MITIGATING ROUTING MISMATCH IN MULTIMODAL KNOWLEDGE EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal knowledge editing aims to update specific knowledge in multimodal large language models (MLLMs) without full retraining. We identify that in methods requiring editing parameter selection, the gap between teacher-forcing training and autoregressive generation is mainly caused by routing mismatch: optimal parameters selected in training are not reliably activated during generation. To bridge this gap, we propose a generation-aligned optimization framework that formulates knowledge editing as a sequential decision process. We simulate generation scenarios to allow the model to perform parameter routing and optimization on prefixes it may encounter during generation. To prevent gradient corruption caused by erroneous prefixes, we further propose a Generation-Aligned Prefix (GAP) loss that selectively propagates gradients only along valid tokens. Experiments on VLKEB and ComprehendEdit demonstrate that our method significantly improves generation accuracy under both single-edit and lifelong-edit settings.

1 INTRODUCTION

Recent advances in multimodal large language models (MLLMs) have enabled powerful reasoning across vision and language (Roumeliotis & Tselikas, 2023; Li et al., 2024; Bai et al., 2023). As these models are increasingly deployed in dynamic environments, multimodal lifelong knowledge editing has emerged as a critical capability, allowing models to efficiently update or correct specific pieces of knowledge without full retraining (Sinitsin et al., 2020; De Cao et al., 2021). Prior works have proposed various editing techniques to localize and achieve knowledge edits (Yao et al., 2023; Hartvigsen et al., 2023; Yu et al., 2024; Pan et al., 2024; Chen et al., 2025; Li et al., 2025a; Guo et al., 2025).

Despite high editing accuracy under teacher forcing (TF), edited models can fail to produce the intended answers during autoregressive generation (AG) (Yang et al., 2025). This gap is commonly associated with *exposure bias*: training conditions on ground-truth prefixes, whereas generation conditions on the model's own predictions, allowing errors to accumulate (Arora et al., 2022). For parameter-routing-based editing methods, however, the discrepancy also concerns *which edited parameters are used to produce each token*. TF selects editing parameters using an answer-dependent representation and reuses them across answer positions, whereas AG selects parameters from the prefix available at each decoding step. The parameters optimized to express an edit may therefore be bypassed at inference, even before an incorrect token has been generated. We refer to this discrepancy as **routing mismatch**. As illustrated in Fig. 1 (a), routing mismatch can initiate a prediction error, causing the subsequent generation to deviate from the ground-truth trajectory. This deviation can in turn affect parameter routing at subsequent steps.

To examine this mechanism, we vary prefix conditioning and expert assignment in ELDER (Li et al., 2025a). Figure 1 (b) shows that retaining the expert assignment used during editing largely preserves performance when ground-truth prefixes are replaced with generated prefixes. Conversely, switching to dynamic routing substantially reduces performance even when ground-truth prefixes are retained. This diagnostic indicates that supplying the correct prefix alone is insufficient when the model activates a different set of edited parameters. A complementary experiment with BalancEdit on Qwen3-VL shows a similar pattern (Appendix A). These results identify routing mismatch as a major source of errors in the evaluated settings and motivate aligning parameter selection during editing with the information available during generation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

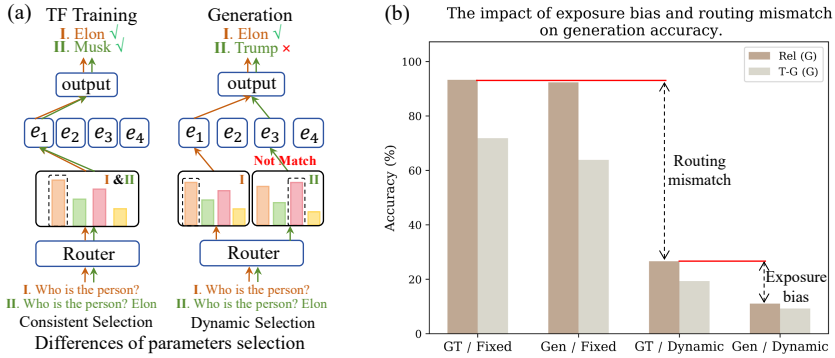

Figure 1: (a) The differences in parameter selection between teacher forcing (TF) training and the actual generation. If the optimal parameters are not activated, incorrect predictions would occur. (b) Analysis of exposure bias and routing mismatch, where GT/Gen denote Ground Truth and generated prefix, and Fixed/Dynamic denote fixed expert assignment and step-wise dynamic expert selection, respectively. The experiments are conducted on the Qwen2.5-VL using the VLKEB benchmark.

We therefore propose a **generation-aligned optimization framework** that exposes the editing modules to prefix-dependent routing during training. Starting from the image-question prompt, the model generates a response autoregressively and recomputes routing from the available prefix at each step. Training then supervises next-token prediction under these prefix-conditioned parameter selections, instead of relying exclusively on an assignment derived from the complete target answer. This approach directly addresses the discrepancy between the parameters optimized during editing and those activated during generation. It also raises a supervision problem: once a generated token differs from the reference, subsequent correct tokens' probability collapses, yielding unstable gradients. Applying supervision indiscriminately to the entire rollout can therefore corrupt acquired knowledge and result in performance degradation.

To address this problem, we introduce the **Generation-Aligned Prefix (GAP) loss**. GAP supervises the ground-truth next token only at positions where the preceding generated prefix still matches the reference. It includes the first erroneous prediction, providing a corrective update at the point of divergence, and excludes subsequent positions conditioned on an already incorrect prefix. Combined with the conventional editing loss, GAP trains prefix-conditioned routing while avoiding supervision on the mismatched suffix.

We additionally address a separate source of instability: the decision of whether an input should activate the editing module at all. Gates based on evolving decoder representations may change this decision as generation proceeds. Our **Semantic-Anchor Gating** instead uses frozen image-text representations of the input to make the activation decision before decoding. This keeps the outer activation decision fixed throughout the response, while the internal router continues to select editing parameters from the current prefix.

Experiments on VLKEB and ComprehendEdit across three MLLM backbones show substantial improvements in generation accuracy. Extensive ablation studies further demonstrate that generation-aligned optimization is crucial for mitigating the performance discrepancy. Applying GAP loss to ELDER and BalancEdit also improves their generation performance, supporting its usefulness beyond the complete proposed framework.

Our contributions are threefold:

We investigate routing mismatch in parameter-routing-based multimodal editors and show that changing parameter selection can substantially degrade prediction even under ground-truth prefix conditioning.

We introduce a generation-aligned editing framework with GAP loss for selective supervision under prefix-conditioned routing, together with Semantic-Anchor Gating for stable module activation.

We evaluate the framework on two multimodal editing benchmarks across three backbones and assess its components and the transfer of GAP to existing editors.

2 Related Work

2.1 Knowledge Editing

Knowledge editing methods can be divided into two categories based on their parameter selection strategy during inference: 1. **Static Inference**: In these methods, the editing parameters selected during generation remain unchanged. To mitigate prediction deviations on unrelated inputs, they typically train a gate or incorporate regularization losses during training. Representative methods include: ROME (Meng et al., 2022a), KN (Dai et al., 2021), and AlphaEdit (Fang et al., 2024), which identify and update critical parameters through input-conditioned localization; MEND (Mitchell et al., 2021), MEMIT (Meng et al., 2022b), and RLEdit (Li et al., 2025b), which dynamically generate layer-specific parameter modifications conditioned on the input; SERAC (Mitchell et al., 2022), which determines whether to employ the additionally updated language model by computing semantic similarity between the input and cached reference texts; and Merging (Fu et al., 2025), which directly merges the updated weights with the original model parameters. 2. **Dynamic Inference**: In these methods, the editing parameters selected may change dynamically during generation. These methods dynamically select a subset of trainable parameters during generation based on the input and the generated tokens. Therefore, the token representations used for routing continuously change during generation. Methods include: WISE (Wang et al., 2024), which employs a trainable gate on the intermediate layer representation to selectively adopt the trained module's outputs; MELO (Yu et al., 2024), GRACE (Hartvigsen et al., 2023) and BalancEdit (Guo et al., 2025), which maintain codebook-structured storage of edited sample representations paired with their corresponding trained parameters, subsequently determining parameter usage through representation distance metrics at intermediate layers; and ELDER (Li et al., 2025a), which leverages a router to dynamically select experts.

2.2 Reinforcement Learning

Reinforcement learning (RL) has long been used to address the performance discrepancy between training and generation in autoregressive sequence models. Early work on sequence-level training observed that maximum likelihood estimation with teacher forcing optimizes token-level likelihood under ground-truth prefixes (Ranzato et al., 2015), while at test time models must condition on their own predictions, leading to exposure bias and compounding errors (Rennie et al., 2017). Policy gradient methods such as REINFORCE (Williams, 1992) and SCST (Rennie et al., 2017) mitigate this issue by optimizing sequence-level rewards under outputs generated by the model itself, thereby aligning the training objective with the generation process. These ideas have been widely adopted in sequence generation tasks, including machine translation, image captioning, and multimodal generation (Ranzato et al., 2015; Wu et al., 2016; Rennie et al., 2017).

In contrast, reinforcement learning has been largely underexplored in knowledge editing. RLEdit (Li et al., 2025b) is the first to introduce reinforcement learning into knowledge editing. It treats the parameter updates generated by an auxiliary network as actions, and the current model parameters, inputs, and generated parameter updates as the state, training this network to generate appropriate updates in response to the evolving model parameters during lifelong editing. However, it still fails to reflect the test-time scenario, leading to poor performance during generation.

3 Method

In this section, we first introduce multimodal lifelong editing and the relevant reinforcement learning background. We then present our empirical observations through experiments and introduce the proposed framework and loss.

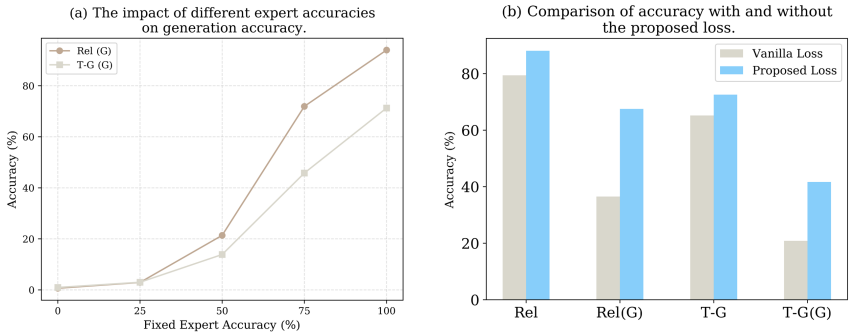

Figure 2: (a) The impact of different routing matching levels on generation accuracy. (b) Comparison of accuracy when using the vanilla autoregressive loss versus the proposed loss. The experiment is conducted on Qwen2.5-VL with VLKEB, $T = 100$.

3.1 Preliminaries

In multimodal lifelong editing, we assume a pretrained multimodal model f_0 and a sequence of data $D_T = \{(x_i, y_i)\}_{i=1}^{T}$, where x_i denotes the combination of a question and an image, and y_i denotes the corresponding answer. Multimodal lifelong editing can then be formulated as

$$f_i = \mathrm{KE}(f_{i-1}, x_i, y_i), i = 1, 2, \cdots, T, \tag{1}$$

where KE denotes the editing method.

3.2 Motivation

Dynamic inference knowledge editing methods select editing parameters according to the input representation and use the selected parameters to predict answer tokens. Under TF training, however, the routing decision can access the ground-truth answer prefix, whereas such information is unavailable during autoregressive generation. Therefore, the TF-AG discrepancy in parameter-routed editing involves two potentially different factors: *routing mismatch* at the parameter-selection stage and *exposure bias* at the sequence-generation stage.

To disentangle these two factors, we conduct single-edit experiments with ELDER, a representative dynamic inference knowledge editing method. As shown in Fig. 1 (a), when the expert assignment is fixed, replacing the ground-truth prefix with the generated prefix causes only a limited degradation in generation accuracy (GT/Fixed vs Gen/Fixed). In contrast, when the routing strategy is changed from fixed assignment to dynamic selection while keeping the ground-truth prefix unchanged, generation accuracy drops substantially (GT/Fixed vs GT/Dynamic). When both generated prefixes and dynamic routing are used, the performance is further degraded. These experiments reveal that the dominant source of the TF-AG degradation is not the change from ground-truth to generated prefixes alone, but the inconsistency in parameter selection between training and generation. Therefore, we refer to this phenomenon as **routing mismatch**: the routing mechanism uses the ground-truth prefix during TF training to identify editing parameters, while the same information is unavailable during autoregressive generation. To further verify the role of routing consistency, we vary the degree of expert matching between training and generation. As shown in Fig. 2 (a), both Rel(G) and T-G(G) increase consistently with expert matching accuracy. These results provide further evidence that consistent parameter activation is closely related to successful generation of the edited knowledge.

To align parameter routing between TF training and generation, a natural solution is to train the model to dynamically select parameters for each token based on the generated prefix, and to compute the Cross Entropy loss along these tokens, which we refer to as the vanilla autoregressive loss. However, directly optimizing autoregressive loss introduces a fundamental challenge: once an intermediate token is incorrectly predicted, the probability of generating the correct subsequent token diminishes, resulting in a sharp increase in the loss. And if we forcibly compute the loss and update the model for all subsequent tokens conditioned on incorrect prefixes, it will severely degrade the model's acquired knowledge. As demonstrated in Fig. 2 (b), for lifelong editing with sequence length $T = 100$, compared to the proposed loss, employing the vanilla autoregressive loss results

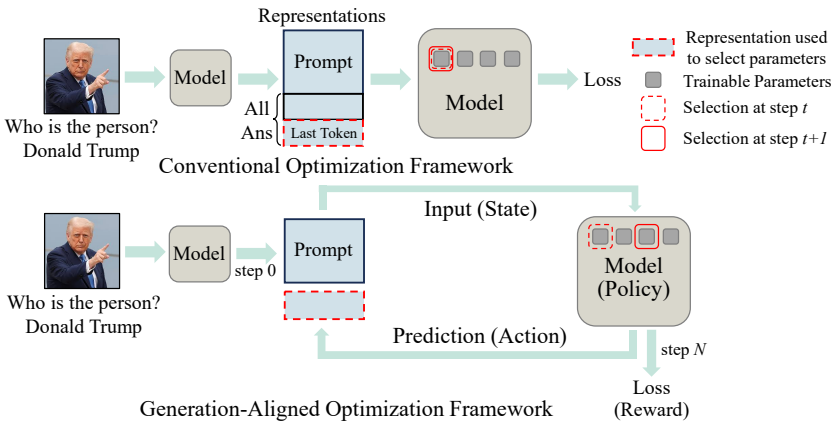

Figure 3: Comparison between the proposed framework and the conventional optimization framework. Conventional methods employ a fixed parameter configuration for calculating loss on all answer tokens, while our proposed framework performs dynamic parameter selection during training.

in degraded generation accuracy and compromised model reliability. Furthermore, this vanilla loss precludes parallel computation, thereby significantly increasing training costs. We present the formulation of the proposed loss in the following section.

3.3 Generation-Aligned Optimization Framework

To mitigate the routing mismatch between training and generation, we propose a generation-aligned optimization framework that explicitly models multimodal knowledge editing as a sequential decision-making process. As shown in Fig. 3, unlike conventional approaches that optimize token probability on the same parameters, our framework optimizes the expected reward of tokens generated by the model itself, thereby aligning training dynamics with autoregressive decoding behavior.

Formally, given an editing sample consisting of image–question tokens s_0 and a router $\mathcal{R}_\theta$, we treat autoregressive decoding as a trajectory $\tau = (a_1, \cdots, a_{N_\tau})$, where each token is sampled from the current policy π_θ:

$$P(\tau;\theta) = \prod_t \pi_\theta(a_{t+1}|s_t, \mathcal{R}_\theta(s_t)), s_{t+1} = \text{concat}(s_t, a_{t+1}). \tag{2}$$

Here, the policy π_θ consists of the backbone model equipped with trainable editing modules. Inspired by the REINFORCE algorithm with a single Monte Carlo sample (Rennie et al., 2017), the objective is to increase the probability of actions taken along high-reward trajectories $\hat{\tau} = (\hat{a}_1, \cdots, \hat{a}_{N_\tau})$:

$$\begin{aligned} \mathcal{L}(\theta) &= -R(\hat{\tau}) \cdot \frac{1}{N_{\hat{\tau}}} \log P(\hat{\tau};\theta), \\ &= -R(\hat{\tau}) \frac{1}{N_{\hat{\tau}}} \sum_{t=0}^{N_{\hat{\tau}}} \log \pi_\theta(\hat{a}_{t+1}|\hat{s}_t, \mathcal{R}_\theta(\hat{s}_t)). \end{aligned} \tag{3}$$

Within this framework, conventional teacher-forcing Cross-Entropy (CE) loss can be viewed as a special case where the routing decision is $\mathcal{R}_\theta(\hat{s}_{N_{\hat{\tau}}})$ and the reward is $R(\hat{\tau}) = 1$. **Importantly, in TF training, the routing decision is computed from the full ground-truth sequence $\hat{s}_{N_\tau}$. This does not align with the routing in generation, i.e., $\mathcal{R}_\theta(\hat{s}_{N_\tau}) \neq \mathcal{R}_\theta(\hat{s}_t)$.** By contrast, our framework optimizes the policy under generated prefixes, ensuring that parameter routing decisions during training reflect those encountered during actual decoding.

3.4 GENERATION-ALIGNED PREFIX LOSS (GAP LOSS)

While trajectory-level optimization aligns training with generation, directly optimizing under model-generated prefixes introduces instability. Once an intermediate token is predicted incorrectly, subsequent decoding proceeds through off-distribution prefixes where enforcing ground-truth supervision may produce large gradients and corrupt previously acquired knowledge.

To address this issue, we introduce a prefix-conditioned reward that selectively propagates gradients only along valid decoding tokens. We adopt a truncated return formulation G_t, where rewards are only assigned to the first incorrectly generated token and all preceding tokens:

$$G_t = \begin{cases} \frac{1}{L_{\text{valid}}} & t \leq L_{\text{valid}} = \min\{i : a_{i+1} \neq \hat{a}_{i+1}\}, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where L_{valid} is the position of the first incorrectly predicted token. By integrating the gradient of the Cross-Entropy (CE) loss with that of this generation-aligned prefix loss, we obtain

$$\begin{aligned} \mathcal{L}(\theta) &= \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{GAP}} \\ &= -\sum_{t=0}^{N_{\hat{\tau}}} [\frac{1}{N_{\hat{\tau}}} \log \pi_\theta(\hat{a}_{t+1} | \hat{s}_t, \mathcal{R}_\theta(\hat{s}_{N_{\hat{\tau}}})) + \lambda G_t \log \pi_\theta(\hat{a}_{t+1} | \hat{s}_t, \mathcal{R}_\theta(\hat{s}_t))], \end{aligned} \tag{5}$$

where λ denotes the weight of the GAP loss. The computation of the GAP loss consists of three steps: (1) generate a response conditioned on the given question; (2) identify L_{valid}; and (3) compute the policy loss for all tokens up to and including L_{valid}, with each token loss weighted by its corresponding G_t. The key difference lies in that routing is computed from ground-truth full sequences in TF CE, but conditioned on generated prefixes in GAP.

In practice, computing the GAP loss requires autoregressive token generation to identify valid loss positions. To improve computational efficiency, we construct multiple candidate sequences $\{\hat{s}_0, \hat{s}_1, \cdots, \hat{s}_{N_{\hat{\tau}}}\}$, and consolidate them into a unified batch, thereby simulating the prefix conditions encountered during inference. In this way, multiple sequences can be processed in parallel during a single forward pass. During inference, we adopt greedy decoding to generate responses and compute the accuracy.

3.5 SEMANTIC-ANCHOR GATING

Most existing knowledge editing methods employ various gating mechanisms to determine whether the model's output should be modified. For example, SERAC interpolates between the original model and a counterfactual edit model according to the retrieval score of the closest edited instance; ELDER evaluates routing consistency by comparing the experts activated by the input with those associated with stored edits; WISE learns to distinguish edited samples from locality samples by modeling differences in their intermediate-layer representation distributions; and MELO, GRACE, and BalanceEdit compare the distance between the input's intermediate representation and cached representations against predefined thresholds.

However, during autoregressive decoding, the representation of the latest token continuously evolves as new tokens are generated. To examine the impact of this dynamic shift, we conduct a single-edit experiment on Qwen2.5-VL using the VLKEB dataset with ELDER. For each generated token, we record (i) the number of experts that match those selected during training and (ii) the Euclidean distance between the representations used for expert retrieval. With 8 experts in total and 4 experts selected for updating, the average token-representation distances corresponding to 0–4 matched experts are [108.45, 109.94, 106.42, 86.97, 38.96], respectively. The proportion of tokens observed at each match level is $[0.70\%, 15.59\%, 45.35\%, 31.74\%, 6.61\%]$. Except for the first and last token position, where the number of samples is too small to be statistically meaningful, **the results indicate a clear trend: the larger the representation drift during generation, the lower the overlap between the selected experts and those used during training.** Consequently, these gating mechanisms that rely on intermediate representations may struggle to maintain consistent activation decisions throughout the generation process.

To address these issues, we propose a semantically consistent gating mechanism based on frozen multimodal pretrained representations. Specifically, we extract joint image-text embeddings using a

frozen model such as CLIP (Radford et al., 2021) and measure cosine similarity between the input and cached representations of edit and locality samples in this semantic space. If the nearest neighbor corresponds to an edited sample, the inserted MoE module is activated; otherwise, the original model output is preserved. Therefore, Semantic-Anchor Gating is used to determine whether the input is editing-related. If it is, the inserted MoE module is applied during the generation process. By decoupling the gating decision from autoregressive hidden states, this lightweight activation mechanism is robust to representation shifts during decoding.

4 EXPERIMENTS

Datasets and Backbones: The experiments are conducted on ComprehendEdit (Ma et al., 2025), VLKEB (Huang et al., 2024) with sequence length $T = 1, 100, 1000$, with a batch size of 1. We use recent Qwen2.5-VL (Bai et al., 2023), LLaVA-OneVision (Li et al., 2024), and Qwen3-VL as the backbones. The compared methods mainly include dynamic inference methods, i.e., WISE (24 NIPS) (Wang et al., 2024), , MELO (24 AAAI) (Yu et al., 2024), BalancEdit (25 ICML) (Guo et al., 2025), and ELDER (25 AAAI) (Li et al., 2025a). We also compare with classical static inference methods Merging (25 ACL) (Fu et al., 2025) and RLEdit (25 ICML) (Li et al., 2025b).

Implementation details: The learning rate is 1e-3 for MELO, ELDER, and Ours, 1e-4 for WISE and Merging, 1e-6 for RLEdit. We insert an MoE module into *24.mlp.down_proj* of Qwen2.5-VL 7B, *34.mlp.down_proj* of Qwen3-VL 8B, and *32.mlp.down_proj* of LLaVA-OneVision 8B. We set random seed 42 and randomly utilize $p = 10\%$ locality samples in the training set. The locality samples in the training and testing sets do not overlap. The number of experts and their rank are (4, 8) for $T = 1$, (8, 8) for $T = 100$ and (20, 16) for $T = 1000$. During generation, we set the maximum number of tokens to 32. For each sample, we update the top $k = 4$ selected experts during editing. In addition, we store the image and text representations of the locality samples and editing samples, which are extracted by CLIP (Radford et al., 2021). To measure similarity between samples, we compute the image-level and text-level similarities separately and take their average. The code will be released soon.

Evaluation Metrics: The evaluation metrics primarily include Rel (Reliability, whether the edited model correctly answers the edited question), T-G (Text Generality, whether the edited model correctly answers rephrased variants of the edited question), and Loc (Locality, whether the edited model preserves its original responses to unrelated pure-text questions, multimodal questions). For locality evaluation, the metrics include T-L, M-L, and KPI. These metrics are typically measured using the locality samples provided within the benchmark or external datasets such as NQ dataset (Kwiatkowski et al., 2019) and OKVQA (Marino et al., 2019).

In addition, we evaluate performance under free-form generation, reporting the generation accuracy of Rel and T-G, denoted as Rel (G) and T-G (G), respectively. We measure whether the generated response exactly matches the ground-truth answer. In summary, the final average accuracy is expressed as

$$\text{AVG} = \frac{1}{3}[\frac{1}{2}(\text{Rel (G)} + \text{Rel}) + \frac{1}{2}(\text{T-G (G)} + \text{T-G}) + \text{Loc}]. \tag{6}$$

4.1 RESULTS

The main results are presented in Tab. 1, and results on LLaVA-OneVision and Qwen3-VL are included in Appendix A. Due to memory constraints, the performance of RLEdit when $T = 1000$ is not shown in the table. The generation accuracy Rel (G) and T-G (G) reveal that VLKEB is more challenging, attributable to its considerably longer average answer length. With respect to specific accuracy performance, methods that train a single additional module, such as WISE and Merging, maintain high generation accuracy in single-edit scenarios. However, their performance degrades substantially under lifelong editing settings, as the persistent accumulation of knowledge within a single module induces knowledge conflicts, thereby causing catastrophic forgetting of previously edited knowledge.

In contrast, approaches that initialize multiple parameter blocks, including MELO, ELDER and BalancEdit, exhibit markedly inferior generation accuracy even in single-edit scenarios, primarily due to

Table 1: Results on Qwen2.5-VL 7B. **Bold** and underline represent the first and second highest values, respectively.

		ComprehendEdit						VLKEB					
T	Method	Rel(G)	Rel	T-G(G)	T-G	Loc	**AVG**	Rel(G)	Rel	T-G(G)	T-G	Loc	**AVG**
1	WISE	98.98	99.49	**83.10**	**86.93**	**99.12**	**94.46**	61.66	88.59	44.52	81.13	98.58	78.85
	Merging	82.37	88.71	51.07	63.11	98.54	80.39	34.51	81.79	20.34	73.67	97.61	67.59
	MELO	45.53	99.88	34.92	63.44	98.92	73.60	0.78	95.23	0.94	39.91	96.92	55.11
	ELDER	61.27	**100**	45.74	76.13	98.72	80.10	11.03	99.42	9.23	80.07	97.76	65.88
	RLEdit	42.95	56.20	33.74	47.62	98.60	62.95	0.55	41.82	0.94	44.87	**98.83**	47.64
	BalancEdit	93.55	**100**	77.77	86.60	96.28	91.75	20.27	99.99	23.79	**95.73**	91.19	70.36
	Ours	**99.70**	99.95	71.47	78.21	98.94	91.20	**100**	**100**	**83.88**	93.55	97.72	**95.48**
100	WISE	51.00	66.40	38.25	55.19	89.93	65.11	1.50	45.89	1.75	47.15	95.66	47.94
	Merging	43.54	57.71	34.77	48.94	97.14	63.21	0.64	42.10	1.09	45.31	96.48	47.02
	MELO	43.05	**98.97**	34.05	70.09	**99.03**	74.04	0.58	**99.83**	1.00	50.42	**97.93**	57.95
	ELDER	63.74	97.91	50.95	79.43	80.46	75.49	7.42	88.46	7.83	75.56	66.68	52.11
	RLEdit	42.95	57.91	33.89	49.26	96.78	62.93	1.17	51.06	1.75	52.14	96.55	49.87
	BalancEdit	85.22	94.11	**70.44**	**82.29**	73.26	79.76	17.42	92.79	18.25	**87.63**	61.39	56.48
	Ours	**91.89**	95.40	64.89	79.08	96.41	**87.35**	**77.58**	92.38	**59.00**	85.33	93.67	**83.61**
1000	WISE	43.30	40.80	34.30	29.80	87.51	53.87	0.60	0.50	1.20	1.20	91.67	31.14
	Merging	41.90	30.50	33.40	20.80	**98.03**	53.78	0.50	0.50	1.10	1.10	**97.72**	33.11
	MELO	40.90	64.00	33.40	41.90	93.45	61.18	0.50	**90.50**	1.10	38.80	87.58	51.01
	ELDER	58.80	**91.38**	48.40	**76.35**	73.07	70.18	2.20	69.51	3.00	61.59	55.67	41.27
	RLEdit	-	-	-	-	-	-	-	-	-	-	-	-
	BalancEdit	66.80	74.83	56.70	65.19	67.34	66.37	8.30	75.64	7.90	**71.18**	40.86	40.79
	Ours	**79.40**	87.03	**60.60**	74.88	88.39	**79.78**	**36.30**	73.80	**20.20**	66.10	75.87	**58.02**

the potential misalignment between the parameters selected during generation and those employed during training. Our method demonstrates superior performance compared to existing approaches, consistently preserving the capability to generate accurate answers across diverse queries even under lifelong editing scenarios. Nevertheless, owing to the exclusive reliance on edited samples during training, the generation accuracy for textual generalization remains modest. As a result, a gap persists in the generation metrics T-G (G) and Rel (G). Despite this limitation, our method still achieves superior performance compared to existing approaches in the lifelong editing scenario.

4.2 Ablation studies

In this section, unless otherwise specified, all experiments are conducted on Qwen2.5-VL 7B using the VLKEB benchmark with $T = 100$ and $k = 4$.

Table 2: Ablation study about each module. CE means Cross Entropy loss. GAP means the proposed loss.

CE	GAP	Gate	Rel (G)	Rel	T-G (G)	T-G	Loc	AVG
✓			35.42	95.48	25.00	85.68	68.97	63.25
	✓		79.50	71.37	54.58	66.85	69.42	68.52
✓		✓	35.15	95.25	24.57	85.45	93.04	71.08
✓	✓		77.95	92.55	59.55	85.30	63.81	73.83
✓	✓	✓	77.58	92.38	59.00	85.33	93.67	83.61

Module ablation: As shown in Tab 2, when employing the CE loss exclusively, the model demonstrates strong performance on the original accuracy metrics Rel and T-G. Conversely, when utilizing solely GAP loss, the model exhibits superior performance on the generation accuracy Rel (G) and T-G (G). The combination of both losses yields robust performance across both traditional and generation accuracy metrics simultaneously. These results indicate that the CE loss (GAP loss) is predominantly optimized for traditional (generation) accuracy objectives. However, the absence of a gating mechanism adversely affects locality. Gating can accurately identify locality samples based on information from previously processed samples. However, it may misjudge a small number of edited samples, affecting the editing performance. After integrating the gating mechanism, locality is significantly improved, while generation accuracy remains largely unaffected.

Time and memory comparison: Tab. 3 presents the computational requirements for editing a sequence of length $T = 100$, including training time, memory consumption, and the number of trainable parameters across various methods. MELO, ELDER and BalancEdit update only a small subset of parameters per sample and apply the same parameter set uniformly across all tokens within that

Table 3: Comparison of editing time and GPU memory usage of each method. Assuming the weight dimension of the training layer is $\mathbb{R}^{m\times n}$. The experiments are conducted on Qwen2.5-VL with ComprehendEdit.

Method	Time (s/sample)	GPU Memory (GB)	Params
WISE	4.10	22	mn
Merging	3.30	20	mn
MELO	0.70	18.5	$100(m+n)$
ELDER	1.00	18.5	$64(m+n)$
RLEdit	0.47	55	$4mn$
BalancEdit	1.05	19	$100(m+n)$
Ours(wo GAP)	0.80	19.5	$64(m+n)$
Ours	1.20	19.5	$64(m+n)$

Table 4: Verification of GAP loss on other methods. The experiments are conducted on Qwen2.5-VL with VLKEB, $T = 100$.

Method	Rel (G)	Rel	T-G (G)	T-G	Loc	AVG
ELDER	7.42	88.46	7.83	75.56	66.68	52.11
ELDER+GAP	**58.86**	84.01	**35.43**	72.50	63.43	**62.94**
Balancedit	17.42	92.79	18.25	87.63	61.39	56.48
Balancedit+GAP	**56.50**	89.98	**49.50**	86.01	59.54	**66.85**

sample, resulting in substantially reduced editing time. However, this design leads to inferior generation quality and degraded performance in long-sequence scenarios. Compared to solely using CE loss, the GAP loss in this paper incurs less computational consumption but significantly enhances the generation performance. Overall, our method achieves a more favorable trade-off, demonstrating improved memory and parameter efficiency while delivering superior editing effectiveness and generation performance.

The results of other methods combined with GAP loss: We further evaluate the effectiveness of GAP loss on methods that require dynamic routing during generation. As shown in Fig. 4, even without any explicit anti-forgetting mechanism, GAP loss substantially improves the model's generation performance, demonstrating its general applicability across different dynamic-routing methods.

Due to space limitations, we defer the sensitivity analyses of the GAP loss weight λ, the proportion of locality-based samples, random seeds, different feature extractors, and the number of selected experts k to Appendix A. Additionally, Appendix A presents a visualization of the per-token prediction accuracy obtained using the GAP loss.

5 Conclusion

We identify routing mismatch as a major source of generation failures in parameter-routing-based editing methods. To mitigate it, we introduce generation-aligned optimization with a Generation-Aligned Prefix (GAP) loss and Semantic-Anchor Gating. Extensive experiments demonstrate that our approach consistently improves generation accuracy and robustness in lifelong editing settings while maintaining locality and efficiency. Our findings highlight the importance of aligning training objectives with generation dynamics.

AI use statement

In this work, we used generative AI tools for: 1. design or provide feedback on research methodology or experiments, 2. assist with translation, 3. support qualitative and thematic data analysis, 4. interpret results. We have not used generative AI tools for: 1. help develop theoretical models or conceptual frameworks and 2. implement methods. And 1. generate synthetic data sets, 2. formulate mathematical claims, provide critical ingredients for proving mathematical claims, 3. assist in the writing of proofs, 4. propose or refine hypotheses, 5. clean and reformat datasets do not apply to this

work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, 2022.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

Qizhou Chen, Chengyu Wang, Dakan Wang, Taolin Zhang, Wangyue Li, and Xiaofeng He. Lifelong knowledge editing for vision language models with low-rank mixture-of-experts. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 9455–9466, 2025.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Zichuan Fu, Xian Wu, Guojing Li, Yingying Zhang, Yefeng Zheng, Tianshi Ming, Yejing Wang, Wanyu Wang, and Xiangyu Zhao. Model merging for knowledge editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pp. 433–443, 2025.

Dongliang Guo, Mengxuan Hu, Zihan Guan, Thomas Hartvigsen, and Sheng Li. Balancedit: Dynamically balancing the generality-locality trade-off in multi-modal model editing. *arXiv preprint arXiv:2505.01343*, 2025.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.

Han Huang, Haitian Zhong, Tao Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Vlkeb: A large vision-language model knowledge editing benchmark. 2024.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Jiaang Li, Quan Wang, Zhongnan Wang, Yongdong Zhang, and Zhendong Mao. Elder: Enhancing lifelong model editing with mixture-of-lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24440–24448, 2025a.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. *arXiv preprint arXiv:2502.05759*, 2025b.

Yaohui Ma, Xiaopeng Hong, Shizhou Zhang, Huiyun Li, Zhilin Zhu, Wei Luo, and Zhiheng Ma. Comprehendedit: A comprehensive dataset and evaluation framework for multimodal knowledge editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19323–19331, 2025.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pp. 3195–3204, 2019.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pp. 15817–15831. PMLR, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Kaihang Pan, Zhaoyu Fan, Juncheng Li, Qifan Yu, Hao Fei, Siliang Tang, Richang Hong, Hanwang Zhang, and Qianru Sun. Towards unified multimodal editing with enhanced knowledge collaboration. *Advances in Neural Information Processing Systems*, 37:110290–110314, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7008–7024, 2017.

Konstantinos I Roumeliotis and Nikolaos D Tselikas. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192, 2023.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. *arXiv preprint arXiv:2004.00345*, 2020.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. The mirage of model editing: Revisiting evaluation in the wild. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15336–15354, 2025.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*, 2023.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19449–19457, 2024.