forgetting as a post-training optimization problem. Given a trained model and a deletion request, they modify the model through retraining (Wu et al., 2020; Kim & Woo, 2022), parameter perturbation (Khalil et al., 2025; Foster et al., 2024; Fan et al., 2024), gradient-based correction (Nguyen et al., 2026a; Zhou et al., 2026; Zhang et al., 2022; Patel & Qiu, 2025), or distillation-based objectives (Chen et al., 2025; Vasilev et al., 2025; Lin et al., 2023; Zade et al., 2026). Although these approaches can reduce the cost of full retraining, they are generally applied to architectures whose internal structures were not intended to support future deletion. The unlearning algorithm must therefore operate on representations in which forget-related and retain-related influences may already be globally entangled. This suggests that unlearning performance depends not only on the post-training optimization algorithm, but also on how predictive computation is organized during the model's original learning stage.

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.

Mixture-of-Experts (MoE) architectures provide a natural form of computational modularity because each input activates only a sparse subset of expert parameters. Recent MoE unlearning methods exploit this modular structure by identifying forget-relevant experts, restricting unlearning updates, stabilizing routing decisions, or calibrating retain supervision at the expert level (Zhuang et al., 2025; Zhu et al., 2026; Xie et al., 2026). Focusing on unlearning an already trained MoE model, these studies collectively demonstrate that expert routing is an important component of MoE unlearning. However, a standard MoE is typically optimized for predictive performance and conditional computation, rather than for the localizability of future deletion requests. Sparse activation alone does not guarantee that a deletion request is concentrated within a small expert subset, that the selected experts have limited responsibility for retained data, or that routing remains stable when upstream experts are modified. This work investigates whether an MoE model can be trained so that a structured forget set concentrates its predictive computation on a small subset of experts. We call a request *routing-localizable* when a small expert subset covers most forget routes but has limited exposure to retained data. Routing identifies the candidate experts, while controlled interventions verify whether they actually influence forget predictions. The formulation covers class-, domain-, user-, and source-level requests; our experiments evaluate class and domain deletion and use random deletion as a structure-free stress test.

Based on this principle, we introduce *Modular Unlearning via Localized Mixture of Experts* (ModULE), which combines a shared encoder, a learned router, and multiple expert modules. During learning, routing sparsity encourages small per-input pathways, load balancing prevents routing collapse, and a centered kernel alignment (CKA)-based objective discourages similar expert representations. These objectives promote sparse routing, balanced utilization, and representational differentiation, respectively. Given a deletion request, ModULE uses active gated routing mass to select experts that provide sufficient forget coverage with limited retain exposure. Their routing overlap with the retain set defines an *at-risk retain set*. During unlearning, updates are restricted to the selected experts only while retain supervision and output distillation protect the at-risk examples. For multi-layer MoEs, layer-wise route distillation limits downstream routing shifts caused by changes in upstream experts. Under hard routing and frozen shared parameters, ModULE provides a conditional pathway-locality guarantee: a retained input whose original pathway avoids all updated experts retains the same computation and output. This guarantee only pertains to computational locality and does not imply that the selected experts contain exclusive knowledge or that forget-related information cannot be recovered from other components. In summary, our contributions are as follows:

- We formulate *routing-space localizability* through forget coverage and retain exposure, providing a criterion for determining whether a deletion request can be localized to a small expert subset.
- We propose ModULE, which combines sparse routing, load balancing, and centered CKA regularization to improve the routing localizability of structured deletion requests.
- We develop a localized unlearning procedure that selects the expert subset, constructs its at-risk retain set, and limits layer-wise routing shifts while updating only the selected experts.
- We establish a conditional pathway-locality result and validate ModULE through routing analyses, causal expert interventions, and comparisons with matched MoE and retraining baselines.

2 Related Work

Machine unlearning. Machine unlearning seeks to remove the influence of specified training data without retraining a model from scratch. Early approaches accelerate retraining through data partitioning, cached updates, or deletion-aware training procedures Cao & Yang (2015); Bourtoule et al. (2021); Wu et al. (2020); Yan et al. (2022); Liu et al. (2022). Approximate methods instead modify an already trained model using parameter perturbation, gradient-based optimization, boundary manipulation, regularization, or knowledge distillation Kodge et al. (2024); Fan et al. (2024); Foster et al. (2024); Chourasia & Shah (2023); Chen et al. (2023a); Kurmanji et al. (2023); Lin et al. (2023). Although these methods reduce unlearning cost, they generally do not control how the original model distributes forget-related influence across its parameters. Recovery and verification studies further show that behavioral forgetting may not imply complete removal of the targeted influence Marchant et al. (2022); Bertran et al. (2024); Łucki et al. (2025); Xiao et al. (2025); Chien et al. (2024). ModULE is complementary to these unlearning algorithms: it modifies the model's initial learning procedure to support more localized future interventions.

Localized and unlearning-by-design approaches. Several methods localize unlearning after training by identifying salient parameters, dampening selected weights, muting neurons, or applying parameter-efficient updates Jia et al. (2023); Fan et al. (2024); Foster et al. (2024); Basu Roy Chowdhury et al. (2025); Qin et al. (2026). These approaches demonstrate the value of parameter locality, but the localization is typically discovered after the model has already been trained. Recent unlearning-by-design methods instead modify the model architecture or training process in anticipation of future deletion requests. MUNKEY separates instance-specific information into a key-addressed memory, allowing unlearning through key deletion Laguna et al. (2026), while deep semi-parametric models retain sample-associated components that can be removed at inference time Zheng et al. (2026b). Apart from these memory approaches, ModULE remains a parametric MoE model and targets structured requests that can be localized within its routed expert computation. Its goal is not instant sample removal, but reducing the scope of parameter updates required by subsequent unlearning.

Mixture-of-experts and MoE unlearning. MoE models use learned routers to activate a sparse subset of expert parameters for each input, enabling conditional computation and increased model capacity Shazeer et al. (2017); Fedus et al. (2022); Riquelme et al. (2021). Auxiliary balancing objectives are commonly used to prevent routing collapse, but standard MoEs are optimized primarily for prediction and computational efficiency rather than future unlearning. Sparse routing therefore does not by itself ensure that a deletion request is concentrated within a small expert subset or that modifying this subset has limited influence on retained data. Recent methods directly study unlearning in pretrained MoE models. SEUF Zhuang et al. (2025) identifies an expert strongly associated with the forget target and uses an anchor loss to stabilize its activation during unlearning. GRIP constrains router updates geometrically to prevent unlearning from producing superficial forgetting through route changes Zhu et al. (2026). TRACE identifies forget-critical experts from routing statistics and reweights retain supervision according to their forget-side and retain-side activation frequencies Xie et al. (2026).

3 MODULE: MODULAR UNLEARNING VIA LOCALIZED MIXTURE OF EXPERTS

3.1 PROBLEM SETUP AND SCOPE

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ denote the original training set. A deletion request specifies a forget set $\mathcal{D}_f \subset \mathcal{D}$, while the remaining data form the retain set $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. Given a fixed training procedure $\mathcal{A}$, we denote the original model by $\theta_0 = \mathcal{A}(\mathcal{D})$, and the retain-only retrained model by $\theta_r = \mathcal{A}(\mathcal{D}_r)$. The objective of approximate unlearning is to obtain a model θ_u whose behavior approaches that of θ_r without repeating a full training procedure.

ModULE addresses this objective by encouraging deletion requests to become localized in the routing space of an MoE. Informally, a request is *routing-localizable* if a small subset of experts covers most of the active routing mass of $\mathcal{D}_f$ while being used by relatively few retained examples. Such localization allows the unlearning update to be restricted to selected experts, reducing the potential effect on retained predictions. We formalize detailed problem setup in Section 3.4.

Our primary scope is approximate unlearning of structured downstream requests, such as class-, domain-, user-, or source-level deletion. Random-sample deletion is included as a structure-free stress test, since arbitrary samples may not share a localized routing pattern. Routing locality describes how examples use expert pathways and does not imply that the selected experts contain unique knowledge or that forget-related information is absent from other model components.

3.2 FRAMEWORK OVERVIEW

ModULE consists of shared model components, L sparsely routed MoE layers, and a shared prediction head. Let $u_{\ell-1}(x)$ denote the representation entering MoE layer ℓ. The router produces $\pi_\ell(x) = \mathrm{softmax}\left(r_{\theta_\ell}(u_{\ell-1}(x))\right)$, and selects the top-$k$ active experts $\mathcal{E}_{\ell,k}(x) = \mathrm{TopK}\left(\pi_\ell(x), k\right)$. Let $h_{\ell,m}(\cdot;\psi_{\ell,m})$ denote expert m at layer ℓ, parameterized by $\psi_{\ell,m}$. Each expert maps the incoming representation to an expert-specific output. The output of the ℓ-th MoE layer is

$$z_\ell(x) = \sum_{m \in \mathcal{E}_{\ell,k}(x)} g_{\ell,m}(x) h_{\ell,m}\left(u_{\ell-1}(x); \psi_{\ell,m}\right). \tag{1}$$

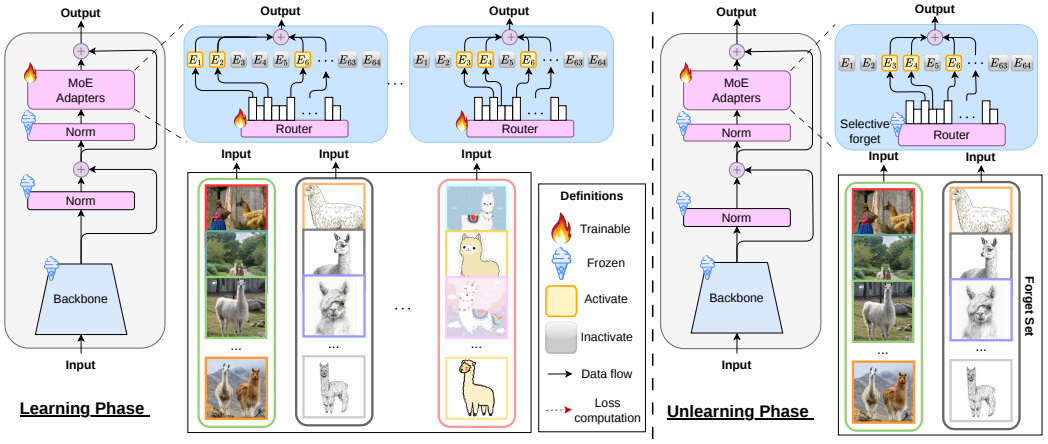


Figure 1: Overview of ModULE. During learning, the backbone remains frozen while the router and experts are trained to encourage group-dependent routing and differentiated expert representations. Given a deletion request, ModULE selects experts based on forget coverage and retain exposure, then updates only these experts with targeted protection for exposed retained examples.

where $g_{\ell,t,m}(x)$ is the routing probability normalized over the active experts. This mixture replaces the original feed-forward output of the Transformer block. After the final modified block, the shared prediction head maps the final class-token representation $v(x)$ to $p_\theta(y \mid x) = \mathrm{softmax}(f_\omega(v(x)))$. The detailed architecture is provided in Appendix A.

Figure 1 provides an overview of the learning and unlearning phases of ModULE. ModULE operates in two stages. During learning, task supervision is combined with routing concentration, load balancing, and expert-representation differentiation to encourage sparse, non-collapsed, and differentiated expert pathways. During unlearning, routing statistics from the original model are used to select experts that provide high forget coverage with limited retain exposure. Only the selected experts are updated, while the shared backbone, routers, prediction head, and non-selected experts remain fixed. Retain supervision and output and route distillation are applied to retained examples whose pathways intersect the selected experts.

3.3 Learning Routing-Localizable Expert Representations

Design rationale. The organization learned during training determines which expert pathways are available for subsequent localized unlearning. Existing MoE unlearning methods primarily operate on pretrained models, identifying forget-relevant experts or controlling their routing and updates Zhuang et al. (2025); Zhu et al. (2026); Xie et al. (2026). ModULE additionally shapes expert organization during training. The goal is for examples targeted by a future deletion request to concentrate their routing mass on a small expert subset, while limiting how many retained examples use that subset. Updating these experts can then affect the forget set while leaving more retained pathways unchanged. Class or domain annotations specify the expected deletion structure, without identifying which group will later be deleted.

Task supervision $\mathcal{L}_{\mathrm{task}}$ optimizes predictive performance but does not explicitly enforce balanced expert utilization (Fedus et al., 2022) or group-dependent routing Chen et al. (2023b). ModULE therefore introduces three complementary structural biases. Routing concentration encourages each sample to rely primarily on a small expert subset. Because concentrated routing may collapse onto the same experts for all samples, gated-mass balancing distributes aggregate expert contribution more evenly. Finally, since balanced utilization does not prevent experts from producing similar representations, expert differentiation reduces their pairwise representation similarity. In summary, the learning objective is

$$\mathcal{L}_{\mathrm{learn}} = \mathcal{L}_{\mathrm{task}} + \lambda_{\mathrm{sp}}\mathcal{L}_{\mathrm{sp}} + \lambda_{\mathrm{bal}}\mathcal{L}_{\mathrm{bal}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}} + \lambda_{\mathrm{sep}}\mathcal{L}_{\mathrm{sep}}, \tag{2}$$

where $\mathcal{L}_{\mathrm{sp}}$, $\mathcal{L}_{\mathrm{bal}}$, $\mathcal{L}_{\mathrm{div}}$, and $\mathcal{L}_{\mathrm{sep}}$ denote the routing-concentration, gated-mass-balancing, expert-representation differentiation, and route-separation losses, respectively.

Routing concentration and gated-mass balancing. Although top-k routing limits each token to k active experts, the router may still assign similar probabilities to many experts. We encourage more concentrated routing by minimizing its entropy. However, concentration alone may cause many inputs to be routed to the same experts. We therefore also balance the average active gated mass across the training distribution as follows:

$$\mathcal{L}_{\text{sp}} = \frac{1}{L}\sum_{\ell=1}^{L}\mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{T_\ell(x)}\sum_{t=1}^{T_\ell(x)}\frac{\mathcal{H}(\pi_{\ell,t}(x))}{\log M}\right], \quad \mathcal{L}_{\text{bal}} = \frac{M}{L}\sum_{\ell=1}^{L}\sum_{m=1}^{M}\left(\bar{g}_{\ell,m}-\frac{1}{M}\right)^2, \tag{3}$$

where $\pi_{\ell,t}(x)$ is the routing distribution before top-k selection, $T_\ell(x)$ is the number of routed tokens, and $\bar{g}_{\ell,m} = \mathbb{E}_{x\sim\mathcal{D}}[g_{\ell,m}(x)]$ is the average token-aggregated active gated mass. Together, the two terms encourage individual tokens to place greater weight on fewer experts while distributing aggregate routing mass across the expert bank. They preserve the fixed top-k activation budget and allow classes and domains to share experts.

Expert-representation differentiation. Balancing controls expert usage but does not directly constrain expert outputs, allowing similarly used experts to learn similar representations. To encourage differentiation, we use centered linear CKA Kornblith et al. (2019) as follows:

$$\mathcal{L}_{\text{div}} = \frac{2}{LM(M-1)}\sum_{\ell=1}^{L}\sum_{1\le m<n\le M}\text{CKA}_c\left(H_{\ell,m}, H_{\ell,n}\right), \tag{4}$$

where $H_{\ell,m} \in \mathbb{R}^{B_p\times r}$ contains the outputs of expert m on a shared set of B_p probe samples. Evaluating experts on the same inputs provides aligned representations for comparison. Minimizing Eq. (4) discourages similar linear representation geometries across experts.

Route separation. The preceding regularizers do not explicitly distinguish routing across classes or domains. Motivating from Chen et al. (2023b), we encourage group-dependent expert usage through mutual information, as follows:

$$\mathcal{L}_{\text{sep}} = -\frac{1}{L}\sum_{\ell=1}^{L}\frac{I(Z;E_\ell)}{\min\{\log G, \log M\}}, \tag{5}$$

where Z indexes G training groups (e.g., classes or domains) and E_ℓ is an expert index distributed according to the active gated mass at layer ℓ. This objective encourages distinct group-conditioned routing patterns while learning which experts serve each group. It imposes no fixed expert assignments, and its finite weight allows sharing through the trade-off with task supervision. The grouping axis is known during learning, but no particular deletion request is required. We evaluate the resulting localization through forget coverage and retain exposure.

The four regularizers encourage concentrated token-level routing, balanced aggregate utilization, differentiated expert representations, and group-dependent expert usage, respectively. They permit shared experts while introducing a soft bias toward localization along the chosen grouping axis, without specifying a particular deletion request. Implementation details and learning-time overhead are provided in Appendix A.2, with the route-separation estimator in Appendix A.3.

3.4 Localizing a Deletion Request

Once a deletion request arrives, ModULE uses the learned routing patterns to determine which experts to update and which retained examples require protection. Given a forget set $\mathcal{D}_f$, ModULE identifies a small set of layer-expert pairs whose routed computation is strongly associated with the request but has limited overlap with retained data. All localization statistics are computed using the frozen pre-unlearning model. Let $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_L\}$, where $\mathcal{S}_\ell \subseteq \{1, \ldots, M\}$ denotes the selected experts at layer ℓ. We define the routed mass assigned to $\mathcal{S}$ for sample x as

$$q_{\mathcal{S}}(x) = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{m\in\mathcal{S}_\ell} g^0_{\ell,m}(x), \tag{6}$$

where $g^0_{\ell,m}(x)$ is the active gated weight produced by the original model. We characterize a candidate subset by its forget coverage $C_f(\mathcal{S})$ and retain exposure $E_r(\mathcal{S};\delta)$ as follows:

$$C_f(\mathcal{S}) = \mathbb{E}_{x\sim\mathcal{D}_f}\left[q_{\mathcal{S}}(x)\right], \qquad E_r(\mathcal{S};\delta) = \mathbb{E}_{x\sim\mathcal{D}_r}\left[\mathbb{I}(q_{\mathcal{S}}(x) > \delta)\right]. \tag{7}$$

Forget coverage measures how much of the forget-set computation passes through the selected experts, whereas retain exposure measures the fraction of retained samples whose pathways intersect them. Setting $\delta = 0$ captures strict pathway intersection, while $\delta > 0$ ignores negligible routed mass. ModULE selects the experts via the following function:

$$\mathcal{S}^\star \in \arg\min_{\mathcal{S}} E_r(\mathcal{S}; \delta) \quad \text{s.t.} \quad C_f(\mathcal{S}) \geq \tau_f, \qquad \sum_{\ell=1}^{L} |\mathcal{S}_\ell| \leq K_{\max}, \tag{8}$$

where τ_f specifies the required forget coverage and $K_{\max}$ limits the number of updated layer–expert pairs. The formulation accounts for expert sharing: experts strongly associated with forgotten examples may also serve retained data. If no feasible subset exists, the request is insufficiently routing-localizable under the specified constraints. The selected experts determine the retained samples most exposed to the subsequent update as

$$\mathcal{D}_r^{\text{risk}} = \{(x, y) \in \mathcal{D}_r : q_{\mathcal{S}^\star}(x) > \delta\}. \tag{9}$$

Only these at-risk samples receive targeted protection during unlearning. This concentrates protection on samples whose original pathways use the updated experts Xie et al. (2026). For $\delta = 0$, excluded examples satisfy the pathway-invariance condition of Lemma 1, provided only selected expert parameters change and computation is deterministic. For $\delta > 0$, filtering is an approximation: excluded examples may still be affected through low-mass intersections. Routing mass indicates computational association rather than exclusive knowledge ownership or causal influence. We therefore assess the functional relevance of the selected experts through expert interventions and unlearning outcomes.

3.5 Localized Expert Unlearning

ModULE uses the expert subset and exposed retained examples identified in Section 3.4 to determine where updates and preservation losses are applied. Localized expert updates and routing stabilization have precedents in MoE unlearning Zhuang et al. (2025); Zhu et al. (2026). Both incorporate preservation mechanisms during unlearning. ModULE additionally prepares expert organization during learning, then uses forget coverage and retain exposure to select the experts to update and the retained examples requiring protection. Let $\theta_{\mathcal{S}^\star} = \{\theta_{\ell,m} : m \in \mathcal{S}_\ell^\star, \ell = 1, \ldots, L\}$ denote the parameters of the experts selected in Section 3.4. During unlearning, only $\theta_{\mathcal{S}^\star}$ is updated. The shared encoder, routers, prediction head, and non-selected experts remain fixed. The localized optimization problem is

$$\theta_{\mathcal{S}^\star}^u = \arg\min_{\theta_{\mathcal{S}^\star}} \left[\mathcal{L}_{\text{forget}}(\mathcal{D}_f) + \beta \mathcal{L}_{\text{retain}}(\mathcal{D}_r^{\text{risk}}) + \gamma \mathcal{L}_{\text{out}}(\mathcal{D}_r^{\text{risk}}) + \zeta \mathcal{L}_{\text{route}}(\mathcal{D}_r^{\text{risk}})\right]. \tag{10}$$

Here, $\mathcal{L}_{\text{forget}}$ is the base forgetting objective for the deletion setting. We use the same base forgetting objective across localized and global baselines. The remaining terms preserve retained task performance, predictions, and routing, respectively.

Protecting at-risk retained examples. The learning stage aims to reduce routing overlap between deletion groups, thereby limiting the retained examples exposed to localized updates. We apply preservation losses only to $\mathcal{D}_r^{\text{risk}}$, avoiding repeated optimization over the full retain set when exposure is low. Constructing this subset still requires retained routing statistics, whose collection cost is included in the localization overhead. We first preserve task performance through:

$$\mathcal{L}_{\text{retain}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_r^{\text{risk}}} \left[\ell\left(p_u(\cdot \mid x), y\right)\right]. \tag{11}$$

We additionally preserve the original predictive behavior through output distillation:

$$\mathcal{L}_{\text{out}} = \tau^2 \mathbb{E}_{x\sim\mathcal{D}_r^{\text{risk}}} \left[D_{\text{KL}}\left(p_0^{(\tau)}(\cdot \mid x) \,\|\, p_u^{(\tau)}(\cdot \mid x)\right)\right], \tag{12}$$

where τ is the distillation temperature, and $p_0^{(\tau)}$ and $p_u^{(\tau)}$ are the temperature-scaled predictions of the original and unlearned models, respectively.

Retained-route stabilization. Freezing routers prevents direct changes to their parameters, but does not fix their assignments in a multi-layer MoE. Another speaking, updates to upstream experts can

change the representations received by downstream routers Zhuang et al. (2025); Zhu et al. (2026). We therefore preserve the original routing distributions of exposed retained examples:

$$\mathcal{L}_{\text{route}} = \frac{1}{L}\mathbb{E}_{x\sim\mathcal{D}_r^{\text{risk}}}\left[\sum_{\ell=1}^{L} D_{\text{KL}}\left(\text{sg}\left[\pi_\ell^0(x)\right] \,\|\, \pi_\ell^u(x)\right)\right]. \tag{13}$$

Here, π_ℓ^0 and π_ℓ^u are the full softmax routing distributions before top-k masking. SEUF's anchor loss targets an indicator of selected experts, whereas our loss matches each retained example's original routing distribution Zhuang et al. (2025). GRIP instead modifies router parameters through geometric constraints or post-unlearning realignment Zhu et al. (2026). With routers fixed in ModULE, gradients through $\pi_\ell^u(x)$ constrain selected upstream experts whose updates alter downstream routing.

This soft penalty discourages routing shifts without guaranteeing identical hard top-k assignments. Its contribution is identically zero at routers with no updated expert upstream, including all routers when updates are restricted to the final MoE layer.

We apply route stabilization only to retained examples. Under the assumptions of Lemma 1, a covered forget example retains its routing assignment at the earliest selected expert on its original pathway, because all preceding computation remains unchanged. It therefore continues to execute at least one selected expert, although subsequent routing and successful forgetting are not guaranteed.

Conditional pathway locality. Restricting unlearning updates to selected experts yields the following invariance property.

Lemma 1 (Conditional pathway invariance)**.** *Consider a deterministic hard top-k MoE with batch-independent expert dispatch. Suppose localized unlearning modifies only the selected experts $\mathcal{S}^\star$, while the shared components, routers, prediction head, and non-selected experts remain fixed. For any input x whose original pathway satisfies $\mathcal{E}_{\ell,k}^0(x) \cap \mathcal{S}_\ell^\star = \varnothing$, $\forall\, \ell \in \{1,\dots,L\}$, all representations and routing assignments along its executed pathway remain unchanged after unlearning. Consequently, $p_u(\cdot \mid x) = p_0(\cdot \mid x)$.*

The result follows by induction over the MoE layers and is proved in Appendix C. When $\delta = 0$ in Eq. (9), it justifies excluding retained examples outside $\mathcal{D}_r^{\text{risk}}$ from the preservation objectives. For $\delta > 0$, this exact guarantee does not cover examples with nonzero but sub-threshold exposure. Conversely, a path-disjoint forget example is also invariant and cannot be affected by localized expert updating. Average routed-mass coverage should therefore be supplemented with sample-wise forget-path coverage when the deletion request is expected to affect every example in $\mathcal{D}_f$. Detailed forgetting objectives and optimization settings are provided in Appendix B.

4 Experimental Evaluation

To evaluate the performance of our method, in this section, we present our evaluation setup and empirical results. Our experiments are conducted under three unlearning scenarios, namely, **random unlearning**, **class unlearning**, and **domain unlearning**, against multiple approximate unlearning algorithm baselines. To better understand the insights of our proposed method, we allocate an in-depth analysis subsection to discuss these aspects. Due to page limitations, details about the experiment setups are presented in Appendix D.

4.1 Experimental Settings

We evaluate ModULE on image classification tasks under three unlearning scenarios, namely random unlearning, **class unlearning**, and **domain unlearning**. Random and class unlearning are evaluated on all datasets, while domain unlearning is evaluated on datasets with explicit domain annotations. We consider two groups of datasets that stress different aspects of unlearning. **Weakly entangled settings** primarily evaluate whether unlearning preserves retained knowledge and generalization after removing the requested information. **Strongly entangled settings** primarily evaluate whether unlearning can erase target information when forget and retain samples share overlapping representations.

Dataset. For weakly entangled settings, we use PACS (Li et al., 2017) and OfficeHome (Venkateswara et al., 2017), which contain explicit domain annotations and visually distinct domain factors. For

Table 1: Class and domain unlearning on PACS. Results are macro-averaged over seven independent class-deletion requests and four independent domain-deletion requests, respectively. We report mean $\pm$ standard deviation across 3 seeds after averaging over deletion targets within each seed. All MoE-based methods share the same architecture, pre-unlearning checkpoint, expert-update budget, forgetting objective, and optimization budget. Green, blue, and yellow indicate the best, second-best, and third-best reported means among unlearning methods under the indicated metric directions, excluding MIA. Reference models are excluded from ranking; tied means share a rank.

Category	Method	Class Unlearning				Domain Unlearning			
		FA ↓	RA ↑	TA ↑	MIA ↑	FA ↓	RA ↑	TA ↑	MIA ↑
Reference models	Original model	99.42 ± 0.16	99.44 ± 0.12	92.62 ± 0.48	0.62 ± 0.014	99.46 ± 0.13	99.42 ± 0.15	89.01 ± 0.67	0.55 ± 0.012
	Retraining	5.21 ± 0.94	99.83 ± 0.06	95.74 ± 0.37	0.55 ± 0.016	29.82 ± 1.86	99.30 ± 0.18	94.27 ± 0.52	0.51 ± 0.010
Full-model unlearning	Fine-Tuning	95.08 ± 0.83	99.85 ± 0.05	94.19 ± 0.43	0.64 ± 0.013	99.50 ± 0.14	99.85 ± 0.06	89.13 ± 0.61	0.55 ± 0.011
	Gradient Ascent	9.34 ± 1.17	99.21 ± 0.19	93.58 ± 0.56	0.58 ± 0.017	20.32 ± 2.14	29.75 ± 2.68	27.39 ± 1.93	0.50 ± 0.014
	Random Labeling	32.87 ± 2.06	99.49 ± 0.13	94.43 ± 0.49	0.62 ± 0.015	10.57 ± 1.42	8.67 ± 1.76	7.64 ± 1.21	0.51 ± 0.018
	SCRUB	9.73 ± 1.08	99.59 ± 0.11	94.55 ± 0.41	0.59 ± 0.014	67.28 ± 2.35	97.16 ± 0.54	85.99 ± 0.87	0.50 ± 0.012
	SG-Unlearning	64.08 ± 2.27	99.71 ± 0.08	94.92 ± 0.38	0.65 ± 0.016	90.89 ± 1.18	99.85 ± 0.04	88.15 ± 0.73	0.55 ± 0.013
	Boundary Expanding	45.04 ± 2.48	99.59 ± 0.14	94.79 ± 0.46	0.63 ± 0.012	69.14 ± 2.71	52.69 ± 2.43	47.45 ± 1.84	0.55 ± 0.019
	Boundary Shrink	18.03 ± 1.63	99.50 ± 0.16	94.31 ± 0.54	0.62 ± 0.018	91.98 ± 1.07	97.04 ± 0.62	85.35 ± 0.96	0.53 ± 0.015
Parameter-localized unlearning	SPM	65.08 ± 2.13	99.82 ± 0.07	96.85 ± 0.32	0.50 ± 0.009	98.99 ± 0.24	99.36 ± 0.17	94.27 ± 0.58	0.53 ± 0.014
	SSD	10.17 ± 1.26	39.36 ± 2.52	39.23 ± 1.89	0.62 ± 0.020	90.41 ± 1.32	98.12 ± 0.36	89.01 ± 0.69	0.55 ± 0.016
	ℓ_1-sparse	9.27 ± 1.12	99.23 ± 0.21	93.58 ± 0.51	0.58 ± 0.015	25.29 ± 2.31	29.69 ± 2.57	27.07 ± 1.78	0.50 ± 0.013
	SalUn	46.05 ± 2.19	99.50 ± 0.12	94.31 ± 0.47	0.63 ± 0.014	43.67 ± 2.04	87.62 ± 1.16	74.68 ± 1.27	0.54 ± 0.017
MoE-based unlearning	Vanilla MoE	45.62 ± 2.36	99.23 ± 0.18	94.79 ± 0.44	0.57 ± 0.016	30.79 ± 1.93	95.12 ± 0.71	85.13 ± 0.92	0.57 ± 0.019
	GRIP	9.33 ± 1.04	99.59 ± 0.10	94.67 ± 0.39	0.53 ± 0.012	23.38 ± 1.75	94.88 ± 0.83	82.96 ± 1.08	0.51 ± 0.014
	SEUF	10.07 ± 1.31	99.68 ± 0.09	95.32 ± 0.42	0.62 ± 0.017	64.08 ± 2.22	99.71 ± 0.08	85.52 ± 0.81	0.49 ± 0.011
	RepSelect	12.89 ± 1.48	92.03 ± 0.94	85.47 ± 1.03	0.55 ± 0.015	66.55 ± 2.46	95.39 ± 0.68	83.92 ± 0.98	0.50 ± 0.013
	ModULE	8.76 ± 1.15	99.64 ± 0.12	95.40 ± 0.45	0.62 ± 0.016	22.15 ± 1.98	97.51 ± 0.49	89.65 ± 0.76	0.49 ± 0.012

strongly entangled settings, we use CIFAR-100 (Krizhevsky et al., 2009) and Tiny-ImageNet (Le et al., 2015), which contain many semantically and visually related categories. These datasets allow us to evaluate ModULE under both generalization-preserving and high-overlap forgetting regimes.

Baselines, Models and Metrics. We compare ModULE with conventional and structured unlearning methods, including the MoE-specific baselines SEUF (Zhuang et al., 2025) and GRIP (Zhu et al., 2026). The original model and retraining serve as references. Unless otherwise specified, all methods use the same DeiT-Small (Touvron et al., 2021) backbone and follow the same pretraining, fine-tuning, and unlearning protocol. We report four standard unlearning metrics: **Forget Accuracy (FA)**, **Retain Accuracy (RA)**, **Test Accuracy (TA)**, and **Membership Inference Attack (MIA)**. To ensure fair comparison, we evaluate all methods at matched forgetting strength by stopping unlearning when FA reaches a predefined target or when the maximum unlearning budget is reached. We additionally report computational and localization diagnostics, including updated-parameter ratio, forget-expert routing mass, and at-risk retain ratio. Full experimental details, hyperparameters, baseline implementations, and extended results are provided in Appendix D.

4.2 Main Results

Comparisons with baselines. Table 1 reports PACS results. For class unlearning, ModULE achieves the lowest mean FA (8.76%), closest to retraining, while maintaining 99.64% RA and 95.40% TA. For domain unlearning, it achieves the lowest mean FA (i.e., 22.15%) and highest mean TA (i.e., 89.65%) among MoE-based methods, although SEUF attains higher RA. Gradient Ascent and Random Labeling yield lower domain FA but severely degrade retained utility. ModULE still falls below retraining in domain TA (89.65% vs. 94.27%); its FA below the retraining reference does not alone establish more faithful unlearning. The detailed results on OfficeHome, CIFAR-100, and TinyImageNet are demonstrated in Appendix H.

Sequential unlearning. We evaluate repeated deletion requests without resetting the model to its original checkpoint. At each step, all methods start from the checkpoint produced by the preceding request; ModULE additionally recomputes its selected experts and at-risk retain set. Figures 2 and 3 show the results for class and domain deletion, respectively. For class deletion, ModULE achieves the lowest FA over the first four requests and remains competitive after the fifth, while preserving the highest final RA and TA. For domain deletion, it achieves the lowest FA at every step, together with near-best RA and the highest TA as requests accumulate. Overall, these results indicate a stronger

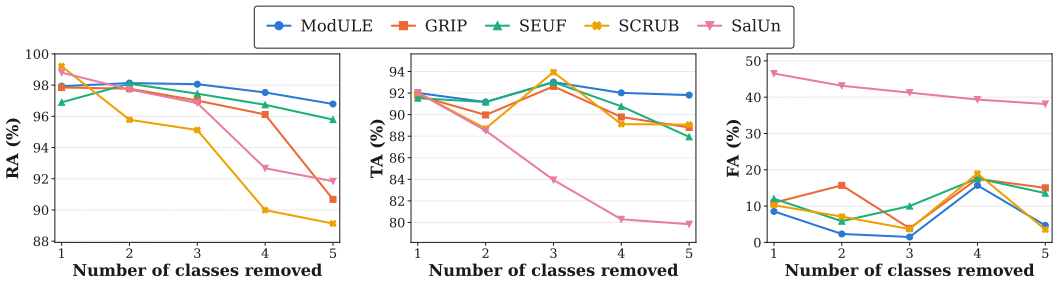


Figure 2: Sequential class-level unlearning on PACS. From left to right, the panels report retain accuracy (RA), test accuracy (TA), and forget accuracy (FA). Each point represents the model after the indicated cumulative number of class-deletion requests, without resetting to the original checkpoint. Higher RA and TA and lower FA are better.

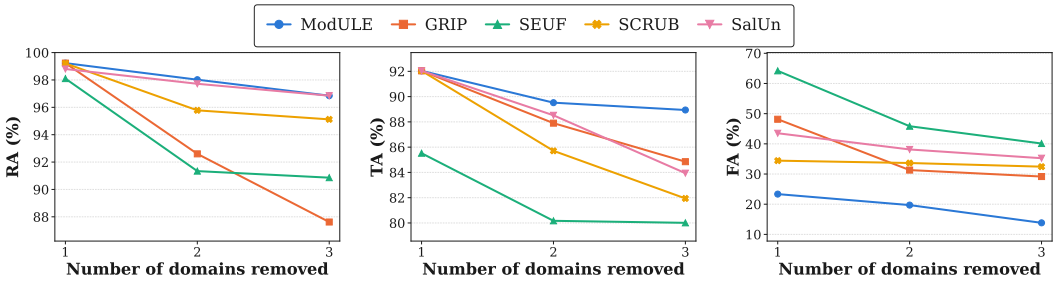


Figure 3: Sequential domain-level unlearning on PACS. From left to right, the panels report retain accuracy (RA), test accuracy (TA), and forget accuracy (FA). Each point represents the model after the indicated cumulative number of domain-deletion requests, without resetting to the original checkpoint. Higher RA and TA and lower FA are better.

sequential forgetting–utility trade-off and are consistent with localized updates limiting cumulative damage to retained performance.

4.3 Ablation and In-Depth Analysis

We defer detailed ablations and diagnostics to the appendix. We study (1) expert-selection rules (Appendix E.1), (2) unlearning update scope (Appendix E.1.1), (3) top-k routing and the expert-update budget (Appendix E.1.2), (4) the number of experts (Appendix E.1.3), (5) architecture controls under matched parameter budgets (Appendix E.1.4), and (6) the number of forgotten classes (Appendix E.1.5).

We further analyze routing-mass distributions (Appendix F.1), routing stability (Appendix F.2), and modularization-loss sensitivity (Appendix F.3). Computational costs are reported in Appendix G, and complete results on OfficeHome, CIFAR-100, and TinyImageNet in Appendix H.

5 Conclusion

We presented ModULE, a framework that organizes MoE computation during learning to support localized unlearning of future structured deletion requests. Its learning objectives encourage concentrated, balanced, differentiated, and group-dependent expert usage while permitting expert sharing. Given a deletion request, ModULE selects a bounded set of experts using forget coverage and retain exposure, then updates the selected experts with targeted protection for exposed retained examples and layer-wise route stabilization. Experiments on PACS demonstrate competitive forgetting–utility trade-offs under class and domain unlearning. These results motivate treating expert organization during learning as a design consideration for subsequent unlearning.

REFERENCES

Somnath Basu Roy Chowdhury, Krzysztof Choromanski, Arijit Sehanobish, Kumar Dubey, and Snigdha Chaturvedi. Towards scalable exact machine unlearning using parameter-efficient fine-tuning. In *International Conference on Learning Representations*, volume 2025, pp. 84753–84779, 2025.

Martin Bertran, Shuai Tang, Michael Kearns, Jamie Morgenstern, Aaron Roth, and Zhiwei S Wu. Reconstruction attacks on machine unlearning: Simple models are vulnerable. *Advances in Neural Information Processing Systems*, 37:104995–105016, 2024.

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

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

Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7766–7775, 2023a.

Tianqi Chen, Shujian Zhang, and Mingyuan Zhou. Score forgetting distillation: A swift, data-free method for machine unlearning in diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 18980–19004, 2025.

Zitian Chen, Yikang Shen, Mingyu Ding, Zhenfang Chen, Hengshuang Zhao, Erik G Learned-Miller, and Chuang Gan. Mod-squad: Designing mixtures of experts as modular multi-task learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11828–11837, 2023b.

Eli Chien, Haoyu Wang, Ziang Chen, and Pan Li. Certified machine unlearning via noisy stochastic gradient descent. *Advances in Neural Information Processing Systems*, 37:38852–38887, 2024.

Dasol Choi and Dongbin Na. Distribution-level feature distancing for machine unlearning: Towards a better trade-off between model utility and forgetting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 2536–2544, 2025.

Rishav Chourasia and Neil Shah. Forget unlearning: Towards true data-deletion in machine learning. In *International conference on machine learning*, pp. 6028–6073. PMLR, 2023.

Arnav Devalapally, Poornima Jain, Kartik Srinivas, and Vineeth N. Balasubramanian. $\oslash$ source models leak what they shouldn't $\nrightarrow$: Unlearning zero-shot transfer in domain adaptation through adversarial optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1543–1553, June 2026.

Zonglin Di, Sixie Yu, Yevgeniy Vorobeychik, and Yang Liu. Adversarial machine unlearning. In *International Conference on Learning Representations*, volume 2025, pp. 21612–21633, 2025.

Junhao Dong, Hao Zhu, Yifei Zhang, Xinghua Qu, Yew Soon Ong, and Piotr Koniusz. Machine unlearning via task simplex arithmetic. *Advances in Neural Information Processing Systems*, 38: 166676–166707, 2026.

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

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

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

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

Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11516–11524, 2021.

An Huang, Zhipeng Cai, and Zuobin Xiong. A survey of machine unlearning in generative ai models: Methods, applications, security, and challenges. *IEEE Internet of Things Journal*, 2025.

Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems*, 36:51584–51605, 2023.

Yasser Khalil, Mehdi Setayesh, and Hongliang Li. Coun: Empowering machine unlearning via contrastive learning. *Advances in Neural Information Processing Systems*, 38:131254–131283, 2026.

Yasser H Khalil, Leo Brunswic, Soufiane Lamghari, Xu Li, Mahdi Beitollahi, and Xi Chen. Not: Federated unlearning via weight negation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25759–25769. IEEE, 2025.

Junyaup Kim and Simon S Woo. Efficient two-stage model retraining for machine unlearning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4361–4369, 2022.

Sangamesh Kodge, Gobinda Saha, and Kaushik Roy. Deep unlearning: Fast and efficient gradient-free class forgetting. *Transactions on Machine Learning Research*, 2024.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMlR, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987, 2023.

Sonia Laguna, Jorge da Silva Gonçalves, Moritz Vandenhirtz, Alain Ryser, Irene Cannistraci, and Julia E Vogt. Rethinking machine unlearning: Models designed to forget via key deletion. In *New Frontiers in Associative Memories - Workshop at ICLR 2026*, 2026. URL https://openreview.net/forum?id=PZ0ERnmVUu.

Anjie Le, Can Peng, Yuyuan Liu, and J. Alison Noble. Pour: A provably optimal method for unlearning representation via neural collapse. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10273–10282, June 2026.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Justin Lee, Zheda Mai, Jinsu Yoo, Chongyu Fan, Cheng Zhang, and Wei-Lun Chao. Continual unlearning for text-to-image diffusion models: A regularization perspective. *International Conference on Learning Representations*, 2026.

Bo Li, Yifei Shen, Jingkang Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. Sparse mixture-of-experts are domain generalizable learners. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=RecZ9nB9Q4.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Na Li, Chunyi Zhou, Yansong Gao, Hui Chen, Zhi Zhang, Boyu Kuang, and Anmin Fu. Machine unlearning: Taxonomy, metrics, applications, challenges, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Shen Lin, Xiaoyu Zhang, Chenyang Chen, Xiaofeng Chen, and Willy Susilo. Erm-ktp: Knowledge-level machine unlearning via knowledge transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20147–20155, 2023.

Yi Liu, Lei Xu, Xingliang Yuan, Cong Wang, and Bo Li. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE conference on computer communications*, pp. 1749–1758. IEEE, 2022.

Jakub Łucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramèr, and Javier Rando. An adversarial perspective on machine unlearning for AI safety. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Neil G Marchant, Benjamin IP Rubinstein, and Scott Alfeld. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7691–7700, 2022.

Minh-Duong Nguyen, Senura Hansaja, Le-Tuan Nguyen, Quoc-Viet Pham, Ken-Tye Yong, Nguyen H. Tran, and Dung D. Le. Computation and communication efficient federated unlearning via on-server gradient conflict mitigation and expression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2026a.

Thanh Linh Nguyen, Marcela Tuler de Oliveira, An Braeken, Aaron Yi Ding, and Quoc-Viet Pham. Towards verifiable federated unlearning: Framework, challenges, and the road ahead. *IEEE Internet Computing*, 2026b.

Gaurav Patel and Qiang Qiu. Learning to unlearn while retaining: Combating gradient conflicts in machine unlearning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4211–4221, 2025.

Martin Pawelczyk, Jimmy Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. In *International Conference on Learning Representations*, volume 2025, pp. 50542–50573, 2025.

Laiqiao Qin, Tianqing Zhu, Linlin Wang, and Wanlei Zhou. Machine unlearning on pre-trained models by residual feature alignment using lora. *IEEE Transactions on Dependable and Secure Computing*, 2026.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in neural information processing systems*, 34:8583–8595, 2021.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=B1ckMDqlg`.

Filip Sondej, Yushi Yang, and Adam Mahdi. Repselect: Robust LLM unlearning via representational selectivity. In *Continual Adaptation at Scale: Towards Sustainable AI Workshop*, 2026. URL `https://openreview.net/forum?id=4oNRpt9T8s`.

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

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Stefan Vasilev, Christian Herold, Baohao Liao, Seyyed Hadi Hashemi, Shahram Khadivi, and Christof Monz. Unilogit: Robust machine unlearning for llms using uniform-target self-distillation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 22453–22472, 2025.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

Shixuan Wang, Jingwen Ye, and Xinchao Wang. Machine unlearning in 3d generation: A perspective-coherent acceleration framework. *Advances in Neural Information Processing Systems*, 38: 137831–137862, 2026.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning*, pp. 10355–10366. PMLR, 2020.

Yaxin Xiao, Qingqing Ye, Li Hu, Huadi Zheng, Haibo Hu, Zi Liang, Haoyang Li, and Yijie Jiao. Reminiscence attack on residuals: Exploiting approximate machine unlearning for privacy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3058–3068, 2025.

Jingyi Xie, Yijun Lin, Yinjiang Xiong, Zhikun Zhang, and Sai Li. Routing-aware expert calibration for machine unlearning in mixture-of-experts language models. *arXiv preprint arXiv:2606.10338*, 2026.

Haonan Yan, Xiaoguang Li, Ziyao Guo, Hui Li, Fenghua Li, and Xiaodong Lin. Arcane: An efficient architecture for exact machine unlearning. In *Ijcai*, volume 6, pp. 19, 2022.

Saleh Zare Zade, Xiangyu Zhou, Sijia Liu, and Dongxiao Zhu. Attention smoothing is all you need for unlearning. *International Conference on Learning Representations*, 2026.

Zijie Zhang, Yang Zhou, Xin Zhao, Tianshi Che, and Lingjuan Lyu. Prompt certified machine unlearning with randomized gradient smoothing and quantization. *Advances in Neural Information Processing Systems*, 35:13433–13455, 2022.

Amber Yijia Zheng, Yu-Shan Tai, and Raymond A. Yeh. Designing to forget: Deep semi-parametric models for unlearning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17409–17419, June 2026a.

Amber Yijia Zheng, Yu-Shan Tai, and Raymond A. Yeh. Designing to forget: Deep semi-parametric models for unlearning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17409–17419, June 2026b.

Xuyang Zhong, Haochen Luo, and Chen Liu. Dualoptim: Enhancing efficacy and stability in machine unlearning with dual optimizers. *Advances in Neural Information Processing Systems*, 38: 85464–85496, 2026.

Shiji Zhou, Tianbai Yu, Zhi Zhang, Heng Chang, Xiao Zhou, Dong Wu, and Han Zhao. Efficient utility-preserving machine unlearning with implicit gradient surgery. *Advances in Neural Information Processing Systems*, 38:4531–4569, 2026.

Andy Zhu, Rongzhe Wei, Yupu Gu, and Pan Li. GRIP: Algorithm-agnostic machine unlearning for mixture-of-experts via geometric router constraints. In *Third Conference on Language Modeling*, 2026. URL `https://openreview.net/forum?id=voKCuTimaW`.

Haomin Zhuang, Yihua Zhang, Kehan Guo, Jinghan Jia, Gaowen Liu, Sijia Liu, and Xiangliang Zhang. Seuf: Is unlearning one expert enough for mixture-of-experts llms? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8664–8678, 2025.

AI USE STATEMENT

AI tools were used only to assist with language editing and improve the clarity of the manuscript. All scientific content, analyses, results, and conclusions were produced and verified by the authors.

ETHICS STATEMENT

This work does not involve new human-subject data collection or deployment decisions. We use existing research datasets and follow their established research-use protocols.

REPRODUCIBILITY STATEMENT

We have made several efforts to ensure the reproducibility of our results. The ModULE architecture, including MoE placement, token-level routing, gate normalization, sample-level routing statistics, and the set of parameters updated during unlearning, is described in Section 3.2 and Appendix A. The estimators for the four learning-time regularizers (routing entropy, gated-mass balancing, centered CKA with its probe construction, and group–expert route separation), together with their gradient flow and learning-time overhead, are given in Appendices A.2 and A.3. The deletion-localization procedure (forget coverage, retain exposure, expert selection, and construction of the at-risk retain set) is defined in Section 3.4. The localized unlearning objective, the type-specific forgetting objectives, the mini-batch estimators, the treatment of frozen routers, and the optimization protocol are described in Section 3.5 and Appendix B. For our theoretical result, the assumptions of Lemma 1 (deterministic hard top-k routing, batch-independent expert dispatch, and frozen shared components) are stated in Section 3.5, and a complete proof, including the conditions under which the guarantee does not apply, is given in Appendix C. The datasets and deletion scenarios are described in Appendices D.1 and D.2, and the baselines and their hyperparameters in Appendix D.3 and Table 2. Training and unlearning configurations for ModULE and the retraining reference are given in Appendix D.4, and the default configuration used for all ablations is listed at the beginning of Appendix E. Evaluation, efficiency, localization, and routing-stability metrics are defined in Appendices D.5–D.8. Results in Table 1 are reported as mean $\pm$ standard deviation over three random seeds, macro-averaged over deletion targets within each seed. The computational cost of unlearning is reported in Appendix G, and results on OfficeHome, CIFAR-100, and Tiny-ImageNet are given in Appendix H. Source code, configuration files, data-split indices, and scripts to reproduce the reported tables and figures are included in the anonymous supplementary material.

A DETAILED IMPLEMENTATIONS

A.1 ARCHITECTURE AND FORWARD COMPUTATION

MoE placement. ModULE uses a pretrained Transformer backbone and replaces the feed-forward networks in its final two blocks with MoE sublayers. We index these modified blocks by $\ell \in \{1, \dots, L\}$, where $L = 2$. The attention and normalization components remain shared across all inputs. Let T denote the number of tokens, including the class token, and let

$$X_0(x) = b_\phi(x) \in \mathbb{R}^{T \times d} \tag{14}$$

denote the representations produced by the embedding layer and the Transformer blocks preceding the two MoE-equipped blocks.

Shared attention sublayer. Each modified block follows the pre-normalized Transformer architecture. Its multi-head self-attention sublayer (MHA) computes

$$A_\ell(x) = X_{\ell-1}(x) + \mathrm{MHA}_\ell\left(\mathrm{LN}_\ell^{\mathrm{attn}}(X_{\ell-1}(x))\right). \tag{15}$$

The representations entering the router and experts are

$$V_\ell(x) = \mathrm{LN}_\ell^{\mathrm{moe}}(A_\ell(x)). \tag{16}$$

Token-level routing. The router operates independently on every token:

$$\pi_{\ell,t}(x) = \mathrm{softmax}\left(r_{\theta_\ell}(V_{\ell,t}(x))\right) \in \mathbb{R}^M. \tag{17}$$

The active expert set is

$$\mathcal{E}_{\ell,t,k}(x) = \mathrm{TopK}\left(\pi_{\ell,t}(x), k\right), \tag{18}$$

and the normalized active gate is

$$g_{\ell,t,m}(x) = \mathbf{1}\left[m \in \mathcal{E}_{\ell,t,k}(x)\right] \frac{\pi_{\ell,t,m}(x)}{\sum_{j\in\mathcal{E}_{\ell,t,k}(x)} \pi_{\ell,t,j}(x)}. \tag{19}$$

Thus, $g_{\ell,t,m}(x) = 0$ for inactive experts and $\sum_{m=1}^{M} g_{\ell,t,m}(x) = 1$.

Expert computation. Each MoE layer contains M independently parameterized expert feed-forward networks. Expert m at layer ℓ is

$$h_{\ell,m}(v;\psi_{\ell,m}) = W^{(2)}_{\ell,m}\sigma\left(W^{(1)}_{\ell,m}v + b^{(1)}_{\ell,m}\right) + b^{(2)}_{\ell,m}, \tag{20}$$

where $\psi_{\ell,m}$ contains the expert parameters and σ is the activation function. Experts with the same index at different layers have separate parameters. The MoE output for token t is

$$Z_{\ell,t}(x) = \sum_{m=1}^{M} g_{\ell,t,m}(x) h_{\ell,m}\left(V_{\ell,t}(x);\psi_{\ell,m}\right). \tag{21}$$

This output replaces the original feed-forward output and is incorporated through the residual connection

$$X_\ell(x) = A_\ell(x) + Z_\ell(x). \tag{22}$$

The resulting token representations are passed to the next modified Transformer block.

Prediction head. After the second MoE-equipped block, the model extracts the normalized class-token representation

$$v(x) = \left[\mathrm{LN}_{\mathrm{out}}(X_L(x))\right]_{\mathrm{cls}}. \tag{23}$$

The shared classifier produces

$$p_\theta(y \mid x) = \mathrm{softmax}\left(f_\omega(v(x))\right). \tag{24}$$

Sample-level routing statistics. Although routing is performed per token, deletion requests and retained examples are defined at the sample level. We therefore aggregate the active gated mass across tokens:

$$g_{\ell,m}(x) = \frac{1}{T}\sum_{t=1}^{T} g_{\ell,t,m}(x). \tag{25}$$

These sample-level statistics are used to compute forget coverage, retain exposure, and the at-risk retain set.

Parameters updated during unlearning. For a selected expert collection $\mathcal{S}^\star = \{\mathcal{S}^\star_\ell\}_{\ell=1}^{L}$, localized unlearning updates only

$$\{\psi_{\ell,m} : m \in \mathcal{S}^\star_\ell\}. \tag{26}$$

The preceding backbone, shared attention and normalization components, routers, prediction head, and non-selected experts remain fixed. During deterministic unlearning and evaluation, routing noise and dropout are disabled. No batch-dependent expert dropping is used, ensuring that executed pathways are determined solely by the token-level top-k routing decisions.

A.2 Learning-Objective Implementation Details

Active routing weights. Let $T_\ell(x)$ denote the number of routing units at MoE layer ℓ. For sample-level routing, $T_\ell(x) = 1$; for token-level routing, each token constitutes one routing unit. The router produces

$$\pi_{\ell,t}(x) = \mathrm{softmax}\left(r_{\theta_\ell}(u_{\ell-1,t}(x))\right), \tag{27}$$

and selects

$$\mathcal{E}_{\ell,t,k}(x) = \mathrm{TopK}\left(\pi_{\ell,t}(x), k\right). \tag{28}$$

The normalized active gate is

$$g_{\ell,t,m}(x) = \mathbf{1}\left[m \in \mathcal{E}_{\ell,t,k}(x)\right] \frac{\pi_{\ell,t,m}(x)}{\sum_{j\in\mathcal{E}_{\ell,t,k}(x)} \pi_{\ell,t,j}(x)}. \tag{29}$$

Consequently, $g_{\ell,t,m}(x) = 0$ for inactive experts and $\sum_{m=1}^{M} g_{\ell,t,m}(x) = 1$. For the sample-level routing statistics used during deletion localization, we average over routing units:

$$g_{\ell,m}(x) = \frac{1}{T_\ell(x)} \sum_{t=1}^{T_\ell(x)} g_{\ell,t,m}(x). \tag{30}$$

Routing-entropy estimator. For a minibatch $\mathcal{B}$, let $N_\ell(\mathcal{B}) = \sum_{x\in\mathcal{B}} T_\ell(x)$ denote the number of routing units at layer ℓ. We estimate the routing-concentration objective as

$$\widehat{\mathcal{L}}_{\mathrm{sp}} = -\frac{1}{L} \sum_{\ell=1}^{L} \frac{1}{N_\ell(\mathcal{B}) \log M} \sum_{x\in\mathcal{B}} \sum_{t=1}^{T_\ell(x)} \sum_{m=1}^{M} \pi_{\ell,t,m}(x) \log(\pi_{\ell,t,m}(x) + \varepsilon). \tag{31}$$

The normalization by $\log M$ places the entropy in $[0, 1]$. Top-k routing provides exact computational sparsity, while Eq. (31) controls how concentrated the routing probabilities are.

Load-balancing estimator. We compute the mean active gated mass assigned to expert m as

$$\widehat{\bar{g}}^{\mathcal{B}}_{\ell,m} = \frac{1}{N_\ell(\mathcal{B})} \sum_{x\in\mathcal{B}} \sum_{t=1}^{T_\ell(x)} g_{\ell,t,m}(x). \tag{32}$$

The minibatch balancing loss is then

$$\widehat{\mathcal{L}}_{\mathrm{bal}} = \frac{M}{L} \sum_{\ell=1}^{L} \sum_{m=1}^{M} \left(\widehat{\bar{g}}^{\mathcal{B}}_{\ell,m} - \frac{1}{M}\right)^2. \tag{33}$$

This loss balances the routed contribution rather than the full pre-top-k probabilities. It is a soft regularizer and does not require every expert to process exactly the same number of routing units. For routing diagnostics, we additionally record the normalized selection frequency

$$\widehat{f}^{\mathcal{B}}_{\ell,m} = \frac{1}{kN_\ell(\mathcal{B})} \sum_{x\in\mathcal{B}} \sum_{t=1}^{T_\ell(x)} \mathbf{1}\left[m \in \mathcal{E}_{\ell,t,k}(x)\right]. \tag{34}$$

The gated mass in Eq. (32) measures expert contribution, whereas Eq. (34) measures how frequently each expert is selected. We use the former in the learning objective and the latter only for evaluation.

Centered CKA computation. A valid cross-expert comparison requires every expert to process the same aligned inputs. At each MoE layer, we sample B_p routing units from the current minibatch and collect their representations into

$$U^p_\ell \in \mathbb{R}^{B_p \times d}. \tag{35}$$

For every expert, we compute

$$H_{\ell,m} = h_{\ell,m}\left(\mathrm{sg}(U^p_\ell)\right) \in \mathbb{R}^{B_p \times r}, \tag{36}$$

where sg denotes stop-gradient. Thus, the differentiation loss updates the experts without propagating gradients into their shared input representations. Let

$$J_{B_p} = I_{B_p} - \frac{1}{B_p}\mathbf{1}\mathbf{1}^\top, \qquad \widetilde{H}_{\ell,m} = J_{B_p} H_{\ell,m}. \tag{37}$$

The centered linear CKA similarity is

$$\mathrm{CKA}_c\left(H_{\ell,m}, H_{\ell,n}\right) = \frac{\left\|\widetilde{H}_{\ell,m}^{\top}\widetilde{H}_{\ell,n}\right\|_F^2}{\left\|\widetilde{H}_{\ell,m}^{\top}\widetilde{H}_{\ell,m}\right\|_F \left\|\widetilde{H}_{\ell,n}^{\top}\widetilde{H}_{\ell,n}\right\|_F + \varepsilon}. \tag{38}$$

We average over unordered expert pairs:

$$\widehat{\mathcal{L}}_{\mathrm{div}} = \frac{2}{LM(M-1)} \sum_{\ell=1}^{L} \sum_{1 \le m < n \le M} \mathrm{CKA}_c\left(H_{\ell,m}, H_{\ell,n}\right). \tag{39}$$

Centering removes batchwise mean effects, while CKA normalization reduces sensitivity to representation scale. Linear CKA also detects similarity under orthogonal transformations, preventing two experts from appearing different merely because they express similar representations in rotated coordinate systems. Nevertheless, low CKA does not establish statistically independent or mutually exclusive expert knowledge.

Gradient flow. The task loss updates the components participating in the routed forward pass, including the active experts. The routing entropy and balancing losses provide gradients to the router through $\pi_{\ell,t,m}$ and $g_{\ell,t,m}$. The CKA objective updates all experts evaluated on the probe inputs. Because U_ℓ^p is detached, the CKA objective does not update the shared backbone through the probe computation.

Learning-time overhead. The task, sparsity, and balancing losses are computed from the standard sparse forward pass. The additional cost arises from evaluating all M experts on the B_p probe routing units. Let $N_\ell(\mathcal{B})$ be the number of regular routing units at layer ℓ. If already computed active-expert outputs are reused, the approximate additional expert computation is

$$\frac{\Delta C_\ell}{C_\ell^{\mathrm{routed}}} \approx \frac{B_p(M-k)}{N_\ell(\mathcal{B})k}. \tag{40}$$

Choosing $B_p \ll N_\ell(\mathcal{B})$ limits this overhead. The probe size, regularization coefficients, numerical constant ε, and measured wall-clock overhead are reported in the experimental settings.

A.3 Group–Expert Route Separation

Let $\mathcal{D}_z$ contain the training examples in group z, with $P(z) = |\mathcal{D}_z|/|\mathcal{D}|$. Using the token-averaged active gated mass, we define

$$P_\ell(m \mid z) = \mathbb{E}_{x \sim \mathcal{D}_z}[g_{\ell,m}(x)], \qquad P_\ell(m) = \sum_z P(z) P_\ell(m \mid z). \tag{41}$$

These distributions induce

$$I(Z; E_\ell) = \sum_{z=1}^{G} \sum_{m=1}^{M} P(z) P_\ell(m \mid z) \log \frac{P_\ell(m \mid z)}{P_\ell(m)}, \tag{42}$$

with zero-probability contributions defined by continuity. For $G, M > 1$, mutual information is bounded above by $\min\{\log G, \log M\}$, giving the normalization in Eq. (5).

The separation term complements the other regularizers: it encourages group-dependent usage, whereas balancing controls aggregate utilization, routing concentration controls individual routing distributions, and CKA compares expert representations. High mutual information does not alone guarantee compact group-level support or disjoint hard routes; these properties are assessed through localization diagnostics.

B Localized Expert Unlearning Details

Frozen reference model and localization statistics. Before unlearning, we retain a frozen copy of the original model, denoted by Θ^0. This model provides the teacher predictions $p_0^{(\tau)}(\cdot \mid x)$ and

routing distributions $\{\pi_\ell^0(x)\}_{\ell=1}^L$. The selected expert collection $\mathcal{S}^\star$ and the at-risk retain set $\mathcal{D}_r^{\text{risk}}$ are computed once using Θ^0 and remain fixed throughout unlearning. Recomputing either set during optimization would create a moving localization target.

Unless otherwise specified, we use strict pathway exposure:

$$\mathcal{D}_r^{\text{risk}} = \{(x, y) \in \mathcal{D}_r : q_{\mathcal{S}^\star}(x) > 0\}. \tag{43}$$

Routing noise, dropout, and batch-dependent expert dropping are disabled during localization and unlearning.

Default forgetting objective. The default implementation uses gradient-ascent forgetting:

$$\mathcal{L}_{\text{forget}}^{\text{GA}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_f}\left[\ell\left(p_u(\cdot \mid x), y\right)\right]. \tag{44}$$

Minimizing Eq. (44) increases the original task loss on the forget set. Because this objective is not bounded below, we optimize it for a fixed number of unlearning steps rather than to convergence. The retained-task and distillation terms constrain the resulting parameter drift.

ModULE is not specific to gradient-ascent forgetting. To isolate the effect of parameter localization, every comparison between localized and global updating uses the same forgetting objective, optimization budget, and preservation terms. The two variants differ only in which parameters are trainable.

Type-specific alternatives. Different deletion requests may require different behavioral targets. For class deletion with forgotten label set $\mathcal{Y}_f \subseteq \{1, \ldots, C\}$, a bounded alternative is

$$\mathcal{L}_{\text{forget}}^{\text{class}} = \mathbb{E}_{x\sim\mathcal{D}_f}\left[D_{\text{KL}}\left(q_{\neg\mathcal{Y}_f} \,\|\, p_u(\cdot \mid x)\right)\right], \tag{45}$$

where

$$q_{\neg\mathcal{Y}_f}(c) = \frac{\mathbb{I}[c \notin \mathcal{Y}_f]}{C - |\mathcal{Y}_f|}. \tag{46}$$

This objective suppresses forgotten labels by distributing probability mass over the remaining classes.

For sample deletion, we additionally consider confidence flattening:

$$\mathcal{L}_{\text{forget}}^{\text{sample}} = \mathbb{E}_{x\sim\mathcal{D}_f}\left[D_{\text{KL}}\left(U_C \,\|\, p_u(\cdot \mid x)\right)\right], \tag{47}$$

where U_C is the uniform distribution over the C task labels. Confidence flattening is a behavioral forgetting surrogate and does not by itself establish equivalence to retain-only retraining.

For domain deletion, we use Eq. (44) when the objective is to suppress task-predictive behavior on the forgotten domain. If the objective is instead to remove domain information while preserving class semantics, a fixed domain probe can be used:

$$\mathcal{L}_{\text{forget}}^{\text{domain}} = \mathbb{E}_{x\sim\mathcal{D}_f}\left[D_{\text{KL}}\left(U_D \,\|\, p_{\text{dom}}(\cdot \mid z_u(x))\right)\right], \tag{48}$$

where U_D is uniform over the training domains and p_{dom} is a frozen domain probe. These objectives represent different deletion semantics and are evaluated separately rather than combined.

Mini-batch estimation. At each unlearning step, we independently sample $\mathcal{B}_f \subseteq \mathcal{D}_f$ and $\mathcal{B}_r \subseteq \mathcal{D}_r^{\text{risk}}$. Each objective term is averaged over its corresponding mini-batch:

$$\widehat{\mathcal{L}}_{\text{retain}} = \frac{1}{|\mathcal{B}_r|}\sum_{(x,y)\in\mathcal{B}_r} \ell\left(p_u(\cdot \mid x), y\right), \tag{49}$$

$$\widehat{\mathcal{L}}_{\text{out}} = \frac{\tau^2}{|\mathcal{B}_r|}\sum_{x\in\mathcal{B}_r} D_{\text{KL}}\left(p_0^{(\tau)}(\cdot \mid x) \,\|\, p_u^{(\tau)}(\cdot \mid x)\right), \tag{50}$$

$$\widehat{\mathcal{L}}_{\text{route}} = \frac{1}{L|\mathcal{B}_r|}\sum_{x\in\mathcal{B}_r}\sum_{\ell=1}^{L} D_{\text{KL}}\left(\text{sg}[\pi_\ell^0(x)] \,\|\, \pi_\ell^u(x)\right). \tag{51}$$

If $\mathcal{D}_r^{\text{risk}}$ is empty, all three preservation terms are defined as zero.

Both output and route distillation use the original model as a stop-gradient teacher. The routing loss is computed over the complete pre-top-k softmax distribution rather than only the selected expert

indices. This provides a differentiable stability signal, although it does not guarantee unchanged hard top-k assignments.

Gradient flow through frozen routers. Only the parameters in $\theta_{\mathcal{S}^\star}$ are included in the optimizer. All other parameters have gradient updates disabled. However, the current-model router computations are retained in the computation graph. Consequently, the gradient of $\widehat{\mathcal{L}}_{\text{route}}$ can pass through a frozen downstream router to selected experts at preceding layers. The frozen router weights receive no update.

The teacher distributions $\pi^0_\ell(x)$ are detached, whereas $\pi^u_\ell(x)$ must not be detached. Router distributions at layers whose inputs are unaffected by the selected experts remain identical and contribute zero to the routing loss.

No routing loss is applied to $\mathcal{B}_f$. For a covered forget example, let

$$j(x) = \min\left\{\ell : \mathcal{E}^0_{\ell,k}(x) \cap \mathcal{S}^\star_\ell \neq \varnothing\right\} \tag{52}$$

denote its earliest selected-expert intersection. Since no selected expert is executed before layer $j(x)$, all preceding representations and the input to router $j(x)$ remain unchanged. Its assignment at layer $j(x)$ therefore remains fixed, ensuring that at least one selected expert stays active while allowing subsequent routes to adapt during forgetting.

Optimization protocol. Unless otherwise specified, unlearning is performed for five epochs with mini-batches of size 128 using AdamW and weight decay 0.05. The unlearning learning rate is selected from

$$\left\{a \times 10^{-b} : a \in \{1,3,5,7\},\ b \in \{2,3,4\}\right\}. \tag{53}$$

We use $\beta = \gamma = \zeta = 0.25$ in the default configuration. The distillation temperature τ and any dataset-specific deviations are reported with the corresponding experimental results. Localized and global variants use the same optimizer, learning rate, number of epochs, mini-batch construction, and objective coefficients.

A deletion request is considered unsuitable for purely localized updating when some targeted forget examples have no pathway intersection with $\mathcal{S}^\star$. Such examples are functionally independent of the trainable parameters under the assumptions of Lemma 1. We report their sample-wise pathway coverage and use the prescribed fallback strategy when the required coverage cannot be attained within the expert budget.

C Proof of Conditional Pathway Invariance

Proof of Lemma 1. Let $U_\ell^v(x)$ denote the complete hidden representation entering MoE layer ℓ in model $v \in \{0, u\}$, where 0 and u indicate the original and unlearned models, respectively. For routing unit t, the routing distribution is

$$\pi_{\ell,t}^v(x) = \mathrm{softmax}\left(r_{\theta_\ell}\left(U_{\ell,t}^v(x)\right)\right), \tag{54}$$

and the active expert set is

$$\mathcal{E}_{\ell,t,k}^v(x) = \mathrm{TopK}\left(\pi_{\ell,t}^v(x), k\right). \tag{55}$$

Because the router parameters and tie-breaking rule are fixed, identical layer inputs produce identical routing distributions, active sets, and gated weights.

We prove the result by induction over the MoE layers. Before the first MoE layer, all shared components are fixed and receive the same input x. Therefore,

$$U_1^u(x) = U_1^0(x). \tag{56}$$

Suppose that the hidden representations entering layer ℓ are identical:

$$U_\ell^u(x) = U_\ell^0(x). \tag{57}$$

The frozen router consequently produces

$$\pi_{\ell,t}^u(x) = \pi_{\ell,t}^0(x), \qquad \mathcal{E}_{\ell,t,k}^u(x) = \mathcal{E}_{\ell,t,k}^0(x) \tag{58}$$

for every routing unit t. By the condition of Lemma 1, none of these active experts belongs to $\mathcal{S}_\ell^\star$. Their parameters are therefore identical before and after unlearning:

$$h_{\ell,m}^u = h_{\ell,m}^0, \qquad m \in \mathcal{E}_{\ell,t,k}^0(x). \tag{59}$$

Because both their inputs and parameters are unchanged, all active expert outputs are identical. Together with the unchanged gated weights, this gives

$$Z_\ell^u(x) = Z_\ell^0(x), \tag{60}$$

where $Z_\ell^v(x)$ denotes the aggregated output of MoE layer ℓ.

All shared transformations between layers, including residual connections, attention modules, and normalization operations, are fixed and deterministic. Hence,

$$U_{\ell+1}^u(x) = U_{\ell+1}^0(x). \tag{61}$$

This establishes the induction step. Repeating the argument across all MoE layers shows that the final hidden representations are identical. Since the prediction head is also fixed, the two models produce identical logits and predictive distributions:

$$p_u(\cdot \mid x) = p_0(\cdot \mid x). \tag{62}$$

The proof ends. □

The condition is stated using the pre-unlearning pathway, but no assumption of externally fixed routing assignments is required. The induction shows that the routes remain unchanged because every preceding representation remains unchanged. If an input activates an updated expert, this argument no longer applies, since the modified expert may change the representations received by downstream routers.

The proof also requires deterministic, batch-independent dispatch. If capacity constraints, token dropping, or other batch-dependent routing operations are used, identical routing logits do not necessarily imply identical executed pathways. Finally, the result applies only to strict path disjointness and not to examples excluded from the at-risk set using a positive exposure threshold.

D EXPERIMENT SETUPS

D.1 DATASETS

We adopt PACS Li et al. (2017), OfficeHome Venkateswara et al. (2017), CIFAR-100 Krizhevsky et al. (2009) and Tiny-ImageNet Le et al. (2015) as benchmark datasets for our experiments. Details about each dataset are mentioned below.

PACS. consists of $9,991$ images structured across 7 semantic classes. It features four distinct domains that present different visual and style shifts, namely, Photo, Art Painting, Cartoon, and Sketch.

OfficeHome. contains approximately $15,588$ images representing 65 object categories typically found in office and home environments, across four domains, namely, Art, Clipart, Product, and Real-World.

CIFAR-100. consists of $60,000$ color images with a resolution of 32×32 pixels representing 100 classes. Each class contains 500 training images and 100 test images.

Tiny-ImageNet. is a scaled-down version of ImageNet, containing $100,000$ training images of 64×64 pixels evenly distributed across 200 classes, with 500 images per class.

D.2 UNLEARNING SCENARIOS

We compare our proposed method with baselines under three unlearning scenarios, namely, Random Unlearning, Class Unlearning, and Domain Unlearning. Details about each unlearning scenario are described below.

Random Unlearning. A certain percentage of the training data is randomly sampled to construct the forget set $\mathcal{D}_f$, while the remaining data forms the retain set $\mathcal{D}_r$.

Class Unlearning. The forget set $\mathcal{D}_f$ is created by concatenating all training samples from one or more target classes, while the rest is used as the retain set $\mathcal{D}_r$.

Domain Unlearning. The forget set $\mathcal{D}_f$ is constructed from all samples belonging to some specific domains, while the rest is used as the retain set $\mathcal{D}_r$

D.3 BASELINES

We compare ModULE with five groups of unlearning baselines, organized by their principal mechanisms. The original model and retraining on retained data serve as reference models.

Basic unlearning. We include Fine-Tuning (FT) (Golatkar et al., 2020), Gradient Ascent (GA) (Golatkar et al., 2020), and Random Labeling (RL) (Graves et al., 2021), which respectively train on retained data, maximize prediction loss on forgotten data, and train on forgotten examples with reassigned labels.

Objective- and representation-based unlearning. We include SCADA (Devalapally et al., 2026), POUR (Le et al., 2026), Q-MUL (Tong et al., 2025), SCRUB (Kurmanji et al., 2023), SG-Unlearning (Di et al., 2025), Boundary Expanding (BE) and Boundary Shrink (BS) (Chen et al., 2023a), and RepSelect (Sondej et al., 2026). These methods use mechanisms including adversarial optimization, distillation, gradient reweighting, decision-boundary modification, and representation interventions to balance forgetting and retention.

Parameter-based unlearning. We include SalUn (Fan et al., 2024), SSD (Foster et al., 2024), and ℓ_1-sparse (Jia et al., 2023). SalUn restricts updates using weight saliency, SSD dampens parameters according to their relative importance to forgotten data, and ℓ_1-sparse introduces sparsity regularization during unlearning.

Architecture-based unlearning. We include SPM (Zheng et al., 2026a), which uses a semi-parametric architecture to support deletion of training-sample contributions without changing model parameters. This provides a comparison with another approach that designs model structure to facilitate subsequent unlearning.

MoE-specific unlearning. We include SEUF (Zhuang et al., 2025) and GRIP (Zhu et al., 2026). SEUF selects forget-associated experts and anchors their activation, whereas GRIP constrains or realigns routers to preserve retained routing while the underlying unlearning algorithm updates model parameters. For the vanilla MoE, we use GMoE (Li et al., 2023) as reference baseline, and apply the unlearning on the entire experts. These baselines evaluate alternative mechanisms for unlearning in pretrained MoEs.

Table 2: Baseline-specific hyperparameters and their default values, referenced from the original papers.

Algorithm	**Key Hyperparameter**	**Default Value**
Fine-Tuning (FT)	Learning Rate (η)	Same as pre-training
Gradient Ascent (GA)	GA Learning Rate Scaling	$0.1 \times \eta$
Random Labeling (RL)	Noise Ratio	0.1
SCADA (Devalapally et al., 2026)	Adversarial Scaling Factor (λ)	0.5
POUR (Le et al., 2026)	Projection Radius (R)	1.0
Q-MUL (Tong et al., 2025)	Quantization Bits / Reweight Factor	4-bit / 0.8
SCRUB (Kurmanji et al., 2023)	Maximization Step Weight (γ)	1.5
	Distillation Temperature (T)	4.0
SG-Unlearning (Di et al., 2025)	Adversarial Step Size	0.01
Boundary Expanding (BE) (Chen et al., 2023a)	Expansion Margin (δ)	0.5
Boundary Shrink (BS) (Chen et al., 2023a)	Shrinking Step Size	0.1
MoE-specific unlearning		
GRIP (Zhu et al., 2026)	Null-Space Cutoff (ϵ_{null})	1×10^{-2}
SEUF (Zhuang et al., 2025)	Router Anchor Loss Weight (α)	1.0
RepSelect (Sondej et al., 2026)	Cached-Update Strength (α)	0.1
	Retained Principal Components (k)	512
SPM (Zheng et al., 2026a)	Fusion Parameter (α)	0.7
SalUn (Fan et al., 2024)	Saliency Masking Threshold (τ)	Top 5%
SSD (Foster et al., 2024)	Dampening Coefficient (α)	10.0
	Selection Ratio (θ)	0.1
l_1-sparse (Jia et al., 2023)	Sparsity Regularization Weight (λ)	1×10^{-4}

D.4 Model Architecture and Implementation Details

Model Architecture. For ease of notation, we refer to the DeiT-S Touvron et al. (2021) version that is augmented with the ModULE proposal as ModULE-S. As ModULE introduces architectural and training objective differences during the learning phase, for a fair comparison, all experiments are conducted using ModULE-S as the base model.

Implementation Details. During the learning phase, the base model is trained for 100 epochs, with a batch size of 128 applied for all datasets, using the AdamW optimizer with an initial learning rate of 5×10^{-4} and a weight decay of 0.05. We use the exact same training configuration during the learning phase to train the retrained-from-scratch baseline (the gold standard) for each dataset. During the unlearning phase, the unlearning process is run for 5 epochs using the same batch size and optimizer as in the learning phase. The unlearning learning rate is tuned via grid search within the range of $\{a \times 10^{-b}\}$ where $a \in \{1, 3, 5, 7\}$ and $b \in \{2, 3, 4\}$, and the best performance is reported.

D.5 Evaluation Metrics

We evaluate unlearning performance using five metrics, namely Forget Accuracy (FA), Retain Accuracy (RA), Test Accuracy (TA), Membership Inference Attack (MIA), and Average Gap (Avg. Gap).

Forget Accuracy). FA assesses the model's accuracy on the forget set $\mathcal{D}_f$. Lower FA values indicate better removal of the forgotten information.

Retain Accuracy. RA assesses the model's accuracy on the retain set $\mathcal{D}_r$, where higher values indicate better retention of the model's remaining knowledge.

Test Accuracy. TA assesses the model's accuracy on the test set, which is used to measure the impact of unlearning on the model's overall generalization capacity.

Membership Inference Attack. MIA evaluates whether an attacker can distinguish between forget samples and unseen samples using model outputs. Following the protocol in (Di et al., 2025), we utilize MIA to quantify membership leakage. An MIA value close to 0.5 indicates that the attacker performs no better than random guessing, suggesting that membership information regarding the forget set has been successfully obscured.

Average Gap (Avg. Gap). Avg. Gap quantifies the overall differences across the 4 metrics (FA, RA, TA, and MIA) between the unlearned model and the gold-standard. It is calculated by averaging the absolute differences of these metrics against the retrained-from-scratch baseline. A lower Avg. Gap indicates the closer an approximate unlearning method matches the behavior of the ideal retrained model.

D.6 Efficiency Metrics

To quantify computational efficiency, we report the Updated-parameter Ratio.

Updated-parameter Ratio (URP). The updated-parameter ratio is defined as $\mathrm{UPR} = \frac{\#\text{parameters updated during unlearning}}{\#\text{total parameters}}$. A lower UPR indicates more localized unlearning.

D.7 Localization Diagnostics

Because ModULE is designed to support localized expert-level forgetting, we report additional diagnostics that measure whether forget-set information is concentrated in a small subset of experts.

Forget-Expert Routing Mass (FEM). Let $\mathcal{M}_f$ denote the selected forget-relevant experts and let $\pi_m(x)$ be the router probability assigned to expert m for input x. We define forget-expert routing mass as $\mathrm{FEM}(\mathcal{D}_f) = \frac{1}{|\mathcal{D}_f|}\sum_{x\in\mathcal{D}_f}\sum_{m\in\mathcal{M}_f}\pi_m(x)$. Higher FEM indicates that forget samples are concentrated in the selected forget experts.

At-risk Retain Ratio (ARR). Let $\mathcal{E}_k(x)$ denote the top-k experts activated by input x. We define the at-risk retain ratio as $\mathrm{ARR} = \frac{|\{x\in\mathcal{D}_r:\mathcal{E}_k(x)\cap\mathcal{M}_f\neq\emptyset\}|}{|\mathcal{D}_r|}$. Lower ARR indicates that fewer retained samples depend on the experts updated during unlearning, indicating better modular separation and lower risk of collateral forgetting.

Retain-forget Routing Overlap (RFO). We further measure the overlap between experts activated by forget and retain samples: $\mathrm{RFO}(\mathcal{D}_f,\mathcal{D}_r) = \frac{1}{|\mathcal{D}_f||\mathcal{D}_r|}\sum_{x_f\in\mathcal{D}_f}\sum_{x_r\in\mathcal{D}_r}\frac{|\mathcal{E}_k(x_f)\cap\mathcal{E}_k(x_r)|}{|\mathcal{E}_k(x_f)\cup\mathcal{E}_k(x_r)|}$. Lower overlap indicates that forget and retain samples activate more distinct expert subsets, indicating that localized unlearning is less likely to damage retained knowledge.

D.8 Router Evaluation During the Learning Phase

We evaluate whether the expert-routing structure learned from the training data remains stable on held-out samples from the same domains. The evaluation is performed independently at each MoE layer and considers both the soft routing probabilities and the hard expert selections used in the forward pass.

Let $\mathcal{L}$ the set of MoE layers. For domain $d \in \mathcal{D}$, MoE layer $\ell \in \mathcal{L}$, and split $s \in \{\mathrm{tr}, \mathrm{te}\}$, let

$$\boldsymbol{\pi}^s_{i,d,\ell} = [\pi^s_{i,d,\ell,1}, \ldots, \pi^s_{i,d,\ell,M}] \in \Delta^{M-1}$$

denote the router softmax probabilities for token i. All measurements are computed with the model in evaluation mode and with the same test-time transformations applied to both splits. This disables router noise and avoids confounding routing differences with training-time data augmentation.

The reported layer index enumerates only the MoE layers encountered in the network. Thus, `layer0` denotes the first MoE layer rather than the first Transformer block. Moreover, expert identities are local to each MoE layer, so expert m at layer ℓ is distinct from expert m at another layer.

D.8.1 Domain-Conditioned Routing-Mass Distribution

For every domain and MoE layer, we aggregate the soft router probabilities over all evaluated tokens:

$$\bar{\boldsymbol{\pi}}^s_{d,\ell} = \frac{1}{N^s_{d,\ell}} \sum_{i=1}^{N^s_{d,\ell}} \boldsymbol{\pi}^s_{i,d,\ell}, \qquad \sum_{m=1}^{M} \bar{\pi}^s_{d,\ell,m} = 1, \tag{63}$$

where $N^s_{d,\ell}$ is the number of tokens from domain d and split s.

The vectors $\bar{\boldsymbol{\pi}}^{\mathrm{tr}}_{d,\ell}$ and $\bar{\boldsymbol{\pi}}^{\mathrm{te}}_{d,\ell}$ represent the complete training and test routing-mass distributions. Logged quantities such as `mass_train` and `mass_test` are only individual coordinates:

$$\texttt{mass_train}_{d,\ell,m} = \bar{\pi}^{\mathrm{tr}}_{d,\ell,m}, \qquad \texttt{mass_test}_{d,\ell,m} = \bar{\pi}^{\mathrm{te}}_{d,\ell,m}.$$

They are therefore not interpreted as standalone system-level metrics.

To expose how the routing distribution changes, we define the signed shift

$$\Delta\bar{\boldsymbol{\pi}}_{d,\ell} = \bar{\boldsymbol{\pi}}^{\mathrm{te}}_{d,\ell} - \bar{\boldsymbol{\pi}}^{\mathrm{tr}}_{d,\ell}. \tag{64}$$

A positive coordinate indicates that the corresponding expert gains routing mass on test data, while a negative coordinate indicates that it loses mass. The per-expert absolute difference

$$\texttt{mass_diff}_{d,\ell,m} = \left|\bar{\pi}^{\mathrm{te}}_{d,\ell,m} - \bar{\pi}^{\mathrm{tr}}_{d,\ell,m}\right|$$

is retained only as a diagnostic because it discards the direction of the change and does not describe the complete distribution.

We summarize the difference between the complete training and test distributions using total variation distance:

$$\mathrm{TV}^{\mathrm{mass}}_{d,\ell} = \frac{1}{2} \left\|\bar{\boldsymbol{\pi}}^{\mathrm{tr}}_{d,\ell} - \bar{\boldsymbol{\pi}}^{\mathrm{te}}_{d,\ell}\right\|_1. \tag{65}$$

This quantity lies in $[0, 1]$. A value of zero indicates identical routing distributions, while a larger value indicates that more probability mass must be reassigned to make the two distributions agree. We average this quantity across domains at each layer:

$$\mathrm{MassMeanTV}_\ell = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \mathrm{TV}^{\mathrm{mass}}_{d,\ell}. \tag{66}$$

D.8.2 Domain-Conditioned Expert-Selection Distribution

The routing-mass distribution measures soft router confidence, but the sparse forward pass activates only the top-k_g experts, where k_g is the gating cardinality. We define the token-level selection indicator as

$$a^s_{i,d,\ell,m} = \mathbb{I}\left[m \in \mathrm{TopK}\left(\boldsymbol{\pi}^s_{i,d,\ell}, k_g\right)\right]. \tag{67}$$

The corresponding expert-selection frequency is

$$q^s_{d,\ell,m} = \frac{1}{N^s_{d,\ell}} \sum_{i=1}^{N^s_{d,\ell}} a^s_{i,d,\ell,m}. \tag{68}$$

Thus, $q^s_{d,\ell,m}$ is the fraction of tokens for which expert m is actually activated.

The quantities `sel_train` and `sel_test` are coordinates of the training and test selection-frequency vectors:

$$\mathbf{q}^{\mathrm{tr}}_{d,\ell} = [q^{\mathrm{tr}}_{d,\ell,1}, \ldots, q^{\mathrm{tr}}_{d,\ell,M}], \qquad \mathbf{q}^{\mathrm{te}}_{d,\ell} = [q^{\mathrm{te}}_{d,\ell,1}, \ldots, q^{\mathrm{te}}_{d,\ell,M}].$$

Because every token activates k_g experts,

$$\sum_{m=1}^{M} q^{s}_{d,\ell,m} = k_g. \tag{69}$$

We therefore obtain a probability distribution by normalizing the selection frequencies:

$$\tilde{\mathbf{q}}^{s}_{d,\ell} = \frac{\mathbf{q}^{s}_{d,\ell}}{k_g}, \qquad \sum_{m=1}^{M} \tilde{q}^{s}_{d,\ell,m} = 1. \tag{70}$$

The signed change in expert activation is

$$\Delta\mathbf{q}_{d,\ell} = \mathbf{q}^{\mathrm{te}}_{d,\ell} - \mathbf{q}^{\mathrm{tr}}_{d,\ell}. \tag{71}$$

In contrast, the logged per-expert value

$$\texttt{sel_diff}_{d,\ell,m} = \left|q^{\mathrm{te}}_{d,\ell,m} - q^{\mathrm{tr}}_{d,\ell,m}\right|$$

measures only the magnitude of the change for one expert.

We quantify the difference between the complete normalized selection distributions using

$$\mathrm{TV}^{\mathrm{sel}}_{d,\ell} = \frac{1}{2}\left\|\tilde{\mathbf{q}}^{\mathrm{tr}}_{d,\ell} - \tilde{\mathbf{q}}^{\mathrm{te}}_{d,\ell}\right\|_1. \tag{72}$$

The domain-averaged selection shift at layer ℓ is

$$\mathrm{SelMeanTV}_{\ell} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \mathrm{TV}^{\mathrm{sel}}_{d,\ell}. \tag{73}$$

A lower value indicates more stable expert activation between the training and test subsets.

D.8.3 Dominant-Expert Agreement

We additionally evaluate whether the most frequently selected experts remain the same between training and test data. Let k_u denote the number of dominant experts used by this diagnostic:

$$\mathcal{T}^{\mathrm{tr}}_{d,\ell} = \mathrm{TopK}\left(\mathbf{q}^{\mathrm{tr}}_{d,\ell}, k_u\right), \tag{74}$$

$$\mathcal{T}^{\mathrm{te}}_{d,\ell} = \mathrm{TopK}\left(\mathbf{q}^{\mathrm{te}}_{d,\ell}, k_u\right). \tag{75}$$

The exact-match rate is

$$\mathrm{SelExactMatchRate}_{\ell} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \mathbb{I}\left[\mathcal{T}^{\mathrm{tr}}_{d,\ell} = \mathcal{T}^{\mathrm{te}}_{d,\ell}\right]. \tag{76}$$

It measures the fraction of domains whose complete top-k_u expert sets agree between training and test data.

To provide partial credit, we compute

$$\mathrm{SelMeanTopKOverlap}_{\ell} = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \frac{\left|\mathcal{T}^{\mathrm{tr}}_{d,\ell} \cap \mathcal{T}^{\mathrm{te}}_{d,\ell}\right|}{k_u}. \tag{77}$$

Both metrics lie in $[0, 1]$, with larger values indicating stronger agreement. When $k_u = 1$, the exact-match rate and mean top-k_u overlap are equivalent. When $k_u > 1$, the overlap distinguishes partial from complete agreement. Setting $k_u = k_g$ aligns the diagnostic set size with the number of experts activated during the forward pass.

Table 3: Router evaluation metrics during the learning phase. The desired direction indicates greater training-to-test routing stability.

Metric	Routing information	Comparison	Desired
$\mathrm{MassMeanTV}$	Soft routing mass	Complete distribution	Lower
$\mathrm{SelMeanTV}$	Hard expert activation	Complete distribution	Lower
$\mathrm{SelExactMatchRate}$	Hard expert activation	Complete top-k_u set	Higher
$\mathrm{SelMeanTopKOverlap}$	Hard expert activation	Partial top-k_u agreement	Higher

D.8.4 Visualization and Interpretation

We visualize the complete distributions using domain-by-expert heatmaps. For each MoE layer, we report the following matrices:

$$\mathbf{P}_\ell^{\mathrm{tr}} = \{\bar{\pi}^{\mathrm{tr}}_{d,\ell,m}\}_{d,m}, \qquad \mathbf{P}_\ell^{\mathrm{te}} = \{\bar{\pi}^{\mathrm{te}}_{d,\ell,m}\}_{d,m},$$
$$\Delta\mathbf{P}_\ell = \{\Delta\bar{\pi}_{d,\ell,m}\}_{d,m}, \qquad \mathbf{Q}_\ell^{\mathrm{tr}} = \{q^{\mathrm{tr}}_{d,\ell,m}\}_{d,m},$$
$$\mathbf{Q}_\ell^{\mathrm{te}} = \{q^{\mathrm{te}}_{d,\ell,m}\}_{d,m}, \qquad \Delta\mathbf{Q}_\ell = \{\Delta q_{d,\ell,m}\}_{d,m}.$$

The training and test heatmaps use the same color scale. The signed-difference heatmaps use a zero-centered diverging scale to distinguish experts that gain usage from those that lose usage. Each MoE layer is visualized separately because expert identities are not shared across layers.

Table 3 summarizes the scalar metrics used alongside these distributional visualizations.

The soft and hard distribution metrics are complementary. A low $\mathrm{MassMeanTV}$ indicates stable router probabilities, whereas a low $\mathrm{SelMeanTV}$ indicates stable expert activation frequencies. Small changes near the top-k_g boundary may produce a noticeable hard-selection shift even when the soft distributions remain similar. Conversely, router confidence may change while preserving the selected experts.

The exact-match and overlap metrics describe only dominant-expert agreement. They do not characterize the complete expert distribution and are therefore used as secondary diagnostics. We treat the two total variation metrics and the corresponding heatmaps as the primary evidence of training-to-test routing stability.

Finally, routing stability does not by itself establish meaningful expert specialization. A collapsed router that sends every domain to the same expert can achieve perfect stability. We therefore interpret these measurements jointly with task accuracy, expert-load balance, dead-expert statistics, and domain-conditioned expert-usage patterns.

E Ablation Studies

In this section, we present ablation studies of our proposed ModULE method. Unless otherwise specified, all ablation experiments are conducted using the PACS dataset, with the number of experts M set to 12, the number of learning experts selected $k = 4$, and the number of unlearning experts activated $k_u = 4$. The regularization weights used in learning phase are configured as $\lambda_{\text{sp}} = 0.1$, $\lambda_{\text{bal}} = 0.25$, and $\lambda_{\text{div}} = 0.1$, while the regularization weights used in unlearning phase are set to $\beta = 0.25$, $\gamma = 0.25$, and $\eta = 0.25$. The MoE layers are by default inserteds to layers 7 and 9 in the ViT backbone, where each expert is structured as a MLP with a depth of 2 and a hidden dimension ratio of 0.1.

E.1 Ablation on Expert Selection Rules

We compare different strategies for selecting forget-relevant experts: **(i)** random expert selection, **(ii)** raw forget routing score, **(iii)** forget-retain difference score, **(iv)** forget-retain ratio score and **(v)** gradient-based influence score.

Random Expert Selection. This strategy involves randomly picking experts to update during the unlearning.

Raw Forget Routing Score. Using the equations to calculate the forget and retain routing masses, the raw forget routing score is defined by the forget routing mass, i.e., $\rho_m^{\text{raw}} = s_m(\mathcal{D}_f)$.

Forget-Retain Difference Score. The forget-retain difference score is calculated as the difference between the forget routing mass and the retain routing mass, i.e., $\rho_m^{\text{diff}} = s_m(\mathcal{D}_f) - \alpha s_m(\mathcal{D}_r)$.

Forget-Retain Ratio Score. The forget-retain ratio score is determined by dividing the forget routing mass by the retain routing mass, i.e., $\rho_m^{\text{ratio}} = \frac{s_m(\mathcal{D}_f)}{s_m(\mathcal{D}_r)+\epsilon}$.

Gradient-Based Influence Score. This scoring strategy selects the top-k experts based on the magnitude of the gradients of each expert on the forget set $\mathcal{D}_f$.

E.1.1 Ablation on Unlearning Update Scope

We evaluate how the choice of training parameters during the unlearning phase affects forgetting, retention, and efficiency. We compare updating only selected experts against progressively expanding the update scope. As described in Table 4 and Table 5,

Table 4: Performance comparison among different parameter update scope strategies under 1 class unlearning scenarios in PACS. The best results are highlighted in green.

Update Scope	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**	**Perf. Gap** ↑
Selected Experts Only	81.75	99.02	91.65	0.649	**25.46**
Selected Experts + Classifier Head	46.49	**99.12**	91.40	0.637	16.39
Selected Experts + Router	81.39	98.91	91.16	0.633	25.13
Selected Experts + Last Encoder Block	5.72	97.58	90.80	0.631	6.60
All Experts	7.53	98.76	**92.49**	0.621	5.94
FULL Model	**0.07**	83.81	79.42	0.530	8.92

Table 5: Performance comparison among different parameter update scope strategies under 1 domain unlearning scenarios in PACS dataset. The best results are highlighted in green.

Update Scope	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**	**Perf. Gap** ↑
Selected Experts Only	35.75	96.35	82.80	0.498	6.53
Selected Experts + Classifier Head	30.14	96.54	83.44	0.484	8.08
Selected Experts + Router	69.64	99.25	85.03	0.467	10.11
Selected Experts + Last Encoder Block	19.47	98.96	84.55	0.502	9.41
All Experts	19.85	92.54	79.14	0.487	**12.56**
Full Model	**19.28**	**99.48**	**88.22**	0.498	8.51

E.1.2 Ablation on Top-k Routing

Expert-update budget. Tables 6 and 7 examine the number of selected experts updated during unlearning, denoted by K_u, while keeping the routing width fixed. Increasing K_u generally strengthens forgetting, but the benefit quickly saturates. For class deletion, increasing K_u from 2 to 6 reduces FA from 26.72% to 0.80%, while RA decreases only from 99.06% to 98.97% and TA from 93.70% to 92.98%. Increasing the budget further to $K_u = 8$ reduces FA by only another 0.51 percentage points, while increasing the updated parameter fraction from 20.94% to 27.91% and further reducing utility. Thus, $K_u = 6$ appears to provide a reasonable operating point for class deletion. For domain deletion, most of the forgetting improvement occurs between $K_u = 2$ and $K_u = 4$, where FA decreases from 50.08% to 24.99% without reducing TA. Increasing K_u from 4 to 8 lowers FA by only another 1.67 percentage points, while RA decreases and the fraction of updated parameters doubles. This suggests that $K_u = 4$ is sufficient under the reported domain-deletion setting. Overall, the appropriate update budget depends on the deletion request: a broader update is helpful until the selected experts cover

most of the relevant forget computation, after which additional experts mainly increase parameter cost and retain exposure. MIA does not exhibit a consistent trend across K_u, so it does not provide a clear preference among these configurations.

Table 6: Effect of the number of updated experts K_u under 1 class unlearning PACS dataset. The best results are highlighted in green.

K_u	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**	**Unlearn Steps** ↓	**Updated Params** (%) ↓
2	26.72	99.06	93.68	0.5727	220	6.98
4	14.92	99.01	93.70	0.5827	220	13.96
6	0.80	98.97	92.98	0.5797	220	20.94
8	0.29	98.81	92.49	0.5864	220	27.91

Table 7: Effect of the number of updated experts K_u under 1 domain unlearning PACS dataset. The best results are highlighted in green.

K_u	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**	**Unlearn Steps** ↓	**Updated Params** (%) ↓
2	50.08	98.49	90.29	0.5018	350	6.98
4	24.99	97.99	90.29	0.4844	350	13.96
6	24.14	97.72	90.45	0.5086	350	20.94
8	23.32	97.37	90.41	0.5118	350	27.91

Routing sparsity. Tables 8 and 9 instead vary the architectural routing width k, namely the number of experts activated for each input, while fixing the expert-update budget at $K_u = 12$. For class deletion, increasing k from 2 to 6 reduces FA from 14.92% to 4.63%, while RA remains at 99.14% and TA decreases modestly from 94.19% to 93.71%. Increasing k beyond 6 produces progressively smaller reductions in FA, reaching 2.75% under dense routing, while TA continues to decline slightly. These results indicate that moderate routing widths capture most of the forgetting benefit without requiring dense expert activation. For domain deletion, increasing k from 2 to 4 reduces FA from 22.47% to 19.44%, but larger values provide almost no further improvement, with FA remaining close to 19%. At the same time, RA and TA become substantially non-monotonic for large k. This behavior suggests that broader routing increases overlap between forget and retain pathways without providing corresponding gains in domain forgetting. Consequently, a moderate routing width such as $k = 4$–6 appears preferable to dense routing in these experiments. The irregular utility results at large k should nevertheless be verified across multiple random seeds and reported with standard deviations.

Table 8: Effect of top-k active experts during unlearning phase, under *1 class unlearning*, $k_u = 12$ PACS dataset. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

k	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**
2	14.92	99.14	94.19	0.59
4	7.31	99.14	94.07	0.57
6	4.63	99.14	93.71	0.57
8	3.54	99.11	93.58	0.569
10	2.82	99.08	93.58	0.56
12	2.75	99.03	93.46	0.56

Together, the two ablations control different aspects of ModULE. The routing width k determines how broadly each input uses the expert bank and must be chosen during model learning, whereas K_u determines how many localized experts are modified for a particular deletion request. The results favor

Table 9: Effect of top-k active experts during unlearning phase, under *1 domain unlearning*, $k_u = 12$. PACS dataset. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

k	**FA** (%) ↓	**RA** (%) ↑	**TA** (%) ↑	**MIA**
2	22.47	94.94	87.58	0.50
4	19.44	94.73	87.10	0.50
6	19.10	93.28	83.12	0.49
8	19.09	87.68	77.39	0.50
10	19.07	83.59	91.64	0.49
12	19.06	93.72	85.83	0.49

moderately sparse routing during learning and a request-dependent update budget during unlearning. Final configurations should also be compared with the Retrain reference, since minimizing FA alone cannot distinguish successful forgetting from excessive degradation.

E.1.3 Ablation on Number of Experts

We vary the number of experts M to analyze how modular capacity affects unlearning. Table 10 shows that increasing the expert-bank size M does not necessarily improve forgetting when the unlearning budget is kept fixed at $k_u = 2$. Although a larger expert bank leads to lower routing overlap, the forget-set computation can also be distributed across a larger number of experts. Consequently, updating only two experts covers a progressively smaller fraction of the forget-relevant routes as M increases, making the fixed unlearning budget insufficient to fully remove the target information. This explains why the forget accuracy deteriorates for larger expert banks despite improved routing separation.

To account for this effect, Table 11 increases k_u together with the number of experts. Once the unlearning budget is scaled accordingly, effective forgetting is recovered across different expert-bank sizes while retain and test performance remain high. Together, Tables 10 and 11 reveal an important interaction between modular capacity and the unlearning budget: increasing M can provide finer routing separation, but it can also spread the forget-set computation over more experts, requiring a correspondingly larger k_u to maintain sufficient forget-route coverage.

Table 10: Effect of the number of experts M, $k_u = 2$. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

M	**TA**↑	**FA**↓	**RA**↑	**MIA**↓	**Routing Overlap**↓
4	96.85	0.14	99.11	0.52	1.00
8	96.84	2.56	99.32	0.51	0.47
12	99.14	7.31	94.07	0.57	0.42
16	91.72	83.34	99.11	0.528	0.36
24	89.01	78.32	94.73	0.484	0.19

E.1.4 Architecture Control Ablation

We compare ModULE with dense and MoE controls under matched parameter budgets. The results are reported in Table 12.

Table 12 separates the effect of the modular architecture from the parameter budget. The dense controls achieve low forget accuracy, but this comes with substantial degradation in retained and test accuracy. In contrast, the standard MoE preserves retained utility but exhibits much weaker forgetting. ModULE achieves 5.72% FA while maintaining 99.55% RA and 95.04% TA; compared with the MoE without modular losses, it further reduces FA from 6.55% to 5.72% while slightly improving

Table 11: Effect of the number of experts M. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

M	K_u	Epoch	TA↑	FA↓	RA↑	MIA↓	Routing Overlap↓
4	2	14	96.85	0.14	99.11	0.52	1.00
8	4	20	96.97	1.09	99.98	0.57	0.49
12	6	20	96.37	3.19	99.03	0.57	0.37
16	8	10	96.13	4.06	98.47	0.51	0.35
24	12	13	96.73	0.36	99.26	0.65	0.19

Table 12: Architecture control under matched parameter budgets. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

Architecture	Total Params. (M)	Active Params. (M)	FA↓	RA↑	TA↑
Dense, same active params	22.05	22.05	1.21	27.89	26.39
Dense, same total params	29.14	29.14	1.34	59.60	53.51
Standard MoE	29.15	22.06	45.62	99.23	94.79
MoE w/o modular losses	29.15	22.06	6.55	99.43	94.79
ModULE	29.15	22.06	5.72	**99.55**	**95.04**

retained and test accuracy. These controls indicate that the observed forgetting–utility trade-off is not reproduced simply by matching total or active parameter budgets.

E.1.5 Effect of the Number of Forgotten Classes

We examine membership leakage when unlearning one, two, or three classes, comparing ModULE with the original model and retraining from scratch. MIA accuracy closer to 50% indicates that the evaluated attacker has less ability to distinguish forgotten examples from unseen examples.

As shown in Figure 4, ModULE remains close to the original model and above Retrain when forgetting one class, indicating limited improvement against this attack. For two- and three-class deletion, its mean MIA accuracy approaches 50% and the corresponding Retrain results. These observations suggest that ModULE's membership leakage depends on the deletion setting. However, the original model's MIA accuracy also varies across settings, so the observed trend cannot be attributed solely to the number of forgotten classes.

F Additional In-depth Analyses

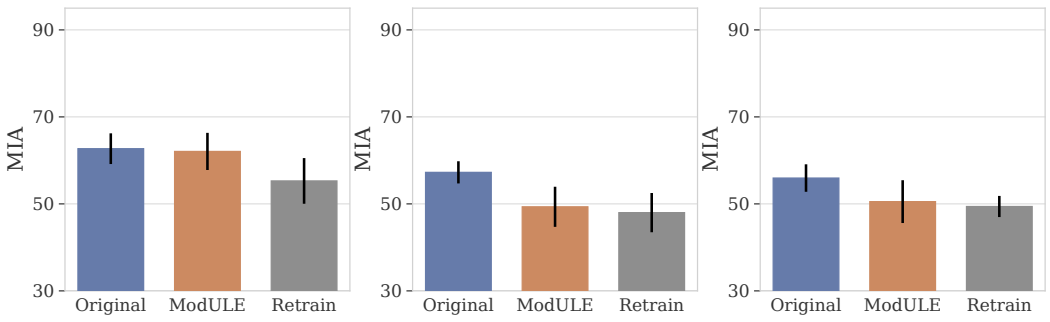
F.1 Routing-Mass Distribution

We further analyze how the learned router distributes routing mass across experts from two complementary perspectives: domains and semantic classes. Since expert identities are specific to each MoE layer, we visualize the routing distributions of the two layers separately.

Domain-wise Routing Distribution. Figure 5 shows the routing-mass distribution conditioned on the four PACS domains.

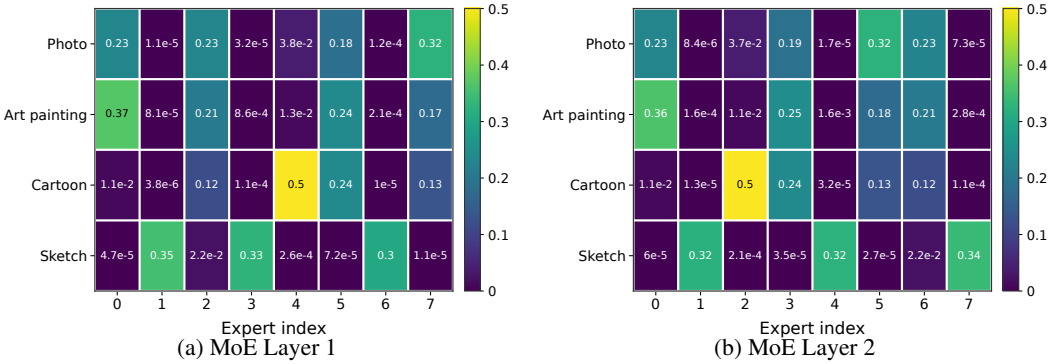
The routing patterns are clearly non-uniform across domains. Certain experts receive substantially higher routing mass for particular domains, whereas others are utilized more consistently across multiple domains. Moreover, the two MoE layers exhibit different allocation patterns, suggesting that they capture complementary aspects of domain-dependent variation rather than learning identical expert assignments.

Class-wise Routing Distribution. We next examine whether expert utilization also varies with semantic class, independently of the domain partition. Figure 6 shows the corresponding class-by-expert routing-mass distributions. The class-wise visualization further reveals that expert utilization



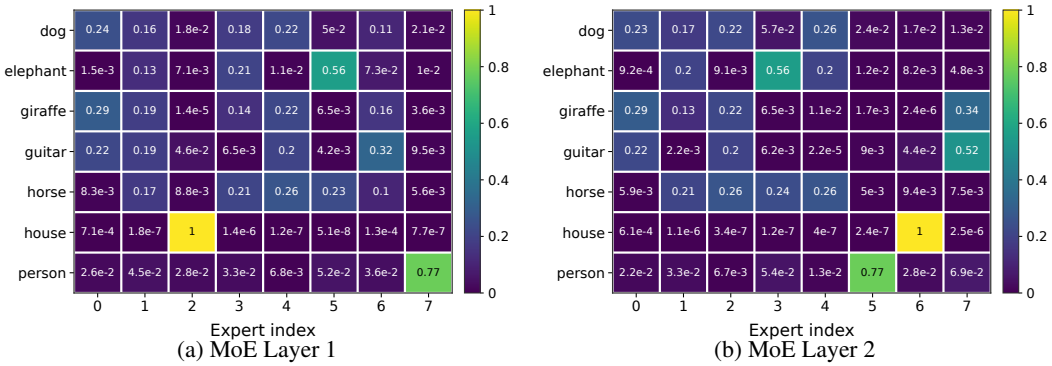
(a) Forget 1 class. (b) Forget 2 classes. (c) Forget 3 classes.

Figure 4: Membership inference attack accuracy under deletion requests targeting one, two, or three classes. Values closer to 50% indicate weaker distinguishability under the evaluated attack.



(a) MoE Layer 1 (b) MoE Layer 2

Figure 5: Domain-wise routing-mass distributions across experts on the PACS test set. Rows correspond to domains and columns to experts. Each cell reports the average routing mass assigned to the corresponding expert for samples from a given domain. Results are shown separately for the two MoE layers.



(a) MoE Layer 1 (b) MoE Layer 2

Figure 6: Class-wise routing-mass distributions across experts on the PACS test set. Rows correspond to semantic classes and columns to experts. Each cell reports the average routing mass assigned to the corresponding expert for samples from a given class. Results are shown separately for the two MoE layers.

is not solely determined by domain identity. Different semantic classes induce distinct routing patterns, with some experts receiving relatively larger mass for particular classes. Together with the domain-wise results, these observations suggest that the learned router organizes expert usage along both domain-related and semantic dimensions.

F.2 Router Stability Evaluation

We next empirically examine whether the routing structure learned from the training data generalizes to held-out samples from the same domains. We use the routing-stability metrics defined in Appendix D.8 and evaluate the two MoE layers separately, since expert identities are local to each layer.

Train-to-Test Routing-Mass Stability. We further examine whether the learned expert-usage patterns remain stable when moving from the training split to held-out test samples. We analyze train-to-test routing-mass changes from both domain-wise and class-wise perspectives, and report the two MoE layers separately.

Domain-wise stability. Figure 7 visualizes the domain-by-expert difference in routing mass between the training and test splits on PACS. Across both MoE layers, most domain–expert pairs exhibit relatively small changes in routing mass, while larger deviations are concentrated in only a limited number of pairs. This indicates that the domain-dependent expert-utilization patterns learned during training are largely preserved on held-out samples rather than being substantially reorganized at test time. The two layers nevertheless exhibit different stability patterns, further supporting the need to analyze their routing behavior separately.

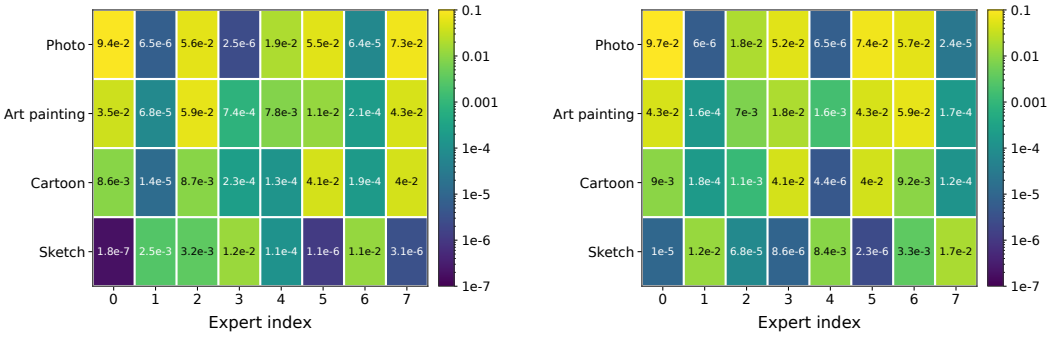


Figure 7: Domain-wise train-to-test routing-mass differences across experts on PACS. Rows correspond to domains and columns to experts. Each cell reports the change in routing mass between the training and test splits. Smaller values indicate greater routing stability. Results are shown separately for the two MoE layers.

Class-wise stability. Figure 8 provides the complementary class-by-expert view. Similar to the domain-wise analysis, the routing-mass differences are generally small for most class–expert pairs, indicating that the semantic expert-utilization patterns learned from the training data transfer consistently to held-out samples. Larger changes are restricted to a smaller subset of experts and classes, suggesting localized variation rather than a global reorganization of the routing structure.

Taken together, the domain-wise and class-wise analyses indicate that the learned routing structure is largely preserved across the training and test splits along both domain-related and semantic dimensions.

Training Dynamics of Routing-Mass Stability. The previous analysis considers routing stability at the final checkpoint. We further examine how $\mathrm{TV}_{d,\ell}^{\mathrm{mass}}$ evolves throughout training. Figure 9 reports the per-domain mass total variation at MoE Layers 0 and 1 across training epochs on PACS. The routing discrepancy is not monotonic during optimization. In particular, both layers exhibit transient increases in total variation during the middle stages of training, indicating that expert assignments continue to reorganize as the model is optimized. At Layer 0, these deviations subsequently decrease and become small toward the end of training. Layer 1 exhibits larger and more domain-dependent fluctuations, although its final routing discrepancies remain substantially below the largest intermediate-training deviations. Taken together with the final-checkpoint results in Figure 8 and Figure 7, these observations suggest that the learned routing structure becomes reasonably stable by the end of training, while deeper routing decisions can remain more sensitive to the data domain.

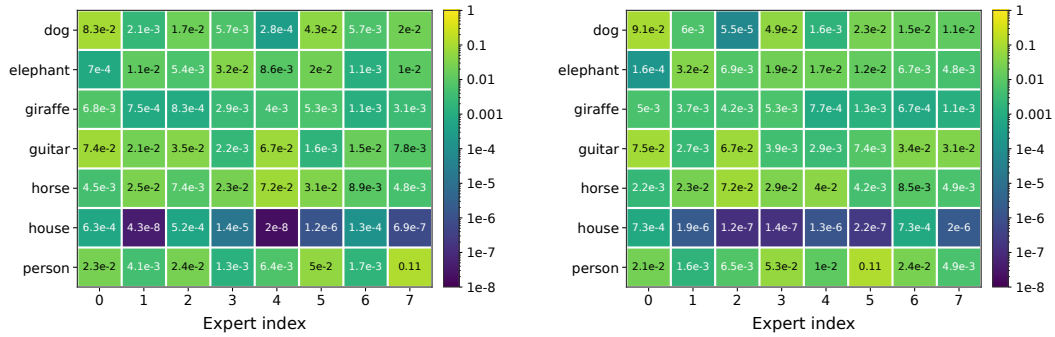


Figure 8: Class-wise train-to-test routing-mass differences across experts on PACS. Rows correspond to semantic classes and columns to experts. Each cell reports the change in routing mass between the training and test splits. Smaller values indicate greater routing stability. Results are shown separately for the two MoE layers.

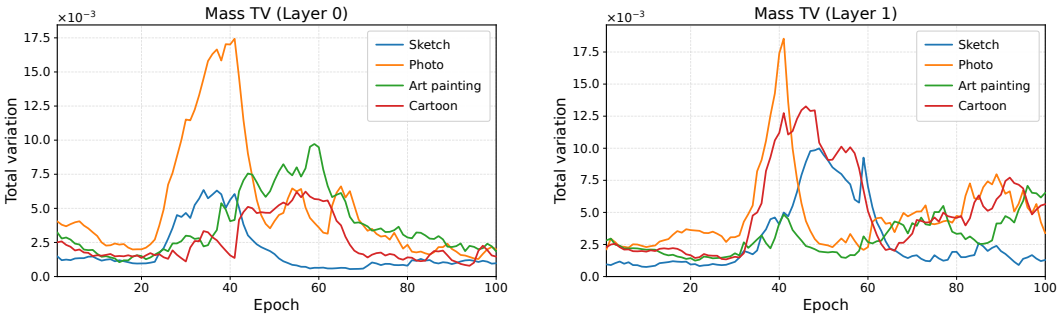


Figure 9: Training dynamics of the domain-conditioned routing-mass total variation at MoE Layers 0 and 1 on PACS. Lower values indicate closer agreement between the training and test routing-mass distributions.

F.3 Sensitivity to Modularization Loss Weights

We vary the loss coefficients controlling sparsity, balance, and expert diversity. Table 13 reports the sensitivity to these modularization loss weights on PACS.

Table 13: Sensitivity to modularization loss weights on PACS. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

$\lambda_{\texttt{sp}}$	$\lambda_{\texttt{bal}}$	$\lambda_{\texttt{div}}$	**TA**↑	**RA**↑	**FA**↓
0	1	1	90.39	99.14	8.18
1	0	1	92.19	97.29	2.34
1	1	0	91.49	99.30	1.88
0.1	1	1	92.49	99.43	0.43
1	0.1	1	91.89	99.62	0.72
1	1	0.1	90.89	98.15	9.99
1	1	1	91.39	99.74	1.67

G Computational Cost Analysis

We decompose unlearning cost into forward cost, backward cost, optimizer-state memory, and number of updated parameters. As shown in Table 14, ModULE updates substantially fewer parameters

and reduces backward-pass cost, optimizer-state memory, and peak VRAM compared with the global-update baselines.

Table 14: Computational cost decomposition during unlearning. The lowest cost in each metric is highlighted in green.

Method	Updated Params. (M)	Forward Time (ms)	Backward Time (ms)	Optimizer Memory (MB)	Peak VRAM (GB)
Full-model GA	29.15	457.5	508.6	222.4	11.87
SalUn	14.58	457.2	509.1	222.4	11.88
Standard MoE global	9.46	456.8	267.2	72.1	4.32
ModULE	**2.37**	456.8	**234.1**	**18.1**	**2.97**

H Full Unlearning Results

We report the complete unlearning results on the remaining datasets in this section. Table 15 reports class- and domain-level unlearning results on OfficeHome, while Table 16 reports class-level unlearning results on CIFAR-100 and TinyImageNet.

Table 15: Class and domain unlearning on OfficeHome. Results are macro-averaged over seven independent class-deletion requests and four independent domain-deletion requests, respectively. We report mean ± standard deviation across 3 seeds after averaging over deletion targets within each seed. All MoE-based methods share the same architecture, pre-unlearning checkpoint, expert-update budget, forgetting objective, and optimization budget. Green, blue, and yellow indicate the best, second-best, and third-best reported means among unlearning methods under the indicated metric directions, excluding MIA. Reference models are excluded from ranking; tied means share a rank.

Category	Method	Class Unlearning				Domain Unlearning			
		FA ↓	RA ↑	TA ↑	MIA ↓	FA ↓	RA ↑	TA ↑	MIA ↓
Reference models	Original model	99.24 ± 0.18	98.80 ± 0.21	69.49 ± 0.57	0.64 ± 0.013	98.26 ± 0.29	99.12 ± 0.16	70.86 ± 0.62	0.64 ± 0.011
	Retraining	7.29 ± 0.84	98.31 ± 0.26	69.72 ± 0.64	0.51 ± 0.010	35.55 ± 1.83	99.00 ± 0.19	68.75 ± 0.71	0.52 ± 0.014
Full-model unlearning	Fine-Tuning	99.11 ± 0.23	99.37 ± 0.12	71.96 ± 0.48	0.65 ± 0.012	98.57 ± 0.25	99.62 ± 0.08	72.01 ± 0.55	0.64 ± 0.010
	Gradient Ascent	1.65 ± 0.54	12.86 ± 1.92	8.09 ± 1.17	0.56 ± 0.019	1.26 ± 0.38	2.26 ± 0.61	1.94 ± 0.47	0.59 ± 0.022
	Random Labeling	46.23 ± 2.46	98.12 ± 0.31	68.18 ± 0.73	0.61 ± 0.016	94.99 ± 0.84	97.11 ± 0.43	66.02 ± 0.89	0.64 ± 0.014
	SCRUB	89.46 ± 1.57	98.30 ± 0.28	67.80 ± 0.81	0.57 ± 0.015	25.31 ± 2.08	85.07 ± 1.36	57.04 ± 1.12	0.55 ± 0.018
	SG-Unlearning	97.72 ± 0.41	99.31 ± 0.14	70.65 ± 0.53	0.63 ± 0.011	98.51 ± 0.27	99.60 ± 0.09	72.10 ± 0.61	0.64 ± 0.013
	Boundary Expanding	37.11 ± 2.19	98.13 ± 0.34	68.88 ± 0.66	0.60 ± 0.017	92.33 ± 1.06	95.73 ± 0.68	63.47 ± 0.94	0.65 ± 0.015
	Boundary Shrink	6.07 ± 1.03	88.83 ± 1.24	57.32 ± 1.09	0.58 ± 0.018	7.55 ± 1.21	14.75 ± 2.13	7.66 ± 1.28	0.60 ± 0.021
Parameter-localized unlearning	SPM	10.34 ± 1.42	72.38 ± 1.78	54.24 ± 1.31	0.50 ± 0.009	66.65 ± 2.27	70.96 ± 1.92	54.75 ± 1.46	0.56 ± 0.017
	SSD	96.96 ± 0.58	98.81 ± 0.20	69.72 ± 0.49	0.63 ± 0.014	97.94 ± 0.36	98.92 ± 0.18	70.51 ± 0.58	0.64 ± 0.012
	ℓ_1-sparse	1.65 ± 0.49	12.80 ± 1.84	8.09 ± 1.11	0.56 ± 0.017	1.26 ± 0.42	2.26 ± 0.57	1.94 ± 0.44	0.59 ± 0.020
	SalUn	30.01 ± 1.96	98.96 ± 0.17	70.49 ± 0.62	0.61 ± 0.015	97.17 ± 0.47	98.92 ± 0.22	70.51 ± 0.65	0.66 ± 0.014
MoE-based unlearning	MoE (1)	47.53 ± 2.38	93.89 ± 0.76	60.78 ± 0.92	0.51 ± 0.012	50.73 ± 2.16	90.36 ± 0.98	63.22 ± 0.83	0.56 ± 0.016
	GRIP	6.21 ± 0.97	62.20 ± 2.04	41.53 ± 1.53	0.48 ± 0.013	8.35 ± 1.14	71.92 ± 1.67	45.16 ± 1.29	0.52 ± 0.011
	SEUF	1.25 ± 0.43	49.45 ± 2.31	29.43 ± 1.72	0.52 ± 0.014	9.30 ± 1.36	52.34 ± 2.09	33.01 ± 1.58	0.56 ± 0.019
	RepSelect	71.33 ± 2.12	91.79 ± 0.91	60.25 ± 1.06	0.62 ± 0.018	50.17 ± 2.43	69.12 ± 1.85	44.19 ± 1.41	0.60 ± 0.017
	ModULE	**2.04 ± 1.74**	**93.73 ± 0.82**	**63.02 ± 0.87**	**0.56 ± 0.015**	**6.27 ± 1.08**	**97.15 ± 0.39**	**67.43 ± 0.76**	**0.62 ± 0.018**

Table 16: Comparison under class unlearning on CIFAR-100 and TinyImageNet. Results are macro-averaged over seven independent class-deletion requests and four independent domain-deletion requests, respectively. We report mean ± standard deviation across 3 seeds after averaging over deletion targets within each seed. All MoE-based methods share the same architecture, pre-unlearning checkpoint, expert-update budget, forgetting objective, and optimization budget. Green, blue, and yellow indicate the best, second-best, and third-best reported means among unlearning methods under the indicated metric directions, excluding MIA. Reference models are excluded from ranking; tied means share a rank.

Category	Method	CIFAR-100				TinyImageNet			
		FA ↓	RA ↑	TA ↑	MIA ↓	FA ↓	RA ↑	TA ↑	MIA ↓
Reference models	Original model	98.60 ± 0.24	98.84 ± 0.17	76.65 ± 0.38	0.64 ± 0.012	98.64 ± 0.31	98.24 ± 0.22	61.86 ± 0.56	0.68 ± 0.015
	Retraining	4.15 ± 0.83	99.12 ± 0.14	76.61 ± 0.46	0.51 ± 0.009	12.85 ± 1.47	98.12 ± 0.28	60.07 ± 0.63	0.59 ± 0.018
Full-model unlearning	Fine-Tuning	96.95 ± 0.52	99.92 ± 0.03	78.82 ± 0.34	0.66 ± 0.011	97.18 ± 0.64	99.94 ± 0.02	64.26 ± 0.48	0.69 ± 0.013
	Gradient Ascent	6.85 ± 1.16	19.94 ± 2.37	14.92 ± 1.68	0.54 ± 0.017	3.12 ± 0.79	8.95 ± 1.84	5.11 ± 0.96	0.55 ± 0.021
	Random Labeling	18.89 ± 1.72	97.59 ± 0.36	73.91 ± 0.61	0.58 ± 0.014	41.40 ± 2.35	94.29 ± 0.73	57.81 ± 0.89	0.64 ± 0.019
	SCRUB	19.62 ± 1.43	96.04 ± 0.58	72.16 ± 0.74	0.55 ± 0.012	7.79 ± 1.08	84.07 ± 1.26	51.45 ± 1.04	0.52 ± 0.016
	SG-Unlearning	98.60 ± 0.24	99.49 ± 0.09	77.98 ± 0.41	0.64 ± 0.012	98.49 ± 0.27	99.54 ± 0.11	63.42 ± 0.52	0.69 ± 0.014
	Boundary Expanding	13.69 ± 1.28	96.63 ± 0.47	72.10 ± 0.68	0.56 ± 0.016	27.67 ± 2.06	86.18 ± 1.14	49.79 ± 1.21	0.62 ± 0.020
	Boundary Shrink	11.42 ± 1.05	97.49 ± 0.32	74.71 ± 0.57	0.59 ± 0.013	1.05 ± 0.36	51.70 ± 2.48	29.55 ± 1.76	0.56 ± 0.018
Parameter-localized unlearning	SPM	14.03 ± 1.61	54.89 ± 2.13	43.64 ± 1.49	0.50 ± 0.008	16.04 ± 1.92	44.43 ± 2.64	33.01 ± 1.83	0.50 ± 0.011
	SSD	63.32 ± 2.24	97.80 ± 0.29	76.14 ± 0.43	0.63 ± 0.015	80.29 ± 1.86	98.25 ± 0.24	62.15 ± 0.59	0.65 ± 0.017
	ℓ_1-sparse	6.80 ± 1.09	19.83 ± 2.16	14.88 ± 1.57	0.54 ± 0.016	1.02 ± 0.41	8.12 ± 1.63	5.00 ± 0.87	0.55 ± 0.019
	SalUn	9.80 ± 0.94	98.96 ± 0.18	76.74 ± 0.36	0.57 ± 0.011	32.41 ± 2.17	98.46 ± 0.21	62.91 ± 0.67	0.65 ± 0.016
MoE-based unlearning	MoE (1)	17.16 ± 1.37	90.07 ± 0.82	72.84 ± 0.65	0.53 ± 0.010	21.64 ± 1.74	95.66 ± 0.49	65.31 ± 0.54	0.52 ± 0.013
	GRIP	10.00 ± 1.12	85.13 ± 1.19	62.22 ± 0.93	0.58 ± 0.018	6.75 ± 0.86	62.97 ± 1.95	37.18 ± 1.38	0.51 ± 0.012
	SEUF	8.17 ± 0.88	73.58 ± 1.56	52.80 ± 1.17	0.50 ± 0.009	1.38 ± 0.47	68.05 ± 1.72	41.96 ± 1.29	0.52 ± 0.014
	RepSelect	9.30 ± 1.21	38.72 ± 2.41	32.41 ± 1.62	0.52 ± 0.013	8.86 ± 1.34	35.08 ± 2.23	25.55 ± 1.58	0.54 ± 0.017
	ModULE	**7.42 ± 1.03**	**99.34 ± 0.13**	**77.75 ± 0.45**	**0.54 ± 0.014**	**7.57 ± 1.18**	**98.16 ± 0.26**	**65.66 ± 0.62**	**0.49 ± 0.012**