HiEdit: Lifelong Model Editing with Hierarchical Reinforcement Learning

Yangfan Wang[1], Tianyang Sun[1], Chen Tang[4], Jie Liu[1,3], Wei Cai[2,3]*, Jingchi Jiang[1,3]*
[1]Harbin Institute of Technology, [2]Beidahuang Information Co., Ltd.
[3]State Key Laboratory of Smart Farm Technologies and Systems
[4]AI Research Center, Midea Group (Shanghai) Co., Ltd.
{yf.wang,25s103212}@stu.hit.edu.cn, travistang@foxmail.com
icaiweig@gmail.com, {jieliu,jiangjingchi}@hit.edu.cn

Abstract

Lifelong model editing (LME) aims to sequentially rectify outdated or inaccurate knowledge in deployed LLMs while minimizing side effects on unrelated inputs. However, existing approaches typically apply parameter perturbations to a static and dense set of LLM layers for all editing instances. This practice is counter-intuitive, as we hypothesize that different pieces of knowledge are stored in distinct layers of the model. Neglecting this layer-wise specificity can impede adaptability in integrating new knowledge and result in catastrophic forgetting for both general and previously edited knowledge. To address this, we propose **HiEdit**, a hierarchical reinforcement learning framework that adaptively identifies the most knowledge-relevant layers for each editing instance. By enabling dynamic, instance-aware layer selection and incorporating an intrinsic reward for sparsity, HiEdit achieves precise, localized updates. Experiments on various LLMs show that HiEdit boosts the performance of the competitive RLEdit by an average of 8.48% with perturbing only half of the layers per edit. Our code is available at: https://github.com/yangfanww/hiedit.

1 Introduction

Large language models (LLMs) acquire vast knowledge through extensive training on pre-training corpora (Petroni et al., 2019; Brown et al., 2020). However, the knowledge stored in deployed LLMs inevitably becomes outdated or inaccurate over time (De Cao et al., 2021; Lazaridou et al., 2021). Given the actual need for continuous knowledge updates, retraining LLMs from scratch is both costly and impractical. Lifelong model editing (LME) has emerged as a promising solution, aiming to continuously perform targeted updates on deployed LLMs without degrading performance on general or previously edited knowledge (Hartvigsen et al., 2023). Recent advances in LME typically follow a "locating-then-editing" paradigm (Meng et al., 2022), which involves first identifying influential parameters $\mathcal{W}$, and then editing them by introducing perturbations $\tilde{\nabla}_{\mathcal{W}}$. Current methods can be categorized into two primary approaches: one (Meng et al., 2022, 2023; Fang et al., 2025) calculates closed-form solutions through repetitive matrix operations for parameter updates, which can be costly and error-prone; the other (Mitchell et al., 2022; Tan et al., 2024; Li et al., 2025) employs hypernetworks to generate parameter updates efficiently, leveraging low-rank decomposition theory of gradients and parameters. Notably, RLEdit (Li et al., 2025) pioneers the framing of LME as a reinforcement learning (RL) task, treating parameter updates as actions to capture the long-range dependencies of editing trajectories, enabling lifelong model editing across

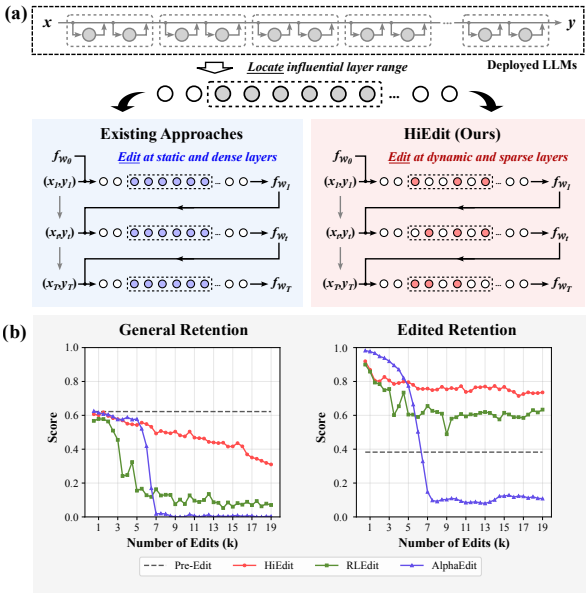


Figure 1: (a) provides a comparison between HiEdit and existing LME approaches. (b) illustrates the ability of various methods to retain general and previously edited knowledge during long-range sequential editing.

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 39924–39942

tens of thousands of knowledge instances.

However, a critical limitation persists in existing approaches: *they perturb parameters at static and dense LLM layers for all knowledge instances*. Recent research indicates that distinct knowledge activates different components[1] within LLMs, with only a small subset associated with specific knowledge (Yao et al., 2024; Wang et al., 2024). Consequently, applying perturbations to static and dense layers indiscriminately leads to the over-modification of parameters, impeding the precise integration of new knowledge and causing the forgetting of existing information. Theoretically, enforcing updates on irrelevant layers unnecessarily constrains the optimization landscape of hypernetworks, resulting in suboptimal solutions. Experimentally, this lack of adaptive layer selection manifests as severe catastrophic forgetting. As illustrated in Figure 1(b), when performing 20,000 sequential edits from the ZsRE dataset on Llama-3-8B model, the ability of the LLM to retain both general and previously edited knowledge begins to decline significantly after just 5,000 sequential edits using existing LME methods.

To address these limitations, we propose HiEdit, a novel solution using hierarchical reinforcement learning (HRL) to facilitate adaptive and localized LME. We frame LME as a hierarchical decision-making process, explicitly decoupling the complex editing task into two subtasks: layer selection (where to edit) and parameter updating (how to edit). This hierarchical structure transforms the flat and intractable action space into a manageable structured framework, allowing the editor to dynamically pinpoint appropriate components for different knowledge. Specifically, the high-level hypernetwork operates as a manager, leveraging gradient signals to adaptively output a layer importance distribution. The low-level hypernetwork then acts as a worker, generating effective parameter updates for the selected components. To enable end-to-end training across discrete decisions, we bridge the high-level and low-level policies to ensure efficient gradient flow. Furthermore, we introduce an intrinsic reward mechanism that quantifies the trade-off between partial-layer and full-layer updates, incentivizing the model to achieve editing goals with minimal parameter modifications. Unlike existing approaches that perturb entire static layers (Figure 1(a)), HiEdit dynamically identifies and modifies only the most knowledge-relevant layers, maximizing editing efficiency while minimizing disruption.

Our main contributions are as follows:

- We propose HiEdit, a novel framework for lifelong model editing, which utilizes hierarchical reinforcement learning to restructure the limited and complex action space of parameter updates into a structured hierarchical framework, achieving adaptive and localized knowledge updates.
- We introduce an intrinsic reward mechanism based on the relative advantage of partial-layer versus full-layer updates, which encourages sparse and efficient layer selection, reducing unnecessary parameter modifications while preserving editing performance.
- To the best of our knowledge, we are the first to explore a challenging experimental setup involving timely and long-range sequential editing. Empirical results show that HiEdit improves the performance of the competitive RLEdit by an average of 8.48% with perturbing only half of the layers per edit.

2 Related Works

Lifelong Model Editing (LME). LME extends model editing to sequential scenarios, enabling hundreds to thousands of edits without compromising general performance or prior modifications. Traditional methods follow a "locating-then-editing" paradigm, identifying influential layers through causal tracing or search methods, then applying parameter perturbations. Approaches like GRACE (Hartvigsen et al., 2023), RECT (Gu et al., 2024), PRUNE (Ma et al., 2024), AlphaEdit (Fang et al., 2025), and RLEdit (Li et al., 2025) offer various strategies for preserving knowledge during sequential editing. Unlike existing methods, HiEdit does not treat layer selection and layer update as isolated stages. Instead, it considers layer selection as a learnable high-level action, allowing for the dynamic selection of LLM layers for updates based on varying knowledge. Detailed content is available in Appendix A.

Hierarchical Reinforcement Learning (HRL). HRL enhances exploration and learning efficiency

[1]For ease of understanding, we use the concepts of "component", "layer module", and "layer" interchangeably.

through hierarchical structures, addressing complex action spaces. Methods are mainly categorized into option-based and subgoal-based approaches. Option-based HRL uses temporally extended actions, or options, to simplify decision-making through multi-step operations and task decomposition (Sutton et al., 1999), with studies focusing on autonomous option learning (Bacon et al., 2017; Harb et al., 2018). Subgoal-based HRL defines intermediate objectives to guide learning, traditionally constrained by manual sub-goals (Tessler et al., 2017), but recent work enables autonomous subgoal discovery (Nachum et al., 2018; Jiang et al., 2025). Our approach aligns with option-based HRL, decomposing layer selection and layer updating into subtasks managed by high-level and low-level hypernetworks, facilitating efficient exploration and learning.

3 Preliminary

3.1 Hypernetwork-based Model Editing

HiEdit belongs to the hypernetwork-based model editing approaches, which involve training hypernetworks to generate parameter updates for language models. These methods utilize a set of small auxiliary editing networks $\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_L\}$ to transform gradients $\{\nabla_{\mathcal{W}_1}, \nabla_{\mathcal{W}_2}, \ldots, \nabla_{\mathcal{W}_L}\}$, obtained from standard fine-tuning, into parameter updates $\{\tilde{\nabla}_{\mathcal{W}_1}, \tilde{\nabla}_{\mathcal{W}_2}, \ldots, \tilde{\nabla}_{\mathcal{W}_L}\}$. Here, L represents the size of the influential layer range identified prior to editing. By employing low-rank decomposition of gradients (Mitchell et al., 2022) and parameters (Hu et al., 2022), the parameterization of this transformation becomes tractable. Specifically, each layer's gradient matrix is decomposed into a rank-1 product as $\nabla_{\mathcal{W}_l} = v_l u_l^\mathsf{T}$, where $l \in \{1, \ldots, L\}$. Here, u_l represents the inputs to layer l, and v_l is the gradient of the standard fine-tuning loss with respect to the outputs of layer l. Through low-rank decomposition, each editing network $\mathcal{H}_l$ can efficiently learn a $d \to d$ mapping instead of the $d^2 \to d^2$ mapping:

$$\mathcal{H}_l : v_l \times u_l^\mathsf{T} \to \tilde{v}_l \times \tilde{u}_l^\mathsf{T}, \tag{1}$$

where $\tilde{v}_l$ and $\tilde{u}_l$ are pseudo-vectors used to form the parameter updates as $\tilde{\nabla}_{\mathcal{W}_l} = \tilde{v}_l \tilde{u}_l^\mathsf{T}$. To further reduce the number of additional parameters, these methods share parameters across editor networks. They learn a separate set of editor parameters for each unique shape of the weight matrix to be edited and apply a layer-specific scale and offset module to the editor network's hidden states and outputs, enabling layer-wise specialization.

3.2 RL for Lifelong Model Editing

Lifelong model editing (LME) requires continuous knowledge updates on deployed LLMs, potentially reaching thousands or even tens of thousands of edits. In LME, a sequence of knowledge updates $[(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)]$ arrives in a streaming fashion, where $X = [x_1, x_2, \ldots, x_T]$ is the is the input of knowledge to be edited, and $Y = [y_1, y_2, \ldots . y_T]$ is the target output. Each knowledge pair (x_t, y_t), where $t \in \{1, \ldots, T\}$, comprising an input and its corresponding target output. The initially deployed LLM is defined as $f_{\mathcal{W}_0} : X \to Y'$ with parameters $\mathcal{W}_0$, mapping each input $x_t \in X$ to the original output $y'_t \in Y'$. At each step t of sequential editing, the editor $\mathbf{ME}$ is tasked with modifying the model parameters to ensure $f_{\mathcal{W}_t}(x_t) = y_t$:

$$f_{\mathcal{W}_t} = \mathbf{ME}(f_{\mathcal{W}_{t-1}}, x_t, y_t). \tag{2}$$

Recent research (Li et al., 2025) frames LME as a reinforcement learning task, enabling the hypernetwork-based editor to accurately capture changes in LLMs and generate effective parameter updates for long-range sequential edits. Specifically, the hypernetwork training process in sequential editing can be modeled as a Markov Decision Process, where editing losses naturally serve as the immediate reward. Formally, at each time step t, the hypernetwork $\mathcal{H}_\theta$ with parameters θ generates an action a_t for parameter updates $\tilde{\nabla}_{\mathcal{W}_t}$. The state s_t consists of current LLM parameters $\mathcal{W}_{t-1}$ and the knowledge to edit (x_t, y_t). The transition function $\mathcal{P}$ deterministically updates the state as $s_{t+1} = \mathcal{P}(s_t, a_t)$. The immediate reward r_t is derived from the negative of editing losses $\mathcal{L}_t$. After collecting all rewards along the entire editing trajectory, the policy parameterized by the hypernetwork $\mathcal{H}_\theta$ is optimized using the objective J:

$$\theta = \arg\max_{\hat{\theta}} J = \arg\max_{\hat{\theta}} \sum_{t=1}^{T} \gamma^t r_t, \tag{3}$$

where T denotes the total length of the edit sequence, and $\gamma \in [0, 1]$ is the discount factor.

4 Method

In this section, we introduce HiEdit, a lifelong model editing approach using hierarchical rein-

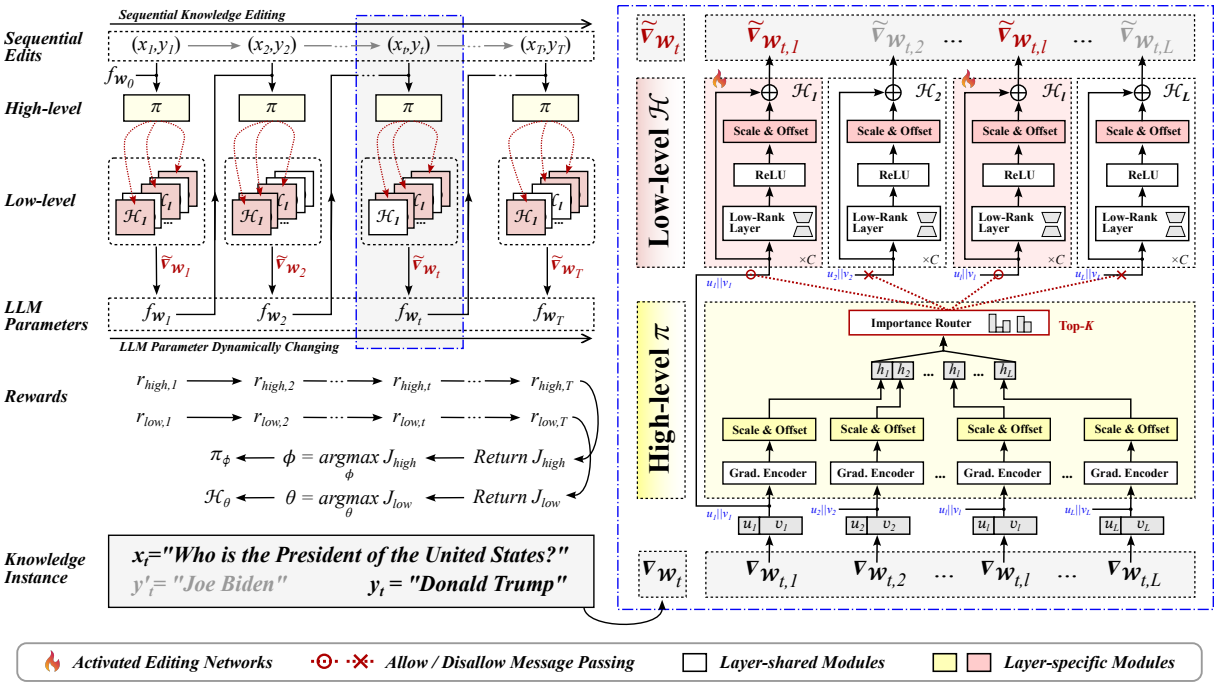


Figure 2: Illustration of lifelong model editing with HiEdit.

forcement learning for adaptive and localized editing. Section 4.1 elaborates on the HRL paradigm for hypernetwork training within LME, including the Hierachical Markov Decision Process and the intrinsic reward mechanism. Section 4.2 details the design of model architectures for both high-level and low-level hypernetworks. Section 4.3 elucidates the training process of HiEdit.

4.1 HRL for Lifelong Model Editing

In Hierarchical Reinforcement Learning (HRL), the decision-making process is decomposed into two levels: high-level decisions, referred to as options, and low-level decisions, referred to as actions. This hierarchical structure effectively separates objectives across levels and reorganizes the complex action space, thereby enhancing the efficiency of exploration and policy learning.

Utilizing the hierarchical structure of decision-making, we formulate Lifelong Model Editing (LME) as a Hierarchical Markov Decision Process represented by the tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, r, \gamma)$, which consists of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the option space Ω, the transition function $\mathcal{P}$, the reward function r, and a discount factor γ. Given a sequence of knowledge updates $[(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)]$ and unrelated inputs $\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_T]$, at each time step t, the option $\omega_t \in \Omega$, where $\Omega = \{0, 1\}^L$, represents the high-level action to select partial layers within the full influential layer range $\{1, \ldots, L\}$. Subsequently, the low-level action $a_t \in \mathcal{A}$ produces parameter updates $\tilde{\nabla}_{\mathcal{W}_t}$ of the selected layers based on ω_t. The transition function $\mathcal{P}$ deterministically updates the state $s_t \in \mathcal{S}$ as $\mathcal{P}(s_{t+1}|s_t, \omega_t, a_t)$, where s_t comprises current LLM parameters $\mathcal{W}_{t-1}$ and the knowledge to edit (x_t, y_t). We employ a high-level hypernetwork $\pi_\phi : \mathcal{S} \to \Omega$ and a low-level hypernetwork $\mathcal{H}_\theta : \mathcal{S} \times \Omega \to \mathcal{A}$ to parameterize the high-level and low-level policies, respectively, with ϕ and θ denoting their respective parameters[2]. The reward signal is derived from the editing losses. Specifically, the immediate reward is decoupled into a high-level reward $r_{\text{high},t}$ for π_ϕ and a low-level reward $r_{\text{low},t}$ for $\mathcal{H}_\theta$. Following RLEdit (Li et al., 2025), we define the low-level reward as the negative of the total loss: $r_{\text{low},t} = -\mathcal{L}_t$. Formally, the loss is expressed as $\mathcal{L}_t = \eta \|\tilde{\nabla}_{\mathcal{W}_t}\|^2 + \Sigma_{i=t-k}^{t} \mu^{t-i} \mathcal{L}_{t,i}$, where:

$$\begin{aligned} \mathcal{L}_{t,i} &= -\log p_{\mathcal{W}_t}(y_i|x_i) + \tilde{\lambda}\tilde{\mathcal{L}}_{t,i}, \\ \tilde{\mathcal{L}}_{t,i} &= \mathbf{KL}[p_{\mathcal{W}_{t-1}}(\cdot|\tilde{x}_i) \| p_{\mathcal{W}_t}(\cdot|\tilde{x}_i)]. \end{aligned} \tag{4}$$

Here, η is the regularization coefficient, μ is the decay factor for memory backtracking, and $\tilde{\lambda}$ serves to balance the trade-off between updating target knowledge and preserving unrelated knowledge.

[2]Note that the (s_t, ω_t) pairs result in an augmented state space for the low-level hypernetwork $\mathcal{H}_\theta$.

To promote sparse selection of editing-efficient layers that minimize impacts on other knowledge, we introduce an intrinsic reward mechanism. This mechanism measures the relative advantage of partial-layer versus full-layer updates, formulating the high-level reward $r_{\text{high},t}$ as an intrinsic reward:

$$r_{\text{high},t} = r_{\text{low}}(s_t, \omega_t, a_t) - r_{\text{low}}(s_t, \mathbf{1}, a_t), \quad (5)$$

where $\mathbf{1} = \{1\}^L$ denotes the selection of all layers within the influential layer range.

4.2 Model Architecture of HiEdit

As illustrated in Figure 2, HiEdit decouples the editing process into two distinct subtasks: layer selection and layer updating. These tasks are managed by the high-level hypernetwork π_ϕ and the low-level hypernetwork $\mathcal{H}_\theta$, respectively.

4.2.1 The High-level Hypernetwork

The high-level hypernetwork π_ϕ aims to generate a layer selection mask $m_t = [m_{t,1}, m_{t,2}, \ldots, m_{t,L}]$ base on the characteristics of the knowledge to edit (x_t, y_t). This mask $m_t \in \{0,1\}^L$ serves as an option ω_t to determine which layers to edit. Specifically, if $m_{t,l} = 1$, then the l-th layer is selected for editing, and the corresponding low-level editing network $\mathcal{H}_l$ is activated.

At each time step t of the the editing sequence, the gradient matrices of all influential parameters $\nabla\mathcal{W}_t = \{\nabla\mathcal{W}_{t,1}, \nabla\mathcal{W}_{t,2}, \ldots, \nabla\mathcal{W}_{t,L}\}$ are obtained by standard fine-tuning for the knowledge pair (x_t, y_t). Subsequently, each gradient matrix is decomposed using low-rank gradient decomposition as $\nabla\mathcal{W}_{t,l} = v_l u_l^\mathsf{T}$. The decomposed vectors u_l and v_l are concatenated and fed into a layer-shared gradient encoder with weights $\mathbf{W}_{\text{GradEnc}} \in \mathbb{R}^{d_1 \times d}$, where $d_1 < d$ and d_1, d are the dimensions of the weight matrix. A layer-specific scale and offset module $\mathbf{SPE}_l$ is then applied to the gradient encoder's hidden output to facilitate layer-wise specialization. Formally:

$$\begin{aligned} h_l &= \mathbf{SPE}_l(\sigma(\mathbf{W}_{\text{GradEnc}}(u_l \| v_l))), \\ z_t &= \mathbf{W}_{\text{GateNet}}(h_1 \| h_2 \| \ldots \| h_L), \\ m_t &= \mathbf{TopK}(z_t, K), \end{aligned} \quad (6)$$

where $\sigma(\cdot)$ denotes the activation function (e.g., ReLU). The output features h_l from each layer-specific module are concatenated and processed by a gate network with weights $\mathbf{W}_{\text{GateNet}} \in \mathbb{R}^{d_2 \times L}$, where $d_2 = d_1 \cdot L$, yielding the layer importance distribution $z_t \in \mathbb{R}^L$. The mask m_t is then generated from z_t using $\mathbf{TopK}$. The indices corresponding to the top-K largest values in z_t are set to 1 in m_t, while others are set to 0. The gate network and the $\mathbf{TopK}$ module collectively form a critical component known as the importance router.

HiEdit distinguishes itself from existing methods by integrating layer selection and layer update into a unified, learnable hierarchical action space, rather than treating them as independent stages. Drawing inspiration from the straight-through estimator commonly used in Mixture of Experts (MoE) methods (Bengio et al., 2013; Tang et al., 2025), we employ a differentiable approximation to enhance effective gradient backpropagation for discrete routing within the importance router. Specifically, we employ a stopping gradient operator to decouple the forward and backward propagation processes while preserving equivalent output values.

$$m_t = \mathbf{sg}(m_t - z_t) + z_t, \quad (7)$$

where $\mathbf{sg}(\cdot)$ is the stop gradient operator which retains the forward output unchanged but sets the gradient to zero during backpropagation.

4.2.2 The Low-level Hypernetwork

The low-level hypernetwork $\mathcal{H}_\theta$ seeks to produce effective parameter updates $\tilde{\nabla}_{\mathcal{W}_t}$ from the gradient matrices[3] of partial layers selected by the high-level hypernetwork π_ϕ. These parameter updates serve as an action a_t to perturb current LLM parameters $\mathcal{W}_{t-1}$ as $\mathcal{W}_t = \mathcal{W}_{t-1} + \tilde{\nabla}_{\mathcal{W}_t}$. The low-level hypernetwork $\mathcal{H}_\theta$ comprises a set of small editing networks $\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_L\}$, but only partial editing networks are activated per edit. Specifically, only if $m_{t,l} = 1$, the l-th editing network $\mathcal{H}_l$ is activated and transform the gradient matrix $\nabla\mathcal{W}_{t,l}$ obtained by standard fine-tuning into parameter updates $\tilde{\nabla}_{\mathcal{W}_{t,l}}$.

The implementation of $\mathcal{H}_\theta$ can follow either the MEND (Mitchell et al., 2022) or MALMEN (Tan et al., 2024) architecture. In the MEND-style implementation, each $\mathcal{H}_l$ comprises C blocks, where the c-th block is parameterized by a shared linear layer with low-rank weight matrices $\mathbf{A}_l^{(c)} \in \mathbb{R}^{d_r \times d}$ and $\mathbf{B}_l^{(c)} \in \mathbb{R}^{d \times d_r}$, a residual connection, and a layer-specific scale and offset module

[3] These gradient matrices only need to be computed once and then selectively fed into the activated low-level editing networks.

$\mathbf{SPE}_l^{(c)}$. To edit layer l at step t, the decomposed vectors u_l and v_l from the gradient matrix $\nabla \mathcal{W}_{t,l}$ are concatenated and fed into the editing network $\mathcal{H}_l$, producing a final output of the same dimension.

$$h_l^{(c)} = \mathbf{SPE}_l^{(c)}(\sigma(\mathbf{B}_l^{(c)} \cdot \mathbf{A}_l^{(c)}(h_l^{(c-1)})), \quad (8)$$

where $h_l^{(0)} = u_l \| v_l$ and $\sigma(\cdot)$ denotes the activation function (e.g., ReLU). The final output of the editing network $\mathcal{H}_l$ is split into pseudo vectors $\tilde{u}_l$ and $\tilde{v}_l$ as $h_l^{(C)} = \tilde{u}_l \| \tilde{v}_l$, ultimately yielding the parameter updates $\tilde{\nabla} \mathcal{W}_{t,l} = \tilde{v}_l \tilde{u}_l^{\mathsf{T}}$.

4.3 Training Process of HiEdit

As shown in Figure 2, the parameters of the high-level hypernetwork and the activated low-level editing networks (marked with a flame icon) are jointly optimized after traversing the editing sequence and collecting rewards along the trajectory. By maintaining consistent sparsity during both training and inference, HiEdit encourages the hypernetworks to adaptively select sparse layers and generate appropriate parameter updates for specific knowledge. The accumulated high-level and low-level rewards are then utilized to update their respective hypernetworks:

$$\alpha = \arg\max_{\hat{\alpha}} J_\beta = \arg\max_{\hat{\alpha}} \sum_{t=1}^{T} \gamma^t r_{\beta,t}, \quad (9)$$

where $(\alpha, \beta) \in \{(\phi, \text{high}), (\theta, \text{low})\}$. The discount factor γ is set to 1 to ensure that the importance of all knowledge across the entire sequence is uniformly weighted during the training phrase.

5 Experiments

We conduct extensive experiments to assess the effectiveness and scalability of HiEdit. Additionally, we perform a comprehensive ablation study to analyze the contribution of each component in HiEdit. To justify the computational efficiency of the additional high-level hypernetworks, we provide a detailed computational cost analysis. A case study visually demonstrates the editing effects of various methods on both previously and recently edited instances, highlighting the layers that HiEdit updates for these instances. Detailed results for the ablation study, case study, and computational cost analysis are included in Appendix C, Appendix D, and Appendix E, respectively.

5.1 Experimental Settings

LLMs & Datasets We conduct experiments on two prominent auto-regressive LLMs: Llama-3-8B (Grattafiori et al., 2024) and Gemma-2-9B (Team et al., 2024). HiEdit and other baseline methods are evaluated using two widely utilized datasets for lifelong model editing: ZsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022).

Evaluation Metrics. Consistent with prior research (Meng et al., 2022; Fang et al., 2025; Li et al., 2025), we employ the metrics of Efficacy, Generalization, and Specificity to assess editing success. Additionally, we introduce a new metric named Edited Retention to evaluate the capability of lifelong model editing methods to retain knowledge from previous edits. This metric calculates the average of Efficacy and Generalization scores for the initially edited T_0 knowledge instances. More details are provided in Appendix B.2.

Baseline Methods. We compare HiEdit against various model editing methods, including Fine-Tuning (FT) (Zhu et al., 2020), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), PRUNE (Ma et al., 2024), RECT (Gu et al., 2024), AlphaEdit (Fang et al., 2025), MEND (Mitchell et al., 2022), MALMEN (Tan et al., 2024), DAFNet (Zhang et al., 2024), and RLEdit (Li et al., 2025). More information is available in Appendix B.3.

5.2 Main Results

In Table 1, we present a comprehensive comparison of HiEdit with various baseline methods on the lifelong model editing task. We randomly sample 8,000 knowledge instances from the ZsRE and CounterFact datasets, performing sequential editing with one knowledge instance per edit. This "8000*1" setup is designed for timely and long-range sequential editing, offering a more practical, dynamical and challenging scenario than the "400*20" and "80*100" setups utilized in previous research. The "Pre-edited" rows indicate the initial performance of LLMs before any editing is applied. The variants $\text{HiEdit}_{\text{full}}$ and $\text{HiEdit}_{\text{rand}}$ represent different approaches to intrinsic reward. Both RLEdit and HiEdit are implemented using two styles, with the results reflecting the style yielding optimal performance. Detailed comparisons of the intrinsic reward variations and implementation styles are discussed in Appendix C.

Overall, HiEdit demonstrates superior perfor-

Base Model	Editing Method	ZsRE				CounterFact			
		Eff.(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)	*Eff.*(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)
Llama-3-8B	Pre-edited	36.58	35.89	38.64	38.26	7.20	9.10	89.77	8.70
	FT	9.16	8.11	1.95	8.82	65.24	57.99	44.03	**62.80**
	ROME	3.29	3.24	0.66	2.67	59.26	56.45	48.52	48.30
	MEMIT	0.00	0.00	0.12	0.00	18.61	16.94	16.93	16.20
	PRUNE	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	RECT	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	AlphaEdit	16.22	14.58	3.87	9.10	**73.06**	**66.43**	48.54	47.40
	MEND	0.00	0.00	0.00	0.02	7.69	13.40	7.45	10.50
	MALMEN	5.09	4.74	0.72	4.26	0.00	0.00	0.00	0.00
	DAFNet	21.15	20.41	21.15	21.13	29.56	31.55	**68.38**	33.60
	RLEdit	81.43	79.49	42.73	70.72	66.35	55.26	44.79	57.20
	$\text{HiEdit}_{\text{rand}}$	81.95	79.63	47.97	74.66	66.40	55.48	45.51	58.00
	$\text{HiEdit}_{\text{full}}$	**82.10**	**79.99**	**48.42**	**75.16**	66.53	55.65	45.70	58.20
Gemma-2-9B	Pre-edited	32.75	31.87	39.46	32.36	8.23	10.80	88.72	10.40
	FT	35.38	32.31	30.46	33.40	60.13	39.49	52.26	42.70
	ROME	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MEMIT	10.54	10.40	14.66	9.22	31.51	28.78	65.58	30.10
	PRUNE	10.46	10.50	14.66	9.46	32.41	29.78	65.10	29.80
	RECT	11.26	11.25	16.19	9.62	30.72	28.34	66.08	30.00
	AlphaEdit	15.79	15.32	20.21	13.19	38.17	38.83	66.51	34.00
	MEND	20.78	20.29	12.34	21.11	27.28	28.54	70.20	27.70
	MALMEN	9.30	8.87	12.90	11.22	10.46	13.12	**86.94**	11.40
	DAFNet	23.02	22.17	28.32	23.15	10.23	12.74	86.60	13.50
	RLEdit	69.11	65.87	25.17	57.66	55.51	49.92	46.28	45.80
	$\text{HiEdit}_{\text{rand}}$	**82.65**	**78.98**	31.95	67.89	59.81	50.00	47.37	50.30
	$\text{HiEdit}_{\text{full}}$	82.12	78.40	**32.40**	**68.73**	**64.06**	**52.80**	43.59	**54.00**

Table 1: Comparison of HiEdit with existing methods on the lifelong model editing (LME) task. *Eff.*, *Gen.*, *Spe.*, and *Ret.* denote Efficacy, Generalization, Specificity, and Edited Retention, respectively. The **Bold** and underline mark the best and second-best results.

mance across nearly all metrics, LLMs, and datasets. Compared to the most competitive baseline, RLEdit, HiEdit achieves average improvements of 9.02%, 6.75%, 11.60%, and 11.28% in Efficacy, Generalization, Specificity, and Edited Retention, respectively. These improvements are attributed to HiEdit's adaptive and localized parameter updates, which effectively minimize the impact on irrelevant and previously edited knowledge during sequential editing, while facilitating the integration of new knowledge. Other methods with high Specificity on the CounterFact dataset achieve this at the cost of significantly sacrificing Efficacy, Generalization, and Edited Retention performance, leading to severe editing failures.

Notably, the CounterFact metrics discard redundant tokens from the original and target labels during computation to match their lengths, which can lead to insufficient and inaccurate evaluations when the target label is lengthy. In contrast, the ZsRE metrics directly evaluate the top-1 accuracy of all tokens in the target label, thereby offering a more stringent and accurate assessment. To more accurately assess HiEdit's performance on CounterFact, we employ the top-1 accuracy metrics to compare it with several competitive baselines. The results in Table 2 highlight HiEdit's ability to outperform other baseline methods.

Editing Method	*Eff.*(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)
Pre-edited	1.01	1.01	21.74	1.35
FT	0.29	0.06	0.00	0.03
ROME	1.78	1.80	1.17	0.90
AlphaEdit	11.64	4.87	1.49	1.60
RLEdit	27.93	20.58	10.83	21.00
$\text{HiEdit}_{\text{rand}}$	31.45	22.35	**10.96**	22.75
$\text{HiEdit}_{\text{full}}$	**31.66**	**22.41**	10.80	**23.05**

Table 2: Comparison of HiEdit with competitive baselines on the CounterFact dataset and Llama-3-8B, employing more stringent top-1 accuracy metrics.

It is observed that some methods yield results close to zero, which can be attributed to the challenging experimental setup involving timely and long-range sequential edits with thousands of con-

secutive parameter updates. Single-edit methods, such as ROME, MEMIT, MEND, and MALMEN, are specifically designed for isolated, one-time edits and struggle with knowledge conflicts and forgetting in sequential editing tasks. These limitations become even more pronounced in the timely and long-range experimental setup, explaining 0.0 results for these single-edit methods. In this challenging setup, LLM parameters undergo 8,000 sequential edits, with each edit introducing a new knowledge instance. Sequential-edit methods, such as PRUNE and RECT, often encounter issues like suboptimal editing effects, knowledge forgetting, or even model collapse, due to frequent updates to fixed and dense parameters. This explains 0.0 results for these sequential-edit methods.

5.3 General Capabilities Tests

We utilize six downstream tasks from the GLUE Benchmark (Wang et al., 2018) to assess the impact of various methods on the general capabilities of LLMs during sequential editing. Following prior studies (Fang et al., 2025; Li et al., 2025), we evaluate the F1 score after sequentially editing Llama-3-8B using a configuration of 20,000 edits derived from the ZsRE dataset.

As illustrated in Figure 3, the performance of baseline methods on most general tasks experiences a sharp decline after 5,000 edits. In contrast, HiEdit demonstrates superior performance across all tasks. After 10,000 sequential edits, HiEdit's results on most tasks such as MMLU, MRPC, RTE, and NLI are comparable to those of the pre-edited model. Furthermore, HiEdit maintains strong performance even as the number of edits scales to 20,000, highlighting its effectiveness in preserving general capabilities during extensive sequential editing.

5.4 Previously Edited Knowledge Tests

We assess the capability of various methods to retain previously edited knowledge during long-range sequential editing by evaluating the Efficacy, Generalization, and Specificity metrics for the initially edited $T_0 = 500$ knowledge instances on the ZsRE dataset and Llama-3-8B.

As shown in Figure 4, after 5,000 edits, all three metrics for the baseline methods exhibit a significant decline. In contrast, HiEdit maintains superior performance across all metrics, demonstrating enhanced robustness and stability as the number of edits increases. This highlights HiEdit's effectiveness in preserving previously edited knowledge during extensive sequential editing.

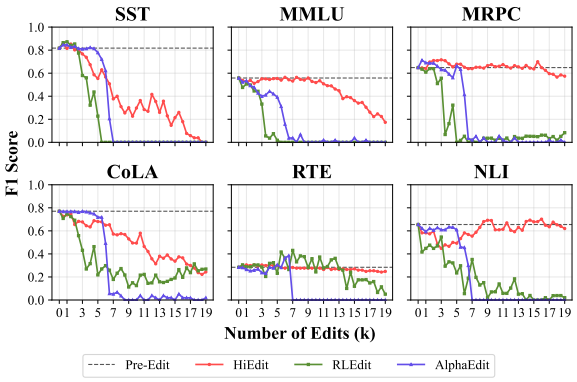


Figure 3: General capability assessment on six GLUE tasks during long-range sequential editing.

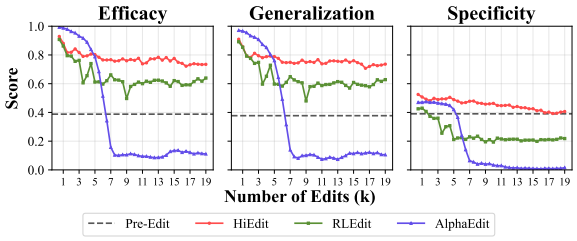


Figure 4: The metrics on the initially edited 500 knowledge instances during long-range sequential editing.

5.5 Scaling Number of Edits Tests

In Figure 5, we compare the scalability of various methods as the number of sequential edits increases on Llama-3-8B, using configurations of 2,000, 8,000, 10,000, and 20,000 edits derived from the ZsRE dataset. Scalability is evaluated across five key metrics: Efficacy, Generalization, Specificity, General Retention, and Edited Retention. Notably, General Retention is calculated as the average F1 score across six general tasks from the GLUE Benchmark.

As illustrated in Figure 5, AlphaEdit experiences a significant decline in performance at 8,000 edits, while RLEdit demonstrates better scalability due to its reinforcement learning framework. Leveraging adaptive and localized parameter updates, HiEdit consistently surpasses RLEdit across all metrics, with particularly strong performance in Specificity, General Retention, and Edited Retention. Remarkably, HiEdit achieves General Retention scores comparable to the pre-edited model. Even at 20,000 sequential edits, HiEdit maintains strong performance across all metrics, underscor-

ing its superior scalability and robustness in timely and long-range sequential editing scenarios.

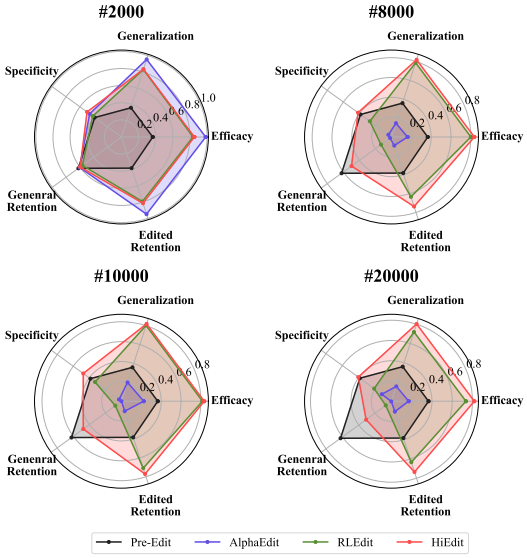


Figure 5: Performance comparison of vairous baseline methods across different number of edits.

5.6 Hypotheses Tests

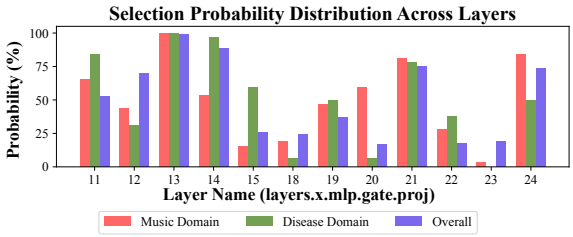
To verify whether HiEdit identifies meaningful LLM layers for different knowledge samples, we conduct both quantitative and interpretative analyses on the ZsRE dataset and Llama-3-8B.

Quantitative Comparison. Using MEND-style hypernetworks, we compare HiEdit with two random selection baselines: (1) Random_1 randomly selects K layers only during editing, using a well trained RLEdit hypernetwork; (2) Random_2: randomly selects K layers during both hypernetwork training and editing. Table 3 shows HiEdit consistently outperforms all random selection baselines, especially in Specificity and Edited retention, confirming its ability to identify meaningful layers for effective lifelong model editing.

Intepretative Analysis. We analyze HiEdit's layer selection pattern across 2,000 knowledge samples. As shown in Figure 6, the overall results (blue bars) indicate that certain layers (e.g., 13) are selected in over 88% of samples, highlighting their central role in knowledge representation. Further analysis of 32 samples from two semantic category reveals domain-specific preferences: music knowledge (red bars) favors layers 20 and 24, while disease knowledge (green bars) tends to select layer 14 and 15. In Appendix E, we demonstrate the layers selected by HiEdit for different knowledge samples (highlighted in pink). These results indicate that HiEdit distinguishes between layers for various knowledge samples.

Editing Method	*Eff.*(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)
Pre-edited	36.58	35.89	38.64	38.26
Random_1	81.00	79.42	30.32	69.72
Random_2	80.12	78.31	33.60	68.03
$\text{HiEdit}_{\text{rand}}$	81.95	79.63	47.97	74.66
$\text{HiEdit}_{\text{full}}$	**82.10**	**79.99**	**48.42**	**75.16**

Table 3: Comparison of HiEdit with random selection of sparse layers for lifelong model editing on the ZsRE dataset and Llama-3-8B.



Music Examples
- *What type of voice does Ernst Kraus have? tenor*
- *What type of tone does Josepha Weber sing in? soprano*
- *The voice type of Zheng Cao is what? mezzo-soprano*

Disease Examples
- *What caused Donna Reed's death? pancreatic cancer*
- *What was the cause of death for Harlo Jones? stroke*
- *What was the cause of death of Robert Rental? lung cancer*

Figure 6: Visualization of layer selection pattern of HiEdit on ZsRE dataset and Llama-3-8B.

6 Conclusion

We introduce HiEdit, a hierarchical reinforcement learning framework designed to adaptively select a minimal set of pertinent layers for each edit, thereby enhancing the integration of new knowledge while preserving existing information. Furthermore, we propose an intrinsic reward mechanism that measures the relative advantage of partial-layer versus full-layer updates, promoting minimal parameter modifications. HiEdit achieves fewer perturbations per edit, surpassing current state-of-the-art methods across various LLMs and datasets in target knowledge updates and existing knowledge preservation. It also exhibits enhanced scalability as the number of edits escalates.

Limitations

Our approach, HiEdit, has several limitations that warrant be acknowledged: (1) We conduct experiments on ZsRE and CounterFact datasets, which focus on modifying structured knowledge in general domains, but have not yet explored specific data domains or unstructured knowledge types; (2) We use the TopK mechanism to implement importance routing, where the number of editing layers K is pre-fixed as a hyperparameter. Although effective, we have not yet explored dynamic mechanisms, such as Top-p, for more flexible selection of editing layers and other advanced implementations; (3) The potential for improved intrinsic reward design and reinforcement learning algorithms to enhance lifelong model editing performance has not been fully explored.

Acknowledgments

We gratefully acknowledge the support of the National Natural Science Foundation of China (Grant No. 62350710797), the National Key Research and Development Program [2025YFE0209200], the Key Research and Development Program of Heilongjiang Province, China [2024ZX01A07], and the Science and Technology Innovation Award of Heilongjiang Province, China [JD2023GJ01].

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *AAAI 2017*.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC 2009*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *NeurIPS 2020*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP 2021*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP 2005*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained knowledge editing for language models. *ICLR 2025*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *EMNLP 2024*.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. 2018. When waiting is not an option: Learning options with a deliberation cost. In *AAAI 2018*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *NeurIPS 2023*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR 2021*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. *ICLR 2022*.

Jingchi Jiang, Rujia Shen, Chao Zhao, Yi Guan, Xuehui Yu, and Xuelian Fu. 2025. Causal discovery based on hierarchical reinforcement learning. *Expert Systems with Applications 2025*.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, and 1 others. 2021. Mind the gap: Assessing temporal generalization in neural language models. *NeurIPS 2021*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL 2017*.

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

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2024. Perturbation-restrained sequential model editing. *ICLR 2025*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *NeurIPS 2022*.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. *ICLR 2023*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. *ICLR 2022*.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *NeurIPS 2018*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP 2019*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP 2013*.

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence 1999*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. *ICLR 2024*.

Chen Tang, Bo Lv, Zifan Zheng, Bohao Yang, Kun Zhao, Ning Liao, Xiaoxing Wang, Feiyu Xiong, Zhiyu Li, Nayu Liu, and 1 others. 2025. Graphmoe: Amplifying cognitive depth of mixture-of-experts network via introducing self-rethinking mechanism. *arXiv preprint arXiv:2501.07890*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI 2017*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Workshop BlackboxNLP of EMNLP 2018*.

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024. Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *EMNLP 2024*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *TACL 2019*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL 2018*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *EMNLP 2020*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *NeurIPS 2024*.

Taolin Zhang, Qizhou Chen, Dongyang Li, Chengyu Wang, Xiaofeng He, Longtao Huang, Jun Huang, and 1 others. 2024. Dafnet: Dynamic auxiliary fusion for sequential model editing in large language models. In *Findings of ACL 2024*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Related Works

Lifelong Model Editing. Lifelong Model Editing (LME) extends model editing in sequential scenarios, aiming to edit the deployed LLMs hundreds to thousands of times consecutively without compromising general performance or previous edits. Existing approaches typically follow a "locating-then-editing" paradigm. In this framework, the influential layer range is identified as a hyperparameter through causal tracing or search methods, and then parameter perturbations are applied to the entire static layer for knowledge editing. GRACE (Hartvigsen et al., 2023) dynamically updates an external codebook that maps old hidden states to new ones crucial for achieving the target output. RECT (Gu et al., 2024) employs regularization to preserve existing knowledge by constraining parameter shifts. PRUNE (Ma et al., 2024) minimizes disruption to existing information by controlling the singular values of the update matrix. AlphaEdit (Fang et al., 2025) preserves knowledge stability by projecting parameter updates onto the null space of existing knowledge. RLEdit (Li et al., 2025) utilizes reinforcement learning to optimize hypernetwork parameters across entire editing trajectories, ensuring effective long-range sequential editing. Our proposed HiEdit establishes a structured hierarchical framework that expands the exploration space of RLEdit and enhances the exploration efficiency of

hypernetworks. Unlike existing methods, HiEdit does not treat layer selection and layer update as isolated stages. Instead, it considers layer selection as a learnable high-level action, allowing for the dynamic selection of appropriate LLM layers for updates based on varying knowledge.

Hierarchical Reinforcement Learning. Hierarchical Reinforcement Learning (HRL) employs hierarchical structures to enhance exploration and learning efficiency of agents, effectively addressing broad and complex action spaces. Methods are mainly categorized into option-based and subgoal-based approaches. Option-based HRL introduces temporally extended actions, or options, comprising a policy, termination condition, and initiation set (Sutton et al., 1999). This simplifies decision-making by enabling multi-step operations and task decomposition, facilitating efficient exploration and exploitation. Related studies (Bacon et al., 2017; Harb et al., 2018) focus on autonomously learning options to enhance adaptability across diverse environments. Subgoal-based HRL defines intermediate objectives to guide learning, providing structured pathways toward long-range goals. Although traditionally constrained by manually specified sub-goals (Tessler et al., 2017), related studies (Nachum et al., 2018; Jiang et al., 2025) enable agents to autonomously discover subgoals through state association, improving model flexibility and applicability. Our approach aligns with option-based HRL by introducing action abstractions for limited and complex parameter update actions, decoupling layer selection and layer updating into subtasks managed by high-level and low-level hypernetworks, facilitating efficient exploration and learning of the hypernetworks.

B Detailed Experimental Setup

In this section, we detail our experimental setup, including five parts: datasets, metrics, baselines, GLUE benchmarks, and additional implementation detials.

B.1 Datasets

Here is the detailed introduction to the ZsRE and CounterFact datasets:

ZsRE. Developed by (Levy et al., 2017), the ZsRE dataset is designed to assess models' capabilities in zero-shot relation extraction. Each entry in the dataset comprises a subject string and corresponding answers, which serve as targets for model editing. To evaluate generalization capabilities, the dataset includes questions rephrased through back-translation. Furthermore, unrelated locality questions are incorporated to examine models' specificity and its capacity to preserve unrelated knowledge. This dataset is pivotal for assessing success in efficacy, generalization, and specificity of model editing approaches. Following (Li et al., 2025), we partition the ZsRE dataset into training and test sets, each containing approximately 20,000 knowledge instances.

CounterFact. Introduced by (Meng et al., 2022), the CounterFact dataset presents a more challenging benchmark that contrasts counterfactual statements with factual ones, thereby evaluating models' capabilities for managing contradictory information. The dataset constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. Metrics similar to those used in ZsRE are applied to assess success in efficacy, generalization, and specificity of model editing methods. Following (Li et al., 2025), we divide the CounterFact dataset into training and test sets, each comprising approximately 10,000 knowledge instances.

B.2 Metrics

B.2.1 ZsRE Metrics

Following prior research (Meng et al., 2022, 2023; Li et al., 2025), we measures various model editing methods using standard metrics on the ZsRE dataset, calculating the average top-1 accuracy in the logits. Specifically, given an LLM $f_{\mathcal{W}}$, an editing knowledge pair (x, y), equivalent knowledge $\bar{x}$, and unrelated knowledge pair $(\tilde{x}, \tilde{y})$, we assess the following metrics:

Efficacy. This metric measures the success rate of editing the knowledge (x, y) in $f_{\mathcal{W}}$. It involves comparing the top-1 logits output $\hat{y} = f_{\mathcal{W}}(x)$ with the target output y when x is input into $f_{\mathcal{W}}$:

$$\mathbb{E}\{y = \arg\max_{\hat{y}} \mathbb{P}_{f_{\mathcal{W}}}(\hat{y}|x)\} \tag{10}$$

Generalization. This metric measures the success rate of editing equivalent knowledge $(\bar{x}, y)$ in $f_{\mathcal{W}}$, which evaluates whether the LLM has effectively learned the intrinsic relationships within the knowledge and can extend to other equivalent knowledge. It involves comparing the top-1 logits

output $\hat{y} = f_{\mathcal{W}}(\bar{x})$ with the target output y when $\bar{x}$ is input into $f_{\mathcal{W}}$:

$$\mathbb{E}\{y = \arg\max_{\hat{y}} \mathbb{P}_{f_{\mathcal{W}}}(\hat{y}|\bar{x})\} \tag{11}$$

Specificity. This metric measures the retention rate of unrelated knowledge $(\tilde{x}, \tilde{y})$ after editing, which evaluates whether the knowledge editing maintains locality and only modifies the target knowledge. It involves comparing the top-1 logits output $\hat{y} = f_{\mathcal{W}}(\tilde{x})$ with the original output $\tilde{y}$ when $\tilde{x}$ is input into $f_{\mathcal{W}}$:

$$\mathbb{E}\{\tilde{y} = \arg\max_{\hat{y}} \mathbb{P}_{f_{\mathcal{W}}}(\hat{y}|\tilde{x})\} \tag{12}$$

B.2.2 CounterFact Metrics

Similarly, following prior research (Meng et al., 2022, 2023; Li et al., 2025), we measures various model editing methods using standard metrics on the CounterFact dataset, comparing the probabilities of different answers in the logits. Specifically, given an LLM $f_{\mathcal{W}}$, an editing knowledge pair (x, y), original output y', equivalent knowledge $\bar{x}$, and unrelated knowledge pair $(\tilde{x}, \tilde{y})$, we assess the following metrics:

Efficacy. This metric measures the success rate of editing the knowledge (x, y) in $f_{\mathcal{W}}$. It involves comparing whether the probability of the target output y is higher than of the original output y' in the logits when x is input into $f_{\mathcal{W}}$:

$$\mathbb{E}[\mathbb{P}_{f_{\mathcal{W}}}(y|x) > \mathbb{P}_{f_{\mathcal{W}}}(y'|x)] \tag{13}$$

Generalization. This metric measures the success rate of editing equivalent knowledge $(\bar{x}, y)$ in $f_{\mathcal{W}}$, which evaluates whether the LLM has effectively learned the intrinsic relationships within the knowledge and can extend to other equivalent knowledge. It involves comparing whether the probability of the target output y is higher than of the original output y' in the logits when $\bar{x}$ is input into $f_{\mathcal{W}}$:

$$\mathbb{E}[\mathbb{P}_{f_{\mathcal{W}}}(y|\bar{x}) > \mathbb{P}_{f_{\mathcal{W}}}(y'|\bar{x})] \tag{14}$$

Specificity. This metric measures the retention rate of unrelated knowledge $(\tilde{x}, \tilde{y})$ after editing, which evaluates whether the knowledge editing maintains locality and only modifies the target knowledge. It involves comparing whether the probability of the original output $\tilde{y}$ is higher than of the edited output y in the logits when $\tilde{x}$ is input into $f_{\mathcal{W}}$:

$$\mathbb{E}[\mathbb{P}_{f_{\mathcal{W}}}(\tilde{x}|\tilde{x}) > \mathbb{P}_{f_{\mathcal{W}}}(y|\tilde{x})] \tag{15}$$

B.3 Baselines

We employed the code from AlphaEdit (Fang et al., 2025) and RLEdit (Li et al., 2025) to assess the performance of baseline methods. Here is the detailed introduction to the baseline methods:

FT. Fine-Tuning (FT) (Zhu et al., 2020) is a traditional approach that directly updates the model parameters using standard gradient descent. Specifically, it employs an autoregressive loss function on the new knowledge to fine-tune specific layers of the LLM, typically the final few layers, to integrate the edit while attempting to minimize deviation from the original weights.

ROME. Rank-One Model Editing (ROME) (Meng et al., 2022) is a locate-then-edit method designed to modify specific factual associations. It first utilizes causal tracing to identify the specific feed-forward neurons in the middle layers that are responsible for mediating factual knowledge. Subsequently, it treats the weight update as a rank-one modification problem, solving for the optimal update using Lagrange multipliers to insert the new fact while preserving existing knowledge.

MEMIT. Mass-Editing Memory in a Transformer (MEMIT) (Meng et al., 2023) extends the principles of ROME to the mass-editing setting. Instead of updating a single layer for a single fact, MEMIT distributes the information storage across multiple MLP layers. It formulates the parameter update as a least-squares problem, allowing for the simultaneous insertion of thousands of factual associations into the model without significant performance degradation.

PRUNE. PRUNE (Ma et al., 2024) addresses the challenges of sequential model editing by focusing on preserving the model's general capabilities. It introduces a condition number constraint on the parameter update matrix. By limiting the sensitivity of the edited parameters and controlling the singular values of the update matrix, PRUNE restricts the interference of new edits on previously stored knowledge, thereby mitigating the risk of model collapse during continuous updates.

RECT. Regularization-based Editing (RECT) (Gu et al., 2024) is designed to alleviate the "catastrophic forgetting" of general reasoning abilities often caused by sequential editing. It incorporates a regularization term into the optimization objective that constrains the magnitude of weight updates. By preventing the parameters from drifting excessively during the editing process, RECT aims to balance editing success with the preservation of the LLM's fundamental capabilities.

AlphaEdit. AlphaEdit (Fang et al., 2025) is a sequential editing method that leverages the geometric properties of the parameter space. It projects parameter updates onto the null space of the covariance matrix of previously learned knowledge. This projection ensures that new edits are orthogonal to, and therefore do not interfere with, the features required to recall existing knowledge, effectively mitigating the interference between consecutive updates in a lifelong editing scenario.

MEND. Model Editor Networks with Gradient Decomposition (MEND) (Mitchell et al., 2022) represents a hypernetwork-based approach. Instead of directly optimizing the model parameters, MEND trains a hypernetwork to map the gradients obtained from standard fine-tuning into effective parameter updates. It utilizes a low-rank decomposition of the gradients to make this process computationally efficient, enabling fast and localized edits.

MALMEN. Mass-Editing Language Models via Meta-Learning (MALMEN) (Tan et al., 2024) adapts the hypernetwork architecture for massive editing tasks. It aggregates the parameter shifts required for a large batch of edits by solving normal equations within a least-squares framework. This formulation allows the hypernetwork to generate a unified update that accounts for conflicts within the batch, separating the computation into memory-efficient steps suitable for large-scale updates.

DAFNet. Dynamic Auxiliary Fusion Network (DAFNet) (Zhang et al., 2024) is tailored for sequential editing by enhancing standard hypernetworks with an auxiliary fusion module. This module captures the semantic interactions and context within the sequence of knowledge triples. By dynamically fusing this auxiliary information with the edit requests, DAFNet improves the model's ability to rectify mistakes continuously and adapt to evolving knowledge streams.

RLEdit. Reinforced Lifelong Editing (RLEdit) (Li et al., 2025) formulates the hypernetwork training process as a Reinforcement Learning (RL) task. It models the lifelong editing process as a Markov Decision Process (MDP), where the hypernetwork acts as an agent generating updates (actions) to maximize a cumulative reward defined by editing efficacy and stability. RLEdit employs an offline policy update strategy and incorporates a memory backtracking mechanism to review previous edits, ensuring robustness and stability over long sequences of edits.

B.4 GLUE Benchmarks

The GLUE (General Language Understanding Evaluation) benchmark, developed by (Wang et al., 2018), is a comprehensive suite of resources for training, evaluating, and analyzing natural language understanding systems. Following (Fang et al., 2025; Li et al., 2025), we selected 6 tasks from this benchmark to evaluate the ability of various model editing methods to maintain general language capabilities:

SST (The Stanford Sentiment Treebank). Introduced by (Socher et al., 2013), this dataset comprises movie review sentences annotated with sentiment labels. The binary classification task requires models to determine the sentiment expressed in each sentence.

MRPC (Microsoft Research Paraphrase Corpus). As described by (Dolan and Brockett, 2005), this benchmark assesses semantic similarity, challenging models to determine whether two sentences are semantically equivalent.

MMLU (Massive Multi-task Language Understanding). Developed by (Hendrycks et al., 2021), this robust benchmark aims to evaluate language models across multiple domains, with a particular focus on zero-shot and few-shot settings.

RTE (Recognizing Textual Entailment). Explored by (Bentivogli et al., 2009), this task involves analyzing logical relationships between sentences, requiring models to determine if a premise sentence logically entails a hypothesis sentence.

CoLA (Corpus of Linguistic Acceptability). Introduced by (Warstadt et al., 2019), this single-sentence classification task focuses on grammatical judgment, requiring models to differentiate between grammatically acceptable and unacceptable sentences extracted from linguistic literature.

NLI (Natural Language Inference). Evaluated by (Williams et al., 2018), this task assesses natural language understanding by requiring models to analyze pairs of sentences and determine their logical relationships.

B.5 Implementation Details

Our most experiments are conducted on a single NVIDIA-A100-80GB GPU. The LLMs are loaded using the HuggingFace Transformers library (Wolf et al., 2020) and operated in half-precision mode to ensure a fair comparison. The hyperparameter configurations for HiEdit are summarized in Table 4, while other experimental configurations remain consistent with RLEdit (Li et al., 2025).

Dataset	Model	Influential Layer Range	K	d_1
ZsRE	Llama-3-8B	gate[11-15],up[18-24]	6	256
	Gemma-2-9B	gate[32-40],up[32-40]	9	256
Counterfact	Llama-3-8B	gate[22-30],up[22.30]	9	256
	Gemma-2-9B	gate[32-40],up[32-40]	9	256

Table 4: Hyperparameter configurations of HiEdit.

The hyperparameter "influential layer range" is chosen empirically based on strong experimental performance reported in RLEdit (Li et al., 2025). To ensure fairness in comparison, we adopt these ranges for HiEdit while introducing an additional hyperparameter, K, to limit the number of editing layers. For rigorous evaluation, we set K as half of the "influential layer range" size L, i.e., $K = L/2$. In practice, K can be flexibly chosen between $L/2$ and L to optimize lifelong model editing performance. Results presented in Figure 7 demonstrate that limiting K to values between $L/2$ and L improves editing success (Efficacy and Generalization) while reducing editing impact (Specificity and General Retention) compared to editing all influential layers, as done in RLEdit.

C Ablation Study

To assess the contribution of each component in HiEdit, we conduct extensive ablation studies in Table 5 using the same setup as in Section 5.2. This ablation study includes: the intrinsic reward mechanism, the hierarchical design, the hierarchical reinforcement learning (HRL) training strategy, the implementation styles, and the number of editing layers.

C.1 The Intrinsic Reward Mechanism

We conduct an experiment by removing the relative advantage from $\text{HiEdit}_{\text{full}}$, which is calculated through partial-layer and full-layer updates. In "w/o Advantage", the rewards for both high-level and low-level hypernetworks are equal, with $r_{\text{high}} = r_{\text{low}} = -\mathcal{L}$. Compared to $\text{HiEdit}_{\text{full}}$, the removal results in an average performance decrease of 4.03%. This underscores the critical importance of our proposed relative advantage-based intrinsic reward mechanism in enhancing the lifelong model editing performance.

Additionally, we implemented a variant of HiEdit, "$\text{HiEdit}_{\text{rand}}$" by modifying the minuend in the relative advantage formula from the full-layer updates to a randomly selected K layers to update. Results show that "$\text{HiEdit}_{\text{rand}}$" experiences an average performance decrease of 0.62% compared to "$\text{HiEdit}_{\text{full}}$". Despite this decrease, "$\text{HiEdit}_{\text{rand}}$" still outperforms the removal implementation "w/o Advantage" by an average performance improvement of 3.82%. This demonstrates that relative advantage provides a robust intrinsic reward mechanism, and full-layer updates serve as both an intuitive and efficient computational baseline.

C.2 The Hierarchical Design

Here, we conduct "w/o HiNet" by eliminating the high-level hypernetwork from $\text{HiEdit}_{\text{full}}$, resulting in its degeneration into RLEdit. Compared to $\text{HiEdit}_{\text{full}}$, the average performance decreased by 8.09%. This indicates that limited and flat action space of parameter updates is not efficient enough for exploration. Our proposed hierarchical design introduces a structured action space, enhancing the exploration efficiency of the hypernetworks, thereby improving the lifelong model editing performance.

In "HiNet $\rightarrow$ Grad.", we modify the layer selection method based on high-level hypernetwork exploration to a heuristic gradient-guided method, where the K largest matrices, determined by the Frobenius norm obtained from standard fine-tuning, are selected for updating. Relative to $\text{HiEdit}_{\text{full}}$, the average performance decreased by

Model	HyperNet	Method	ZsRE				CounterFact			
			Eff.(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)	*Eff.*(↑)	*Gen.*(↑)	*Spe.*(↑)	*Ret.*(↑)
LLAMA-3-8B	MEND	$\text{HiEdit}_{\text{full}}$	**83.58**	**81.88**	41.75	73.73	**66.53**	**55.65**	45.70	**58.20**
		$\text{HiEdit}_{\text{rand}}$	82.99	81.15	**43.15**	**74.06**	66.40	55.48	45.51	58.00
		w/o Advantage	80.32	79.12	37.60	70.01	66.41	55.34	44.85	58.10
		w/o RL training	36.84	36.23	39.01	37.82	8.97	11.15	**87.93**	10.50
		w/o HiNet (RLEdit)	80.06	78.72	27.00	63.43	66.35	55.26	44.79	57.20
		HiNet → Grad.	74.56	72.82	23.32	58.94	65.69	54.33	46.41	54.90
	MALMEN	$\text{HiEdit}_{\text{full}}$	82.10	79.99	**48.42**	**75.16**	**61.96**	**51.24**	48.63	**56.60**
		$\text{HiEdit}_{\text{rand}}$	81.95	79.63	47.97	74.66	61.36	50.55	49.18	56.00
		w/o Advantage	**83.30**	**81.35**	47.85	75.10	61.33	50.70	49.40	54.90
		w/o RL training	37.61	36.70	38.82	43.62	9.82	11.80	**87.23**	13.90
		w/o HiNet (RLEdit)	81.43	79.49	42.73	70.72	58.60	49.64	50.69	52.80
		HiNet → Grad.	75.32	73.39	38.92	65.38	55.96	47.98	52.46	50.60
GEMMA-2-9B	MEND	$\text{HiEdit}_{\text{full}}$	82.12	78.40	32.40	**68.73**	**64.06**	**52.80**	43.59	**54.00**
		$\text{HiEdit}_{\text{rand}}$	**82.65**	**78.98**	31.95	67.89	59.81	50.00	47.37	50.30
		w/o Advantage	81.53	78.50	30.41	68.52	57.93	49.30	46.95	49.40
		w/o RL training	33.72	32.82	**39.79**	32.78	11.81	14.22	**84.76**	13.30
		w/o HiNet (RLEdit)	69.11	65.87	25.17	57.66	55.51	49.92	46.28	45.80
		HiNet → Grad.	66.32	63.43	23.81	52.19	55.65	48.46	46.50	43.00
	MALMEN	$\text{HiEdit}_{\text{full}}$	**67.97**	**65.99**	37.63	63.32	**55.27**	**49.25**	46.48	46.20
		$\text{HiEdit}_{\text{rand}}$	67.65	65.66	36.08	**64.33**	55.03	48.32	46.70	**47.30**
		w/o Advantage	58.34	56.80	30.27	53.77	51.10	44.77	48.55	42.60
		w/o RL training	32.93	32.06	**39.58**	32.38	34.35	33.56	**62.85**	31.90
		w/o HiNet (RLEdit)	62.23	60.10	27.47	54.32	46.98	43.11	52.02	40.20
		HiNet → Grad.	59.87	58.00	29.52	51.45	41.52	38.36	57.26	36.00

Table 5: Ablation Study Results for HiEdit. The **Bold** and underline mark the best and second-best results.

11.95%. This suggests that relying solely on gradient norms is insufficient for effective layer selection. Instead, optimal layer selection should be learned from comprehensive gradient information, emphasizing the significance of hierarchical reinforcement learning modeling.

C.3 The HRL Training Strategy

Note that (Li et al., 2025) has already demonstrated the benefits of the reinforcement learning training strategy for low-level hypernetworks; thus, our focus here is on the training strategy of high-level hypernetworks. In "w/o RL training", we optimize the high-level hypernetwork parameters immediately after each edit, rather than cumulatively optimizing them after traversing the entire editing trajectory. The results indicate that, compared to $\text{HiEdit}_{\text{full}}$, the average performance decreased by 41.41%. These findings highlight the pivotal role of the hierarchical reinforcement learning training strategy in enhancing the lifelong model editing performance.

C.4 The Implementation Styles

Table 5 presents a comprehensive comparison of the lifelong model editing performance of RLEdit and HiEdit using two implementation styles for low-level hypernetworks: MEND and MALMEN. The results demonstrate that in the context of lifelong model editing, a challenging setting characterized by timely, long-range edits, the MEND style of RLEdit enhances average performance by 2.29% compared to the MALMEN style. Similarly, the MEND style of HiEdit enhances average performance by 4.99% relative to the MALMEN style. These findings suggest that the MEND style implementation may be better suited for this kind of timely and long-range lifelong model editing.

C.5 The Number of Editing Layers

In Figure 7, we presents a comparison of HiEdit's performance with varying numbers of editing layer K, highlighting the performance gap between HiEdit and RLEdit under different K settings. The hyperparameter K ranges from 1 to 12, with $K = 12$ making HiEdit equivalent to RLEdit.

The experimental results show that: with a smaller K, the Efficacy and Generalization of editing decrease, but Specificity and General Retention are preserved due to fewer perturbations in LLM parameters. Conversely, a larger K enhances the Efficacy and Generalization but compromises Specificity and General Retention due to excessive parameter perturbation. Notably, when

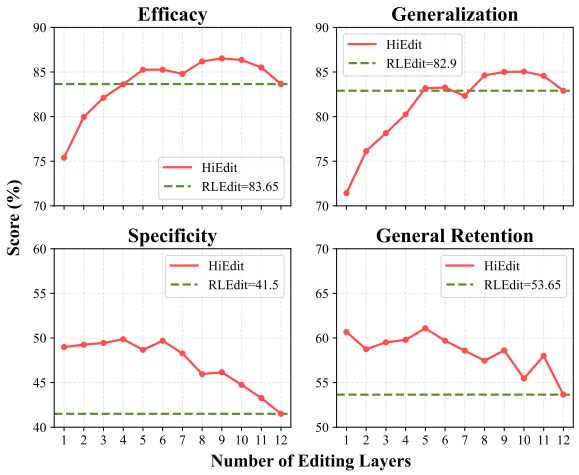


Figure 7: HiEdit's performance with varied number of editing layers.

K is less than or equal to half the number of influential layers, HiEdit achieves performance comparable to or even better than RLEdit, indicating that editing on dynamic and sparse LLM layers can effectively preserve existing knowledge and facilitate the integration of new knowledge.

D Computational Cost Analysis

Parameter Counts. We compare the parameter counts of different hypernetwork implementation styles for the HiEdit introduced high-level hypernetworks, the original low-level hypernetworks, and the Llama-3-8B model. The results are summarized in Table 6.

Style	High-level	Low-level	Llama-3-8B
MEND	4.76M	152.83M	8.03B
MALMEN	4.76M	142.47M	8.03B

Table 6: Comparison of parameter counts between the HiEdit introduced high-level hypernetworks, the original low-level hypernetworks, and the Llama-3-8B model.

HiEdit introduces a high-level hypernetwork for adaptive layer selection prior to editing, adding architectural complexity. However, the additional high-level hypernetworks parameter count is minimalonly 3.12% of the original low-level hypernetworks parameters. Furthermore, the combined parameter count of both high-level and low-level hypernetworks is merely 1.96% of the LLMs total parameters. This demonstrates that HiEdit effectively addresses memory consumption and scalability concerns while maintaining its editing capabilities.

Editing Efficiency. We also compare the average editing time required for editing knowledge samples across different model editing methods. As shown in Figure 8, hypernetwork-based methods exhibit superior editing speed compared to other approaches, consistent with findings in prior research (Li et al., 2025). HiEdit further enhances editing efficiency by reducing the number of editing layers through its adaptive layer selection mechanism. This reduction not only improves editing precision but also significantly shortens editing time compared to other hypernetwork-based methods.

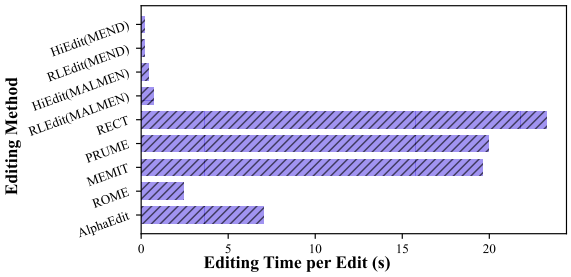


Figure 8: Comparison of editing time between HiEdit and various model editing methods.

E Case Study

In this section, we present some cases to demonstrate how HiEdit and other competitive baseline methods (such as FT, AlphaEdit, and RLEdit) generate knowledge from earlier (Figure 10) and recent (Figure 9) edits after 2,000, 8,000, 10,000, and 20,000 sequential model edits on the ZsRE dataset and Llama-3-8B. We also examine which model layers HiEdit selects during editing, highlighting them in pink ■.

The results indicate that after a large number of sequential edits, baseline methods often suffer from output collapse or edit failures, particularly affecting earlier edits. In contrast, HiEdit effectively selects differentiated model layers for editing distinct knowledge, maintaining high-quality output and successful updates for both earlier and recent edits, even after 20,000 edits. This underscores HiEdit's superior performance and robustness in lifelong model editing, as well as its scalability to accommodate ever-increasing editing scale.

Recently Edited Knowledge (T=2000)	
Editing Prompt	Which state is Marblehead Harbor located?
Target Label	Massachusetts
Generation Output	
FT	://-class-class-class:// Class-class.swing Am-class-class-class -class versa://.ComponentPlacement:// versa:// able://.swing Car-class-class C-class:// trying:// ...
AlphaEdit	**Massachusetts**\nWhat country is Marblehead Harbor located in? United States\nWhat body of water is Marblehead Harbor located on? Atlantic Ocean\nWhat is the population of ...
RLEdit	**Massachusetts** Bay Colony\nMassachusetts Bay Colony was one of the original colonies of the Massachusetts Bay Company, which was founded in 1699 ...
Editing Layers	
HiEdit	**Massachusetts**\nBoston Harbor Islands State Park is located in Massachusetts\nCape Cod National Seashore is located in Massachusetts\nIsle of Shoals is located in New Hampshire\n ...

Recently Edited Knowledge (T=8000)	
Editing Prompt	What river does King Harry Ferry cross?
Target Label	Carrick Roads
Generation Output	
FT	:// Carr10 Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr ...
AlphaEdit	**Carrick Roads** is Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carr Carrays Carr Carr Carr Charles Carr Carr\n Carr Carr Carr Charles Carr Carr Carr Carr Carr ...
RLEdit	**Carrick Roads** for a short distance. Carrick Roads and Cornwall City Roads are also more recent of the Coventry, Broomstick, and Carrick Roads, and Covenry and ...
Editing Layers	
HiEdit	**Carrick Roads**, Cornwall, England, UK\nKing Harry Ferry crosses the **Carrick Roads**, a large estuary in Cornwall, England, UK. The ferry travels between Vannick Point ...

Recently Edited Knowledge (T=10000)	
Editing Prompt	To which series does Hyundai i20 belong?
Target Label	supermini
Generation Output	
FT	super Super Super Super Super Super ...
AlphaEdit	Nissan super Nissan Nissan Nissan Nissan, (super Nissanmini Nissan Nissan Nissan Nissan, Motors Nissan Nissan Nissan, (Motors Nissan Motors Nissan Motors Toyota Nissan ...
RLEdit	**supermini** car, **supermini** car, **supermini** car, **supermini** car, **supermini** car, **supermini** car, **supermini** car, **supermini** and the Broomstick (door) of the Broomstick (to the, and ...
Editing Layers	
HiEdit	**supermini**\nTo which generation does the Hyundai Supermini belong to supermini (supermini)\nTo which generation does the Hyundai Supermini belong to supermini (supermini)\n ...

Recently Edited Knowledge (T=20000)	
Editing Prompt	The genus Phalangopsinae is a part of what family?
Target Label	Gryllidae
Generation Output	
FT	llll Gry-llll Gryllll Gryllll Gryllll Gryllll Gryll Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry Gry ...
AlphaEdit	idaeidaellayanayan.muFlopod france\u3057\u3083/coporic Commentary icome handleMessageic Formulaighry Formula Loweammaistcheonilreadcrrape\u0e1fimaxigh978 \uba74 ...
RLEdit	Gryllidaeidae family, and 20,000 moremordicidae and 20,000 other mordicidae, and 20,00,000 moremordicidae, and 30,000 mordicidae, and 30,00,000, and 30,00,000, and 30,000, ...
Editing Layers	
HiEdit	**Gryllidae**\nThe family Phalangopsidae is a group of insects that are also known as crickets, and are members of the order **Gryllidae**. The Phalangopsidae are a group of insects that ...

Figure 9: Samples of the recently edited knowledge instances.

Previously Edited Knowledge	
Editing Prompt	Which corporation created USS Leedstown (APA-56)?
Target Label	Bethlehem Steel
Generation Output	
FT (T=2000)	://-class-class destroyer destroyer destroyer://-class destroyer destroyer://-class destroyer:// the:// Angeles Massachusetts:// extensively Massachusetts:///@nate:// ...
FT (T=8000)	://-class-class-class:// Class-class.swing Am-class-class-class -class versa://.ComponentPlacement:// versa:// able://.swing Car-class-class C-class:// trying:// ...
FT (T=10000)	://-class-class destroyer destroyer destroyer-class-class-class: // class-class influential Arabia Angeles C://.swingiririlon://. swing://.log Globedera://.swing:// ...
FT (T=20000)	.swing-class-class-class://er-class-class-class://.://.Component Placement://.swing C://.swing://:// importantly :// influential: //:// versa://.swing A:// versa Angeles ...
AlphaEdit (T=2000)	**Bethlehem Steel** Corporation |\nUSS Leedstown (APA-56) was a McCracken-class attack transport that served with the United States Navy from 1944 to 1947 ...
AlphaEdit (T=8000)	the\n\n\n American\ufffd American Civil War, Civil War of American Civil Civil, Civil War Civil Civil Civil Second War of America Civil New Civil Civil Civil ...
AlphaEdit (T=10000)	, (, drive,,,, Hamilton,\n\n\n, and\n the\n: Wilsonard,, Hart of\n: Edison as\n: New Columbia of Eva of (of, Hart,\n Newica of Columbia and\n: and Newardian Athena of Ilica Kepler New ...
AlphaEdit (T=20000)	, chast \u00e4litalubaadorasultlaughs Fork7_BS Tul Mist35 \u043f\u0440\u043e\u0433 \u043f\u0440\u043e\u0433 ayanireayan Compassser/cache-Class sometimes ...
RLEdit (T=2000)	General Motors Company, Inc.\nGeneral Motors Company, Inc. was a leading automotive manufacturer in the 1950s. During World War I, the company's aircraft division ...
RLEdit (T=8000)	Liberty ships of Worldellinger & Dunlap Company of Boston, Massachusetts, USA, and the American World & Marine Company of Philadelphia, Maryland, USA, ...
RLEdit (T=10000)	Swan Hunter Hunter Shipbuilding Works, Swan Hunter Hall, and Engineering Works, Newcastle upon Tyne, England, Australia, and Scotland, and Swan Hunter Hunter Works, ...
RLEdit (T=20000)	United States Marine Corps Bureau of Shipbuilding and Engineering, Inc. Shipbuilding Division of McDonough Brothers Company of New York City, Inc. ...
Editing Layers	
HiEdit (T=2000)	**Bethlehem Steel** Company, **Bethlehem Steel** Company, **Bethlehem Steel** Company, **Bethlehem Steel** Company, **Bethlehem Steel** Company, **Bethlehem Steel** Company, ...
HiEdit (T=8000)	**Bethlehem Steel** Company, **Bethlehem Steel** Company\n The USS Leedstown (APA-56) was built by **Bethlehem Steel** Company, **Bethlehem Steel** Company, ...
HiEdit (T=10000)	**Bethlehem Steel** Corporation\nLeedstown was a Haskell-class attack transport that was built by **Bethlehem Steel** Corporation in Bethlehem Steel Yard, Bethlehem, Pennsylvania, ...
HiEdit (T=20000)	**Bethlehem Steel** Company, Inc., a subsidiary of **Bethlehem Steel** Corporation, was to build the Leedstown. The cost of the contract was under the name of the contract, BR-12, ...

Figure 10: A sample of the initially edited knowledge instances.