Efficient and Stable Lifelong Knowledge Editing in LLMs via Neuron-Level Interventions

Anonymous ACL submission

Abstract

Existing "locate-then-edit" approaches, which identify and perturb key parameters, often struggle in sequential editing scenarios, leading to overfitting, catastrophic forgetting, or model collapse. This paper introduces the Precise Neuron-Level Knowledge Editing (PNKE) framework, designed for efficient, low-interference knowledge updates via fine-grained neuron-level interventions. PNKE employs causal attribution to pinpoint background and trigger neurons tied to target knowledge, followed by an entropy-guided sparse masking mechanism to select a critical neuron subset for targeted parameter updates. Our PNKE ensures editing precision while dynamically adjusting sparsity to maintain model stability during lifelong editing. In extensive lifelong editing experiments, PNKE outperforms state-of-the-art methods, achieving an editing success rate (Rel.) of 0.936, generalization (Gen..) of 0.891, and locality (Loc.) of 0.952 on benchmarks like ZsRE and CounterFact. After 5,000 edits, PNKE sustains robust performance on tasks such as MMLU and GSM8K, underscoring its stability and practical utility for continuous knowledge integration in LLMs.

1 Introduction

Large-scale language models (LLMs)(LLAMA, 2024; Devlin et al., 2019; Brown et al., 2020; Vaswani et al., 2017) exhibit remarkable capabilities in knowledge storage and retrieval(Petroni et al., 2019; Guu et al., 2020), but they often generate erroneous or outdated information(Gautam et al., 2024; Ji et al., 2023), known as "hallucinations". To address this issue, model editing techniques have emerged to enable continuous and dynamic updates, corrections, or removal of sensitive content from model knowledge(Cao et al., 2021; Ji et al., 2023). Among existing model editing methods, a prominent paradigm is "locate-then-edit(Mitchell et al., 2021; Meng et al., 2022a; Dai et al., 2021; Fang et al., 2025)". This approach first identifies key parameters W associated with specific knowledge using techniques like causal tracing, then modifies these parameters by introducing a perturbation Δ to update the stored knowledge. The primary objective is to minimize the output error on the knowledge to be updated, denoted as e_1. Many studies further incorporate the output error on knowledge to be retained, e_0, into the optimization objective to preserve the model's original performance. The optimization goal can be expressed as:$\min_\Delta(\|(W+\Delta)K_1 - V_1\|^2 + \lambda\|(W+\Delta)K_0 - V_0\|^2)$, where K_1 and V_1 represent the key and value matrices for the knowledge to be updated, and K_0 and V_0 denote the retained knowledge.

Despite some success in knowledge updating, these methods face significant challenges in practical applications, particularly in sequential editing scenarios(Ma et al., 2025; Zhou et al., 2024). To prioritize update success (i.e., minimizing e_1), existing studies often assign greater weight to e_1, with insufficient control over e_0. This strategy makes edited LLMs prone to overfitting the updated knowledge, leading to a distribution shift in the model's internal hidden layer representations. As editing iterations accumulate, this overfitting gradually erodes the model's ability to retain original knowledge and generate coherent sentences, potentially resulting in catastrophic model forgetting or even model collapse(Wang et al., 2023; Shi et al., 2025). As reported by AlphaEdit(Fang et al., 2025), even projecting the perturbation Δ onto the null space of the retained knowledge K_0, i.e., $\Delta' K_0 = 0$, to ensure $(W+\Delta')K_0 = WK_0 = V_0$, the perturbation Δ' applied across entire layers or parameter blocks W remains coarse-grained.

Further research reveals that knowledge representations in Transformer models are highly complex(López-Otal et al., 2025; Zhang et al., 2025). Based on cross-task activation patterns, feed-forward network (FFN) neurons can be cat-

egorized into general neurons $\mathcal{N}_{gen}$ (broadly activated), domain-specific neurons $\mathcal{N}_{dom}$ (activated in a specific domain D), and task-specific neurons $\mathcal{N}_{task}$ (activated only for a specific task t). Differences in neuron activation $Act(n_i, \text{task}_j) > \theta$ across tasks indicate that knowledge is sparsely concentrated in a small set of critical neurons, exhibiting regionalized co-activation patterns. Building on this, we further abstract related neurons into: **Background neurons** N_{bg}: Stably activated under semantically similar prompts P_{sem} with activation A_{stable}, primarily responsible for knowledge retrieval. **Trigger neurons** N_{trig}: Exhibit strong local responses to specific prompts P_{spec}, with high attribution weights $Attr(N_{trig}, P_{spec})$.This finding underscores the necessity of fine-grained interventions tailored to different neuron functions, providing a theoretical foundation for precise knowledge editing. To address these challenges and achieve more precise interventions, this paper proposes the **Precise Neuron-Level Knowledge Editing (PNKE)** framework. The framework first tackles the representation conflicts caused by traditional coarse-grained editing by using causal attribution(Chattopadhyay et al., 2019; Sundararajan et al., 2017a,b) f_{causal_attr} to precisely identify the set of background and trigger neurons critical to specific knowledge K_{target}, forming an initial causal neuron set $N_{causal} = f_{causal_attr}(K_{target}, \{N_{bg}, N_{trig}\})$. Next, PNKE innovatively employs an entropy-based dynamic sparse masking mechanism $M_{entropy}$ to select the most critical neuron subset $N_{critical} = M_{entropy}(N_{causal})$ from N_{causal}, applying updates $\Delta W_{critical}$ only to parameters $W_{critical}$ associated with $N_{critical}$. This ensures precision and minimal interference at the neuron level. Finally, the adaptive mask $M_{entropy}$ dynamically adjusts sparsity based on the entropy characteristics of neuron importance distributions(Frankle and Carbin, 2019), optimizing the editing scope and supporting robust lifelong editing with reduced impact on the model's general capabilities. Comprehensive lifelong editing experiments demonstrate that PNKE outperforms state-of-the-art methods in both editing success accuracy and general capability preservation for knowledge integration in LLMs.

Our **main** contributions are: i) A causal attribution function $f_{\text{causal\_attr}}$ that identifies background N_{bg} and trigger N_{trig} neurons for K_{target}, yielding $N_{\text{causal}} = f_{\text{causal\_attr}}(K_{\text{target}}, \{N_{bg}, N_{trig}\})$, and eliminating coarse-grained conflicts; ii) An entropy-guided mask M_{entropy} that selects $N_{\text{critical}} = M_{\text{entropy}}(N_{\text{causal}})$ and updates only W_{critical}, ensuring neuron-level precision; iii) Dynamic sparsity via neuron-importance entropy, which tunes M_{entropy} to balance lifelong editing robustness and overall performance.

2 Related Work

Model Editing Paradigms. Current model editing primarily follows the "locate-then-edit" paradigm. MEND (Mitchell et al., 2021) trains a meta-editor network to generate parameter updates. ROME (Meng et al., 2022b) identifies the storage location of knowledge in the feed-forward network (FFN) (Hendrycks and Gimpel, 2023)layers of Transformer models, directly modifying critical weight matrices. MEMIT (Meng et al., 2023) extends ROME to support batch editing of multiple knowledge entries. KN (Dai et al., 2022) treats knowledge as low-rank updates to maintain coherence between pre- and post-edit knowledge.

Editing Granularity and Representation Conflicts. The issue of representation conflicts caused by coarse-grained editing has gained attention. AlphaEdit (Fang et al., 2025) identifies representation conflicts in retaining knowledge, proposing to project parameter perturbations onto the null space of retained knowledge and adopting a batch strategy with batch size 100. MeLLo (Zhong et al., 2023) uses a memory matrix to store edit information. CALM (Tessler et al., 2023) improves representations via adversarial learning.

Neuron Functionality and Knowledge Representation. Studies on the functionality of neurons within Transformer models provide a theoretical foundation for fine-grained editing. (Geva et al., 2021) finds that FFN neurons can be categorized into general, domain-specific, and task-specific types based on activation patterns. (Meng et al., 2023) demonstrates that knowledge exhibits sparse distribution characteristics in models. (Nanda et al., 2023) and (Olsson et al., 2022) further reveal the hierarchical organization of knowledge embeddings.

3 Methodology

Problem Definition. A language model can be viewed as a function $f_W(P) \to O$. Model editing seeks to learn a parameter perturbation Δ such that the updated model $f_{W+\Delta}$ produces the desired knowledge V_1 for specific inputs P_{edit}, while maintaining original performance V_0 on retained

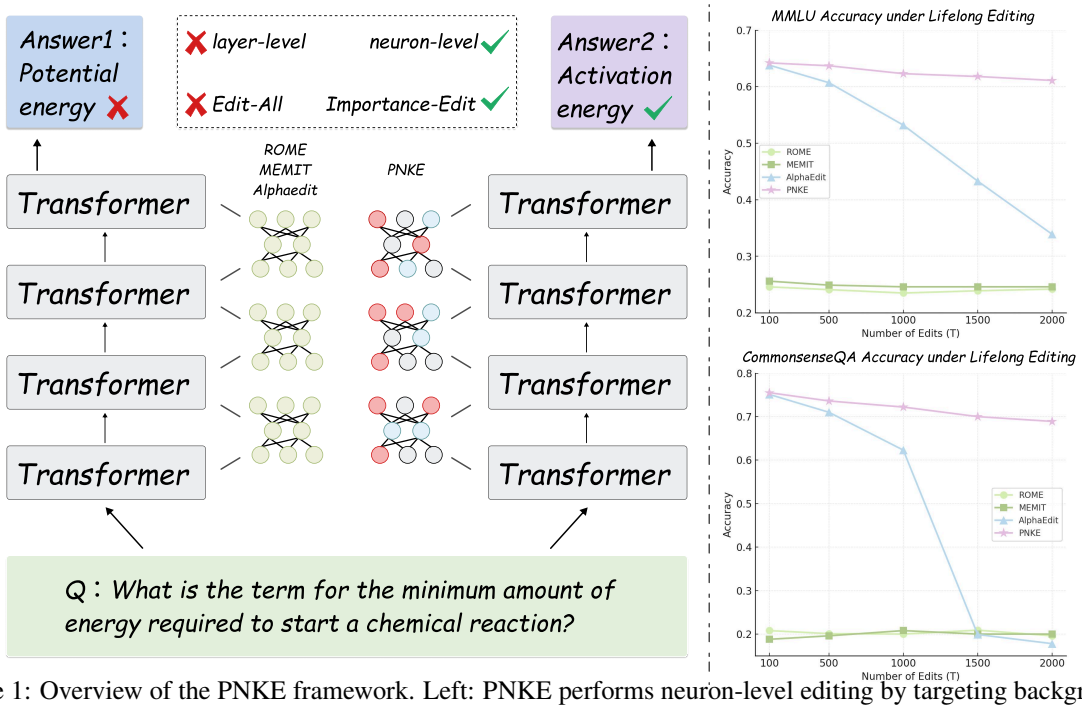

Figure 1: Overview of the PNKE framework. Left: PNKE performs neuron-level editing by targeting background and trigger neurons, offering higher precision than layer-level methods (e.g., ROME, MEMIT, AlphaEdit). Right: Under lifelong editing (T = 10 to 2000), PNKE outperforms baselines on MMLU and CommonsenseQA, demonstrating superior robustness and generalization.

knowledge K_0. Traditional methods often optimize the following objective, where W refers to parameters of relevant layers (e.g., FFN layers):

$$\mathcal{L}(\Delta) = \underbrace{\|(W+\Delta)K_1 - V_1\|_F^2}_{\mathcal{L}_{edit}\ \text{(edit loss)}} + \lambda \underbrace{\|(W+\Delta)K_0 - V_0\|_F^2}_{\mathcal{L}_{preserve}\ \text{(preserve loss)}} \tag{1}$$

Applying Δ to the entire W or its coarse-grained sub-blocks introduces representation conflicts, overfitting, and catastrophic forgetting. As in Figure 2, PNKE addresses these challenges via:

Causal Neuron Identification. To identify neurons critical to specific knowledge K_{target} (triggered by prompt P_{spec}), we distinguish and identify two neuron types: *Background Neurons (N_{bg})*: These neurons exhibit stable, above-baseline activation $Act(n_i, p)$ across multiple semantically similar prompts $P_{sem} = \{p_{sem}^{(1)}, \ldots, p_{sem}^{(m)}\}$. Let $\mathcal{N}$ denote the set of all neurons in a layer. The background neurons are:

$$N_{bg}(K_t) = \left\{ n_i \in \mathcal{N} \mid \underbrace{\mathbb{E}_{p\in P}[Act(n_i,p)] > \theta_{bg}}_{\text{suff. avg. act.}} \wedge \underbrace{\text{std}_{p\in P}(Act(n_i,p)) < \epsilon_s}_{\text{stable act.}} \right\} \tag{2}$$

where θ_{bg_act} and ϵ_{stable} are the activation and stability thresholds, respectively.

Trigger Neurons (N_{trig}): These neurons show strong activation for the specific prompt P_{spec} and have high causal attribution weights $Attr(n_i, P_{spec})$ (e.g., computed via Integrated Gradients). They are defined as:

$$N_{trig}(K_{target}) = \left\{ n_i \in \mathcal{N} \mid \underbrace{Act(n_i, P_{spec}) > \theta_{trig\_act}}_{\text{strong activation for specific prompt}} \wedge \underbrace{Attr(n_i, P_{spec}) > \theta_{attr}}_{\text{high attribution weight}} \right\} \tag{3}$$

where θ_{trig_act} and θ_{attr} are the activation and attribution thresholds, respectively. The initial causal neuron set N_{causal} recte is formed as:

$$\begin{aligned} N_{causal}(K_{target}) &= f_{causal\_attr}(K_{target}, \{N_{bg}, N_{trig}\}) \\ &= N_{bg}(K_{target}) \cup N_{trig}(K_{target}) \end{aligned} \tag{4}$$

Critical Neuron Selection ($M_{entropy}$). To further focus on the most essential neurons, PNKE introduces a dynamic sparse masking mechanism $M_{entropy}$ to select $N_{critical}$ from N_{causal}.

Neuron Importance Quantification. For each $n_i \in N_{causal}$, the importance score s_i is:

$$s_i = \underbrace{\alpha \cdot \text{norm}(Act(n_i, P_{spec}))}_{\text{activation contribution}} + \underbrace{(1-\alpha) \cdot \text{norm}(Attr(n_i, P_{spec}))}_{\text{attribution contribution}} \tag{5}$$

where $\text{norm}(\cdot)$ is a normalization function, and $\alpha \in [0, 1]$ is a balancing coefficient.

Entropy of Importance Distribution. Based on $\{s_i\}$, a normalized probability distribution $P_S = \{p_i = s_i / \sum_j s_j \mid n_i \in N_{causal}\}$ is constructed.

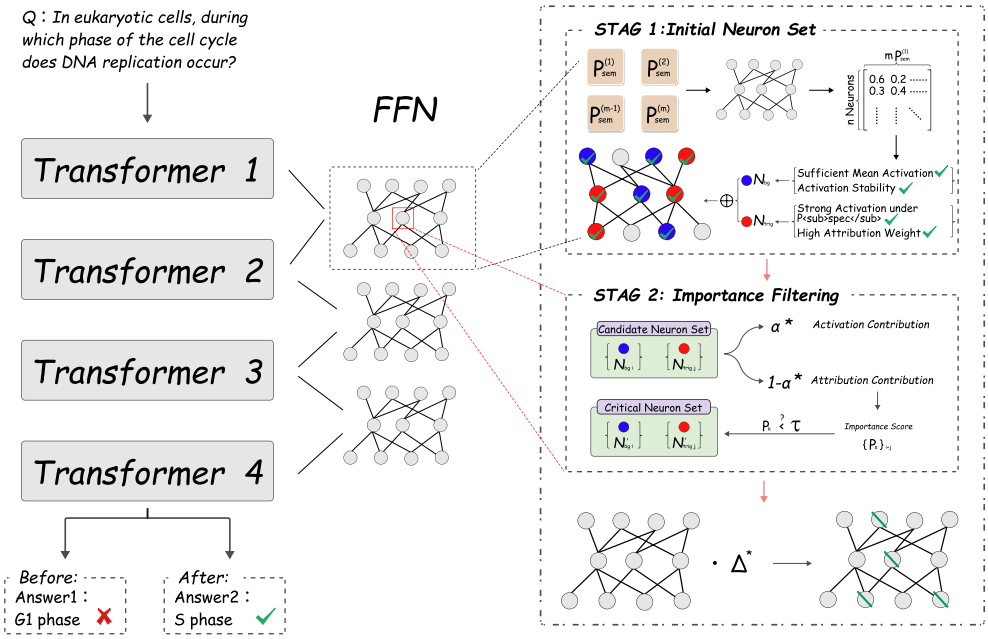

Figure 2: Overview of the PNKE framework, illustrating the three core steps: causal neuron identification, critical neuron selection, and targeted neuron parameter editing.

Its entropy is: $H(P_S) = -\sum_{n_i \in N_{causal}} p_i \log p_i$ $H(P_S)$ reflects the concentration of importance: low entropy indicates importance concentrated in a few neurons, while high entropy suggests a more dispersed distribution.

Dynamic Sparse Selection. $M_{entropy}$ leverages $H(P_S)$ to dynamically adjust the selection strategy. A dynamic threshold τ_H is set as a percentile of the importance scores $\{s_j\}_{j \in N_{causal}}$ as:

$$\tau_H = \text{Percentile}\left(\underbrace{\{s_j\}_{j \in N_{causal}}}_{\text{importance scores of causal neurons}}, \underbrace{q(H(P_S))}_{\text{entropy-determined percentile}} \right) \tag{6}$$

where $q(H(P_S))$ is a function of entropy, e.g., $q(H) = q_{base} + \gamma \cdot (\log |N_{causal}| - H(P_S))$. Lower entropy (more concentrated importance) results in a higher $q(H)$, leading to a higher τ_H and thus fewer, more elite neurons selected. The critical neuron subset is:

$$\begin{aligned} N_{critical} &= M_{entropy}(N_{causal}, \{s_i\}) \\ &= \{n_i \in N_{causal} \mid s_i > \tau_H\} \end{aligned} \tag{7}$$

This enables PNKE to adaptively determine the optimal editing granularity, ensuring effective edits while minimizing redundant perturbations.

Targeted Neuron Parameter Editing. After identifying $N_{critical}$, PNKE modifies only the parameters $W_{critical}$ directly associated with these neurons. For a Transformer's FFN layer (with weights $W_{in} \in \mathbb{R}^{d_{model} \times d_{ff}}$, $W_{out} \in \mathbb{R}^{d_{ff} \times d_{model}}$; biases b_{in}, b_{out}), if $N_{critical}$ corresponds to intermediate FFN neurons, $W_{critical}$ includes the columns of W_{in} and elements of b_{in} corresponding to $N_{critical}$, and the rows of W_{out} corresponding to $N_{critical}$. M_{mask} matching the dimensions of the FFN parameters W_{FFN} is constructed with 1s only at positions associated with $N_{critical}$:

$$(M_{mask})_{param\_idx} = \begin{cases} 1 & \text{if } param\_idx \text{ is associated with } N_{critical} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The parameter update Δ_{FFN} is constrained to the subspace defined by this mask: $\Delta'_{FFN} = \Delta_{FFN} \odot M_{mask}$. The optimization objective from Equation (1) is reformulated in PNKE to solve for Δ_{FFN} under this constraint:

$$\min_{\Delta_{FFN}} \mathcal{L}\left(\underbrace{\Delta_{FFN} \odot M_{mask}}_{\text{update applied only to critical parameters}} \right) \tag{9}$$

Alternatively, existing editing algorithms can be applied to the significantly smaller parameter subspace via $W_{critical}$. For instance, if editing is treated as modifying $W_{out,critical}$ (rows of W_{out} corresponding to $N_{critical}$), with activations of $N_{critical}$ on edit samples K_1 and retain samples K_0 denoted as $h_{1,critical}$ and $h_{0,critical}$, respectively, the optimization problem becomes:

$$\begin{aligned} \min_{\Delta W_{out,critical}} \Big[& \underbrace{\|(W_{out,critical} + \Delta W_{out,critical}) h_{1,critical} - V'_1\|^2}_{\text{edit loss for critical activations}} \\ & + \lambda \underbrace{\|(W_{out,critical} + \Delta W_{out,critical}) h_{0,critical} - V'_0\|^2}_{\text{preserve loss for critical activations}} \Big] \end{aligned} \tag{10}$$

where V_1', V_0' are the target outputs or their changes at the W_{out} layer. This targeted editing significantly reduces interference with the model's overall functionality, enhancing edit robustness and the long-term maintainability of model knowledge.

4 Experiments

Evaluation Benchmarks. We adopt two standard benchmark datasets: *CounterFact*(Meng et al., 2022b), for evaluating factual edits, and *ZsRE*(Levy et al., 2017), for relational question-answering tasks. Following prior studies, we report results using three key metrics: **Rel.** (Edit Reliability)(Hartvigsen et al., 2023), which measures whether the knowledge update is successful; **Gen.** (Generalization)(Zhang et al., 2024), which evaluates the model's ability to extend edits to semantically equivalent expressions; and **Loc.** (Locality)(Zhang et al., 2024), which assesses whether irrelevant knowledge remains unaffected. To further evaluate the generalization capability of the edited model, we incorporate five representative downstream tasks covering mathematical reasoning, question answering, and code generation: *MMLU*(Hendrycks et al., 2021), *GSM8K*(Cobbe et al., 2021), *CommonsenseQA*(Talmor et al., 2019), *BBH (Zero-shot)*(Suzgun et al., 2023), and *HumanEval*(Chen et al., 2021).

Baseline Methods. We compare our PNKE against a range of representative baselines, covering both parameter-modification and parameter-preservation paradigms. Specifically, these include Fine-Tuning (FT)(Zhu et al., 2020), Knowledge Neurons (KN), ROME, PMET(Li et al., 2023), MEMIT, WISE(Wang et al., 2024), and AlphaEdit. All methods are evaluated on the *LLaMA3-8B-Instruct*(LLAMA, 2024) model. Sequential edits are performed at pre-defined steps $T = \{10, 100, 500, 1000, 1500, 2000, 2500\}$, and edit success rate and generalization performance are assessed at each stage.

Generalization After Knowledge Editing. As shown in Table 1, experimental results reveal that as the number of edits increases, the performance of existing methods tends to degrade significantly. Specifically, FT nearly fails on tasks such as *GSM8K* and *HumanEval* after merely 100 edits. Similarly, ROME and MEMIT experience substantial performance drops on benchmarks like *MMLU* when the number of edits exceeds $T = 500$. Although AlphaEdit, currently one of the strongest baselines, mitigates early-stage degradation by leveraging a *null-space projection* mechanism, it still relies on hierarchical-level parameter updates. This reliance inevitably accumulates distributional shifts over time, leading to instability and compromised generalization in long-horizon deployment. By contrast, our PNKE demonstrates significantly better robustness and generalization in the multi-round editing scenario, thanks to our fine-grained neuron-level editing strategy.

Do Sparser Neuron-Level Updates Improve Editing Effectiveness? As shown in Table 2, we conduct a systematic evaluation on the *ZsRE* dataset. The experiment is based on a randomly sampled set of 2,000 instances, where edits are applied sequentially with a batch size of 1. Additional results on the *CounterFact* dataset, including case studies and performance trends across editing steps, are provided in Appendix C. The results indicate that traditional methods (e.g., *FT*, *KN*, and *ROME*) experience noticeable performance degradation even at the early stages of continual editing. Notably, *AlphaEdit* initially suppresses interference in non-target regions through its *null-space projection* mechanism. Nevertheless, under large-scale sequential editing, its performance becomes unstable. At $T = 2000$, its rewrite accuracy drops to 0.319, revealing a robustness bottleneck in maintaining effectiveness over time. In contrast, our PNKE consistently outperforms all baselines across key metrics, including rewrite success (**Rel.**), generalization (**Gen.**), and locality (**Loc.**). This demonstrates PNKE's ability to balance edit precision, semantic generalization, and distributional stability in continual knowledge editing. The superior performance can be attributed to PNKE's attribution-guided sparse masking mechanism, which accurately identifies a minimal set of neurons highly relevant to the target knowledge and confines updates within this subspace. This design effectively mitigates distributional drift and enables efficient, low-interference internal representation updates.

Adaptive Neuron Masking Enhances Edit Success and Stability. We systematically evaluate the performance of four neuron selection strategies for our PNKE: (1) using only *trigger neurons*; (2) using only *background neurons*; (3) a fixed-ratio activation selection strategy; and (4) an entropy-based dynamic masking strategy. As shown in Figure 3, while all four strategies are capable of preserving the model's generalization ability to some

Table 1: Performance comparison across five downstream tasks under lifelong editing. PNKE consistently outperforms all baselines in generalization and editing success, especially under long-horizon interventions.

Method	T = 100					T = 500					T = 1000				
	mmlu	gsm8k	commonsense_qa	bbh	humaneval	mmlu	gsm8k	commonsense_qa	bbh	humaneval	mmlu	gsm8k	commonsense_qa	bbh	humaneval
FT	0.376	0	0.465	0.009	0	0.288	0	0.272	0.002	0	0.246	0	0.213	0.002	0
KN	0.2541	0	0.1941	0	0	0.252	0	0.204	0.0003	0	0.252	0	0.204	0.0002	0
ROME	0.2459	0	0.208	0.002	0	0.241	0	0.201	0.001	0	0.235	0	0.200	0.001	0
MEMIT	0.256	0	0.188	0.002	0	0.249	0	0.196	0	0	0.246	0	0.208	0	0
PMET	0.2319	0	0.18	0.149	0.329	0.2439	0	0.186	0.143	0.329	0.24	0	0.195	0.032	0.197
WISE	0.639	0.761	0.76	0.446	0.28	0.514	0.431	0.692	0.394	0.145	0.342	0.221	0.574	0.256	0.086
AlphaEdit	0.638	0.762	0.751	0.441	0.31	0.607	0.724	0.71	0.414	0.304	0.532	0.251	0.623	0.323	0.195
PNKE	**0.642**	**0.758**	**0.755**	**0.4461**	**0.31**	**0.637**	**0.747**	**0.736**	**0.4407**	**0.286**	**0.623**	**0.737**	**0.722**	**0.4309**	**0.2926**
Method	**T = 1500**					**T = 2000**					**T = 2500**				
	mmlu	gsm8k	commonsense_qa	bbh	humaneval	mmlu	gsm8k	commonsense_qa	bbh	humaneval	mmlu	gsm8k	commonsense_qa	bbh	humaneval
FT	0.279	0	0.23	0.0001	0	0.258	0	0.28	0.0003	0	0.223	0	0.014	0	0
KN	0.228	0	0.185	0	0	0.231	0	0.18	0	0	0.213	0	0.096	0	0
ROME	0.239	0	0.209	0.0002	0	0.242	0	0.196	0.0001	0	0.212	0	0.164	0	0
MEMIT	0.246	0	0.2	0	0	0.246	0	0.200	0	0	0.206	0	0.173	0	0
PMET	0.255	0	0.197	0.0005	0.186	0.255	0	0.197	0	0.164	0.196	0	0.154	0	0.142
WISE	0.292	0.089	0.244	0.132	0	0.231	0	0.163	0	0	0.192	0	0.126	0	0
AlphaEdit	0.433	0	0.199	0.111	0	0.339	0	0.178	0.016	0	0.214	0	0.124	0	0
PNKE	**0.618**	**0.717**	**0.7**	**0.4139**	**0.274**	**0.611**	**0.695**	**0.689**	**0.411**	**0.25**	**0.605**	**0.681**	**0.647**	**0.382**	**0.231**

Table 2: Comparison of Rel., Gen., and Loc. metrics on ZsRE under varying editing steps ($T = 10$ to 2000), where PNKE consistently outperforms all baselines.

Step	T = 10			T = 100			T = 500			T = 1000			T = 1500			T = 2000		
Metric	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.
FT	0.183	0.033	0.012	0.166	0.133	0.033	0.119	0.108	0.004	0.128	0.102	0.016	0.119	0.102	0.015	0.072	0.059	0.006
KN	0.133	0.133	0.658	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ROME	0.975	0.975	0.637	0.103	0.085	0.025	0.0053	0.006	0.022	0.0155	0.0136	0.0158	0.0368	0.0354	0.0218	0.0093	0.0086	0.02
MEMIT	0.0346	0.0214	0.0064	0.0316	0.0216	0.0073	0.0438	0.0438	0.031	0.0434	0.034	0.032	0.0438	0.0438	0.034	0.0442	0.0442	0.033
PMET	0.2333	0.183	0.9125	0.0198	0.0165	0.0529	0	0	0	0	0	0	0	0	0	0	0	0
WISE	0.833	0.7833	1	0.7081	0.6748	1	0.4622	0.4478	1	0.4115	0.3877	1	0.3237	0.3079	1	0.3657	0.3564	1
AlphaEdit	0.996	0.952	0.853	0.995	0.947	0.86	0.957	0.874	0.713	0.926	0.84	0.58	0.642	0.539	0.142	0.319	0.283	0.058
PNKE	0.972	0.874	0.942	0.966	0.865	0.921	0.955	0.842	0.823	0.95	0.854	0.769	0.942	0.857	0.741	0.936	0.852	0.705

extent during multi-round editing, demonstrating the potential of fine-grained neuron-level editing in reducing interference, they differ significantly in terms of editing effectiveness. The entropy-based dynamic masking strategy consistently achieves superior performance throughout the editing process. Even at $T = 2000$, it maintains a rewrite accuracy as high as 0.936, demonstrating both high editing precision and strong resistance to interference. This suggests that the entropy-guided adaptive masking strategy dynamically balances the selection between background and trigger neurons, effectively focusing updates on the subspace most relevant to the target knowledge. As a result, it not only ensures high editing precision, but also significantly enhances model stability and generalization—particularly well-suited for applications such as *Lifelong Knowledge Editing*, where long-term reliability is critical.

The Layerwise Distribution of Knowledge Neurons. As illustrated in Figure 4, we conduct a systematic analysis of the distributional characteristics of *background neurons* and *trigger neurons* across layers 0 to 31 in the *LLaMA3* model. This analysis aims to uncover the structural-functional roles and knowledge representation mechanisms embedded across the model hierarchy. The results reveal a clear layerwise aggregation pattern among background neurons, with a strong concentration in higher layers. Notably, layer 31 accounts for the highest proportion of background neurons, reaching a peak of 0.7682, with an average activation rate of 0.8370. These findings suggest that the top layer plays a central role in encoding high-level semantics and integrating global knowledge—consistent with theoretical perspectives that view upper layers as the core for semantic abstraction and conceptual integration. In contrast, Trigger neurons exhibit a more uniform distribution across layers, with a slight reduction in the deeper layers. This trend may indicate a diminished selectivity in higher layers, where the sensitivity of trigger neurons to specific knowledge stimuli declines as semantic abstraction intensifies, thus relying more on the localization capacity of mid- to low-level layers. More critically, we observe that the overlap between background and trigger neurons reaches a local maximum in the middle layers, particularly between layers 10 and 20. This pattern implies that the intermediate layers may serve as a "fusion hub" for knowledge representation, simultaneously integrating general knowledge signals and responding to specific stimuli. Such functional convergence aligns with prior studies that identify intermediate layers in Transformer models as crucial transition zones bridging local semantics and global abstractions, characterized by high representational plasticity and strong knowledge coupling capabilities. It is important to note that we do not perform full-scale editing on

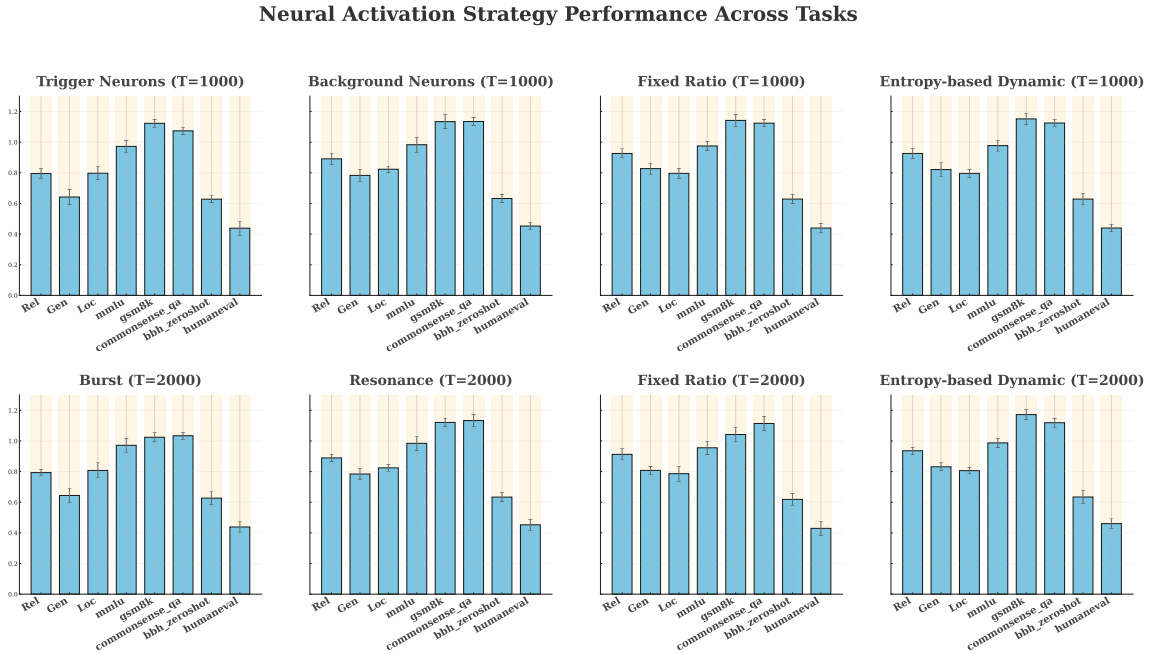

Figure 3: Performance of different neuron selection strategies at $T = 1000$ and $T = 2000$ across five downstream tasks. Entropy-based dynamic masking achieves the best balance between precision and generalization.

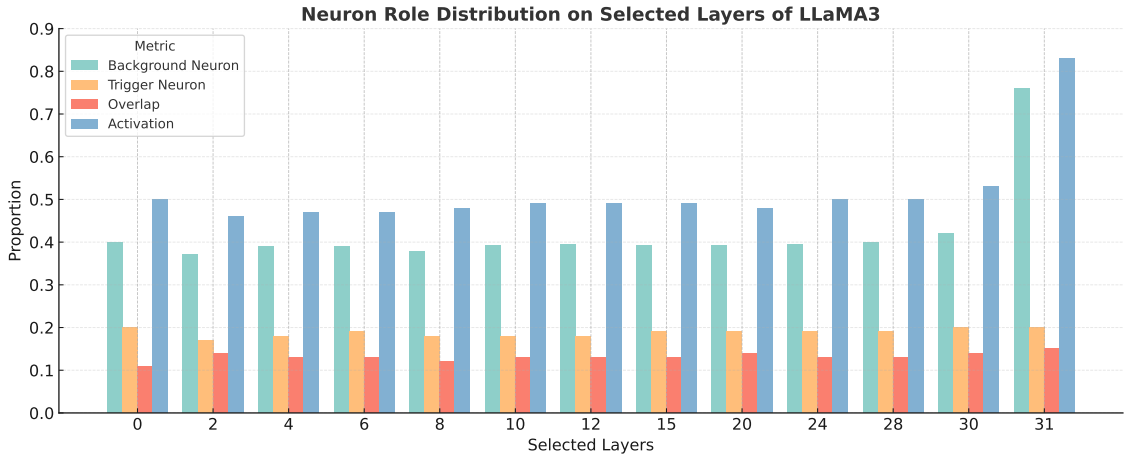

Figure 4: Distribution of neuron roles across selected layers of LLaMA3. Background and trigger neurons are broadly distributed, with increasing overlap and activation density in deeper layers, particularly near layer 31.

all potentially relevant neurons. Instead, we adopt a sparse masking mechanism based on attribution and activation, dynamically selecting a minimal set of neurons highly relevant to the target knowledge. This strategy ensures precise editing with minimal interference, significantly enhancing the specificity of knowledge injection and suppressing redundant perturbations to the global representational space.

Attribution Sensitivity Reveals Tradeoffs in Precision and Generalization. To evaluate the impact of hyperparameter configurations on the performance of knowledge editing, we conduct a sensitivity analysis focusing on two key factors. The first concerns the boundary conditions of neuron activation, specifically, the threshold settings for background and trigger neurons. The second involves the dynamic thresholding strategy used in the entropy-based selection mechanism for identifying critical neurons. Specifically, we adopt the edit reliability metric (Rel.) on the ZsRE dataset as the primary evaluation criterion, systematically analyzing how variations in threshold configurations affect the success rate of knowledge injection, as shown in Figure 5.

Regarding the activation boundaries, we systematically test editing success and generalization performance under varying threshold configurations. Results indicate that moderately relaxing the activation range (e.g., setting the average activation threshold for background neurons to 0.2–0.3, the stable activation threshold to 0.75–0.8, and using 0.2–0.3 for both strong activation and high attribution weight thresholds for trigger neurons) significantly improves the success rate of knowledge

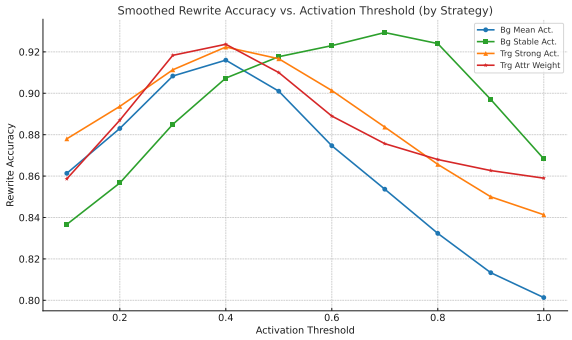

Figure 5: Rewrite accuracy under different activation threshold settings across four neuron types.

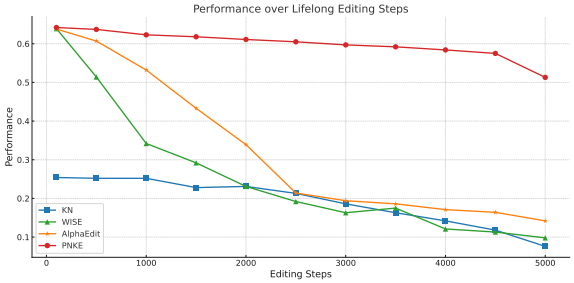

Figure 6: Performance degradation over lifelong editing steps, where PNKE maintains high stability compared to other methods.

injection. This suggests that activating a broader set of neurons helps cover representations more relevant to the target knowledge. However, expanding the editing scope also introduces trade-offs. In certain configurations, we observe slight declines in generalization ability and local consistency (e.g., rewrite accuracy and locality metrics). This indicates that involving too many marginal neurons may introduce irrelevant signals, potentially undermining the model's original knowledge structure. These findings align with our previous observations on the hierarchical distribution of knowledge neurons—while a wider activation range facilitates editing success, it also increases the risk of interference and conflicts during editing.

To enhance the precision and effectiveness of neuron selection, we incorporate a dynamic entropy-based masking mechanism. By increasing the entropy scaling factor, we amplify the contrast between critical and non-critical neurons in terms of attribution scores. Experimental results show that moderate increases in this factor improve the mask's selection accuracy, boosting editing efficiency while minimizing unnecessary perturbations.

Additionally, we find that model scale plays a significant role in determining the demand for activation strategies: smaller models typically require a higher proportion of activated neurons to ensure editing effectiveness, whereas larger models maintain strong performance even under lower activation ratios. This observation suggests a synergistic relationship between model capacity and mask sparsity.

Scaling to 5,000 Edits: Evaluation of Lifelong Robustness.

As illustrated in Figure 6, we scale the knowledge editing task on the LLaMA3 model up to 5,000 steps to systematically evaluate the robustness and generalization capabilities of different methods in a long-horizon editing scenario. The evaluation covers four representative approaches: AlphaEdit, WISE, KN, and our proposed method PNKE. To comprehensively assess the model's ability to retain general capabilities under large-scale interventions, we incorporate the MMLU (Massive Multitask Language Understanding) benchmark to track performance across different rounds of editing.

Experiments show that PNKE outperforms existing methods in editing success and generalization retention, especially at $T = 3{,}000$ and $T = 5{,}000$. While AlphaEdit and WISE degrade significantly in later stages, PNKE maintains higher accuracy (above 0.51 at $T = 5{,}000$), demonstrating superior scalability and stability for long-term knowledge editing. PNKE preserves generalization during intensive editing by precisely updating only the most relevant neurons, minimizing parameter drift and maintaining model accuracy.

5 Conclusion

Precise Neuron-Level Knowledge Editing (PNKE) is a framework for editing large language models (LLMs) that addresses issues like overfitting and catastrophic forgetting, especially in continual editing scenarios. PNKE works by accurately identifying neurons tied to the target knowledge, enabling efficient and minimally invasive updates. Its process includes: (1) causal neuron identification using attribution methods; (2) critical neuron selection via an entropy-based approach to isolate a sparse set of key neurons; and (3) targeted editing, updating only these neurons' parameters to preserve the model's overall behavior.

Limitations

While PNKE demonstrates substantial improvements in editing precision and representational locality, it still faces several intrinsic limitations:

Reliance on Neuron Attribution Reliability: The effectiveness of PNKE significantly depends on the reliability of neuron attribution methods. Since these methods inherently approximate model internals, errors in identifying background or trigger neurons can propagate to the editing stages, potentially leading to unintended parameter drift or partial knowledge overwrite.

Hyperparameter Calibration and Stability of Sparse Masking: The entropy-based sparse mask construction requires careful hyperparameter calibration. Furthermore, its stability across different tasks, model scales, and domains has not yet been sufficiently understood.

Scope of Validation and Generalizability: PNKE has been primarily validated on single-hop factual edits within static textual models. Its capability to generalize to settings that involve multi-modal representations, compositional reasoning, or temporally evolving knowledge has yet to be established.

Latent Representational Shifts and Long-Term Issues: Although localized updates reduce interference with unrelated knowledge, they might also induce latent shifts in representation manifolds. These shifts can accumulate over long editing trajectories, posing open questions regarding the reversibility of edits, long-term robustness, and compatibility with continual pretraining paradigms.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

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

Aditya Chattopadhyay, Piyushi Manupriya, Anirban Sarkar, and Vineeth N. Balasubramanian. 2019. Neural network attributions: A causal perspective. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 550–559.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. In *Proceedings of the 2022 Annual Meeting of the Association for Computational Linguistics (ACL)*, page 582–593.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8493–8502.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the ACL: Human Language Technologies (NAACL-HLT)*, pages 4171–4186. Association for Computational Linguistics.

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. *Preprint*, arXiv:2410.02355.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*. OpenReview.net.

Pranav Gautam, Narayanan Venkit, Zihao Ji, and 1 others. 2024. An audit on the perspectives and challenges of hallucinations in nlp. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–391. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on*

Machine Learning (ICML), volume 119, pages 3929–3938. PMLR.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Preprint*, arXiv:2211.11031.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus). *Preprint*, arXiv:1606.08415.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

LLAMA. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Miguel López-Otal, Jorge Gracia, Jordi Bernad, Carlos Bobed, Lucía Pitarch-Ballesteros, and Emma Anglés-Herrero. 2025. Linguistic interpretability of transformer-based language models: A systematic review. *arXiv preprint arXiv:2504.08001*.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2025. Perturbation-restrained sequential model editing. In *Proceedings of the 2025 International Conference on Learning Representations (ICLR)*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 18350–18364.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022b. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36. ArXiv:2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *arXiv preprint*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Preprint*, arXiv:2209.11895.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Alex Bakhtin, Yu Wu, Alexander H Miller, Andreas Vlachos, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages z2463–2473. Association for Computational Linguistics.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. 2025. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.* Just Accepted.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017a. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3319–3328.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017b. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. 2023. Calm: Conditional adversarial latent models for directable

virtual characters. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, page 1–9. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Preprint*, arXiv:2405.14768.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. 2025. Kabb: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. *Preprint*, arXiv:2502.07350.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, and 3 others. 2024. A comprehensive study of knowledge editing for large language models. *Preprint*, arXiv:2401.01286.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

Xinyi Zhou, Meng Li, Wei Xu, and Jie Sun. 2024. Can we continually edit language models? on the knowledge attenuation after sequential editing. In *Findings of the 2024 Conference of the Association for Computational Linguistics (ACL)*, pages 323–339.

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

A Implementation Details

A.1 Description of Datasets

To comprehensively evaluate the general capabilities of language models across complex tasks, we adopt five widely used downstream benchmarks, covering knowledge understanding, logical reasoning, and code generation.

MMLU(Hendrycks et al., 2021) (Massive Multi-task Language Understanding) is a multiple-choice benchmark consisting of over 16,000 questions across 57 academic and professional subjects, including mathematics, history, law, and medicine. It assesses the model's ability to perform cross-domain knowledge retrieval and multi-field reasoning.

GSM8K (Grade School Math 8K) includes approximately 8,500 math word problems designed at the elementary school level. The benchmark evaluates the model's step-by-step arithmetic reasoning and numerical computation abilities.

CommonsenseQA is a multiple-choice question answering task focused on commonsense reasoning. Each sample consists of a natural language question with five candidate answers, and the model must select the most logically consistent one. This benchmark measures the model's ability to understand everyday scenarios and implicit context.

BBH-Zeroshot is a high-difficulty subset of 23 tasks selected from the BIG-Bench benchmark, spanning logical reasoning, mathematical computation, and code understanding. It is evaluated in a zero-shot setting to examine the model's generalization and reasoning ability on unseen tasks.

HumanEval is a code generation benchmark containing 164 Python programming problems. Each problem provides a function signature, docstring, and input-output examples. The model must generate functionally correct code that passes unit tests, thereby assessing its programming proficiency and semantic correctness.

In addition, to evaluate factual knowledge editing, we adopt two standard benchmarks:

ZsRE is a relation-centric question answering dataset. Each sample includes an edit prompt, a paraphrased variant for generalization testing, and an unrelated locality prompt to assess specificity and non-interference.

CounterFact constructs factual and counterfactual pairs by replacing the subject entity while keeping the predicate fixed. It is used to test whether the model can accurately incorporate new facts, generalize to paraphrased forms, and preserve unrelated knowledge.

A.2 Evaluation Metrics

To comprehensively assess the effectiveness and robustness of knowledge editing, we adopt three standard evaluation metrics: **Rel** (Edit Success), **Gen** (Generalization), and **Loc** (Locality Preservation). These metrics are computed on the editing dataset $\mathcal{D}_{\text{edit}} = \{(x_e^{(t)}, y_e^{(t)}, x_{e'}^{(t)}, x_{\text{loc}}^{(t)})\}_{t=1}^{T}$, where each sample consists of an edit query $x_e^{(t)}$ with the corresponding target output $y_e^{(t)}$, a semantically equivalent paraphrased variant $x_{e'}^{(t)}$ for generalization testing, and a locality probe $x_{\text{loc}}^{(t)}$ to evaluate non-interference with unrelated knowledge.

Given the post-edit model f_{Θ_T}, the three metrics are formally defined as:

$$\begin{aligned}
\text{Rel.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\{f_{\Theta_T}(\mathbf{x}_e^t) = \mathbf{y}_e^t\}, \\
\text{Gen.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\{f_{\Theta_T}(\mathbf{x}_{e'}^t) = \mathbf{y}_e^t\}, \\
\text{Loc.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\{f_{\Theta_T}(\mathbf{x}_{\text{loc}}^t) = f_{\Theta_0}(\mathbf{x}_{\text{loc}}^t)\},
\end{aligned} \tag{11}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function, and f_{Θ_0} is the original (pre-edit) model. Specifically, **Rel** measures whether the model generates the correct output for the edited query, **Gen** tests whether the edit generalizes to paraphrased variants, and **Loc** evaluates whether the model preserves its original behavior on unrelated inputs, thereby reflecting locality and non-interference.

A.3 Descriptions of Compared Model Editors

We compare our approach against a suite of representative knowledge editing methods, which can be broadly categorized into two classes: *parameter-modifying* methods that directly alter the model weights, and *parameter-preserving* methods that achieve editing through external mechanisms without changing the base model.

FT (Fine-tuning) serves as a basic parameter-modifying baseline that updates model parameters using standard autoregressive loss on the edit instance. Despite its simplicity, FT often causes extensive parameter drift and suffers from poor locality due to overfitting.

KN identifies a subset of neurons most relevant to the target fact using attribution techniques and fine-tunes only those neurons. While this approach reduces the scope of parameter changes, it still operates via direct weight updates.

ROME (Rank-One Model Editing) performs closed-form rank-one updates on the MLP weight matrices identified via causal tracing. This method enables localized and efficient interventions, representing a structured and analytically grounded editing technique.

PMET (Precise Model Editing Transformer) formulates editing as a constrained optimization problem and solves for minimal weight changes required to induce the desired output. Unlike ROME's analytical formulation, PMET employs gradient-based methods, offering greater flexibility for complex editing scenarios.

MEMIT extends ROME to support multi-fact editing by computing simultaneous low-rank updates across multiple MLP layers. This allows efficient batch editing of hundreds or thousands of facts, making it well-suited for high-throughput use cases.

AlphaEdit (ours) also performs parameter-modifying edits but incorporates a null-space projection mechanism. It suppresses directions that interfere with unrelated knowledge by projecting the learned update into a minimally invasive subspace, thereby enhancing both precision and generalization.

In contrast, the only parameter-preserving method we compare is:

WISE, which introduces an external memory module to store edits and employs a learned router to dynamically decide whether to use original or edited outputs during inference. This design avoids any direct modification to the base model, achieving strong locality and scalability.

In summary, FT, KN, ROME, PMET, MEMIT, and AlphaEdit implement editing via direct weight modification, while WISE achieves non-intrusive editing through auxiliary routing without altering the original model parameters.

Critical Neuron Attribution Methods and Hyperparameters

B Strategy for Generating P_{sem} (Semantically Similar Prompts) and Parameter Settings

To identify background neurons N_{bg}, the PNKE framework utilizes a set of semantically similar prompts $P_{\text{sem}} = \{p_{\text{sem}}^{(1)}, ..., p_{\text{sem}}^{(m)}\}$.

B.1 Strategy for Generating P_{sem}

The construction of P_{sem} aims to comprehensively cover the core semantics of the target knowledge K_{target} while introducing diversity in expression. The specific generation process is as follows:

1. **Template Construction:** For the target knowledge, standard declarative sentences are manually designed or extracted from datasets to serve as base templates.

2. **Paraphrase Generation:** Leveraging the paraphrasing capabilities of pre-trained language models (e.g., LLaMA3-8B-Instruct), the base templates are diversely rephrased to generate a set of prompts that are semantically equivalent but differ in syntactic structure or wording.

3. **Back-Translation:** To further increase diversity, some templates undergo back-translation ("source language $\rightarrow$ intermediate language $\rightarrow$ source language") using high-quality machine translation systems.

In this study, for each target knowledge K_{target}, $m = 10$ semantically similar prompts are generated to form P_{sem}. This number was determined in preliminary experiments as the optimal trade-off point by evaluating the stability of N_{bg} identification and the final editing performance (Rel, Gen, Loc metrics) for different values of m (ranging from 5 to 15).

B.2 Threshold Parameters in the Definition of N_{bg}

Background neurons are defined in Equation (2) of the main paper as $N_{\text{bg}}(K_t) = \{n_i \in \mathcal{N} | \frac{\sum_{p \in P_{\text{sem}}} \text{Act}(n_i,p)}{m} > \theta_{\text{bg\_act}} \wedge \text{std}_{p \in P_{\text{sem}}}(\text{Act}(n_i, p)) < \epsilon_{\text{stable}}\}$.

- $\theta_{\text{bg\_act}}$ **(Average Activation Threshold):** This threshold is used to filter neurons

that consistently exhibit significant activation across the P_{sem} set. $\theta_{\text{bg\_act}}$ is set to the 75th percentile of the average activation value distribution of all neurons in the corresponding layer over P_{sem}. This setting ensures that the selected neurons have a relatively high average activation level compared to other neurons in that layer.

- ϵ_{stable} **(Activation Standard Deviation Threshold):** This threshold ensures that neurons exhibit consistent activation patterns for different prompts within P_{sem}. ϵ_{stable} is set to the 25th percentile of the activation standard deviation distribution of all neurons in the corresponding layer over P_{sem}. This guarantees that the selected background neurons respond stably to semantically similar but differently phrased inputs.

The principles for setting these thresholds were derived through systematic evaluation (as detailed in the experimental section of the main paper, e.g., the sensitivity analysis of activation thresholds shown in Figure 5), aiming to maximize the effectiveness of subsequent edits and the stability of the model.

B.3 Selection of Causal Attribution Method and Parameter Settings

The identification of trigger neurons N_{trig} and the calculation of neuron importance scores s_i both utilize causal attribution weights $\text{Attr}(n_i, P_{\text{spec}})$.

B.3.1 Selection of Causal Attribution Method

This study employs **Integrated Gradients (IG)** as the method for computing neuron causal attribution weights. The choice of IG is based on its established theoretical properties (e.g., completeness, sensitivity) and its widespread application and validation in explaining the internal mechanisms of deep learning models, including large language models. IG provides a quantitative measure of the contribution of each neuron to the model's output for a specific input P_{spec}.

B.3.2 Limitations of IG

The application of IG requires the definition of a baseline input. In this study, the baseline for neuron activation is set to zero activation. While IG offers effective attribution analysis, its results can be influenced by the choice of baseline, and its explanatory power for highly non-linear systems has inherent limitations due to its linear integration path.

B.3.3 Threshold Parameters in the Definition of N_{trig}

Trigger neurons are defined in Equation (3) of the main paper as $N_{\text{trig}}(K_{\text{target}}) = \{n_i \in \mathcal{N} | \text{Act}(n_i, P_{\text{spec}}) > \theta_{\text{trig\_act}} \wedge \text{Attr}(n_i, P_{\text{spec}}) > \theta_{\text{attr}}\}$.

- $\theta_{\text{trig\_act}}$ **(Strong Activation Threshold):** Used to filter neurons that exhibit a strong activation response to the specific editing prompt P_{spec}. This threshold is set to the 90th percentile of the activation value distribution of neurons in the target layer for P_{spec}.

- θ_{attr} **(High Attribution Weight Threshold):** Used to filter neurons that are not only highly activated but also whose activation makes a highly causal contribution to the model's output for P_{spec}. This threshold is set to the 90th percentile of the attribution weight distribution computed by IG.

These threshold settings are designed to precisely identify a small number of neurons with strong signals that are highly relevant to the specific knowledge point. Their effectiveness has been validated in the ablation studies presented in the main paper (see particularly the discussion related to Figure 5).

B.4 Hyperparameter Settings in the Entropy Mechanism

The entropy-guided critical neuron selection mechanism M_{entropy} depends on the calculation of neuron importance scores s_i and the determination of the dynamic selection threshold τ_H.

B.4.1 Equation (5) (Importance Score s_i): The norm() Function and Balancing Coefficient α

- **The norm() Function:** In the calculation of the importance score $s_i = \alpha \cdot \text{norm}(\text{Act}(n_i, P_{\text{spec}})) + (1 - \alpha) \cdot \text{norm}(\text{Attr}(n_i, P_{\text{spec}}))$, the norm() function employs **min-max normalization**. Specifically, the activation values $\text{Act}(n_i, P_{\text{spec}})$ and attribution weights $\text{Attr}(n_i, P_{\text{spec}})$ are independently normalized within the set of corresponding values for all N_{causal} neurons in their layer, mapping them to the $[0, 1]$ interval. This operation ensures that the activation contribution and attribution contribution have a uniform and comparable scale before weighted summation.

- **Balancing Coefficient α:** This coefficient is used to weigh the relative contributions of activation intensity and attribution weight in the assessment of neuron importance. In this study, α is set to 0.5. This value was determined in preliminary experiments by testing different α values (range $[0.1, 0.9]$, step 0.1) on a validation set for their impact on editing performance, aiming to equally value both activation signals and causal attribution information.

B.4.2 Equation (6) (Definition of τ_H): The $q(H(P_S))$ Function, q_{base}, and γ

The dynamic threshold $\tau_H = \text{Percentile}(\{s_j\}_{j \in N_{\text{causal}}}, q(H(P_S)))$ is determined by the entropy-based function $q(H(P_S)) = q_{\text{base}} + \gamma \cdot (\log |N_{\text{causal}}| - H(P_S))$. The output of the function $q(H(P_S))$ is a percentile value, mapped to the range $[0, 100]$, used to select the threshold from the importance score distribution $\{s_j\}$.

- **q_{base} (Base Percentile):** Represents the base selection percentile adopted when the importance distribution is most dispersed (i.e., entropy $H(P_S)$ reaches its maximum value $\log |N_{\text{causal}}|$). In this study, q_{base} is set to 85. This implies that even in cases of highly dispersed importance, PNKE still selects neurons whose scores are in the top 15% (i.e., above the 85th percentile).

- **γ (Entropy Adjustment Factor):** Controls the sensitivity of the selection threshold to the entropy $H(P_S)$. $\gamma > 0$ ensures that when the importance distribution is more concentrated (smaller entropy), a more elite subset of neurons is selected (i.e., a higher percentile threshold). In this study, γ is set to 10.0. This value was determined by evaluating the combined impact of different γ values on the size of $|N_{\text{critical}}|$ and editing performance on a validation set.

These parameter settings enable PNKE to adaptively adjust the sparsity/granularity of editing based on the concentration of the current knowledge point's representation among neurons.

B.5 Computational Efficiency and Scalability of Neuron-Level Editing

The computational efficiency of PNKE is primarily determined by its three core steps: causal neuron identification, critical neuron selection, and targeted neuron parameter editing.

B.5.1 Composition of Computational Costs

1. **Causal Neuron Identification (N_{causal}):**
   - N_{bg} identification involves m forward passes through the target layers.
   - N_{trig} identification involves one forward pass for the specific prompt P_{spec} and one backward pass process based on Integrated Gradients (including multiple model evaluations for its path integration).

   This step is the main source of computational overhead in PNKE, with its cost being proportional to m, the number of path integration steps in IG, the number of target layers, and the model depth.

2. **Critical Neuron Selection (N_{critical}):** This step includes calculating importance scores for $|N_{\text{causal}}|$ neurons, normalization, entropy calculation, and threshold selection based on percentiles. These operations are primarily vector and a few scalar computations, with computational costs far lower than the neuron identification phase, and roughly linear or quasi-linear with the size of $|N_{\text{causal}}|$.

3. **Targeted Neuron Parameter Editing ($\Delta W_{\text{critical}}$):** Editing operations are confined to the parameter subset W_{critical} directly associated with $|N_{\text{critical}}|$ neurons. If using the method from Equation (10) of the main paper (i.e., applying existing efficient editing algorithms like ROME or MEMIT to the subspace), its computational cost is primarily that of these algorithms on a significantly reduced parameter subset. For example, for a ROME-like solution, its complexity is cubic with respect to the dimension of h_{critical} (i.e., $|N_{\text{critical}}|$), which is far smaller than the original FFN dimension d_{ff}. Therefore, the computational efficiency of this stage is significantly better than methods that edit parameters at the full layer level.

B.5.2 Scalability

The cost of neuron identification in PNKE increases with model scale (total number of neurons). However, as knowledge is typically represented sparsely in large models, the growth rate

of $|N_{\text{causal}}|$ and $|N_{\text{critical}}|$ is expected to be slower than the growth rate of the total model parameters or total neurons. The computational advantage of the editing phase becomes more pronounced as model size increases, because W_{critical} constitutes a smaller fraction of total parameters. In sequential editing scenarios, the full PNKE process is executed for each edit, leading to a total cost that grows linearly with the number of edits. Optimizing the identification process (e.g., by leveraging information from previous edits) is a potential direction for improving efficiency in large-scale sequential editing.

B.6 Detailed Explanation of the Parameter Editing Mechanism

The specific implementation of parameter editing in Section 3.3.3 of the main paper, particularly the relationship between Equation (9) and Equation (10), is clarified as follows.

B.6.1 Equation (9) – General Constrained Optimization Framework

Equation (9) from the main paper, $\min_{\Delta_{\text{FFN}};\Delta_{\text{FFN}}=\Delta_{\text{FFN}}\odot M_{\text{mask}}} \mathcal{L}(\Delta_{\text{FFN}})$, provides the high-level constraint for parameter editing in PNKE. It stipulates that any parameter update Δ_{FFN} must be confined to the critical parameter subspace defined by the binary mask M_{mask} (where elements with a value of 1 correspond to parameter positions in W_{critical}). The loss function $\mathcal{L}(\Delta_{\text{FFN}})$ retains the definitions of the edit loss $\mathcal{L}_{\text{edit}}$ and the preservation loss $\mathcal{L}_{\text{preserve}}$ as defined in Equation (1) of the main paper. This constrained optimization problem is solved using standard gradient-based methods (e.g., Adam optimizer), where gradients are computed and applied only to the parameters indicated by M_{mask}.

B.6.2 Equation (10) – Instantiation of Editing in a Specific Subspace

Equation (10) from the main paper, $\min_{\Delta W_{\text{out,critical}}}[\|(W_{\text{out,critical}} + \Delta W_{\text{out,critical}})h_{1,\text{critical}} - V_1'\|_F^2 + \lambda\|(W_{\text{out,critical}} + \Delta W_{\text{out,critical}})h_{0,\text{critical}} - V_0'\|_F^2]$, is a specific and efficient way to implement the idea of Equation (9), particularly suitable when editing is primarily achieved by modifying $W_{\text{out,critical}}$ (the part of the FFN output layer weights corresponding to N_{critical}) to achieve the target output V_1'. In this equation, $h_{1,\text{critical}}$ and $h_{0,\text{critical}}$ represent the activation vectors of the critical neurons N_{critical} for the edit and preservation samples, respectively (i.e., the output of the FFN's intermediate layer, but only selecting dimensions corresponding to N_{critical}). V_1' and V_0' are the desired (modified) outputs or output changes at the W_{out} layer for these activations. This least-squares problem often has an analytical solution or can be solved efficiently by iterative methods (e.g., a ridge regression solver).

B.6.3 Application Strategy

This study primarily adopts a strategy based on Equation (10) to implement parameter editing. That is, N_{critical} and its corresponding W_{critical} (particularly the $W_{\text{out,critical}}$ part) and activations h_{critical} are first identified through the initial two steps of PNKE. Then, drawing on the principles of methods like ROME and MEMIT, the optimization problem described in Equation (10) is solved for this significantly reduced $W_{\text{out,critical}}$ and h_{critical} to compute the parameter update $\Delta W_{\text{out,critical}}$. This approach combines the precision of PNKE's neuron selection with the maturity and computational efficiency of existing high-performance editing algorithms.

C Comparison of Computational Overhead between PNKE Framework and Baseline Methods

The PNKE (Precise Neuron-Level Knowledge Editing) framework excels in precision and long-term stability for knowledge editing tasks. However, its fine-grained neuron-level operations introduce computational overhead. This section analyzes the sources of this overhead and compares them with baseline methods, with results summarized in a table 3.

C.1 Computational Overhead of PNKE

The computational cost of PNKE arises from three key steps:

- **Causal Neuron Identification (N_{causal}):**
  - *Background Neurons (N_{bg}):* Requires m forward passes per target knowledge point, using m semantically similar prompts (P_{sem}) to compute average neuron activation and stability.
  - *Trigger Neurons (N_{trig}):* Involves one forward pass with a specific prompt (P_{spec}) to obtain activation values, followed by one backward pass using Integrated Gradients (IG), potentially re-

quiring multiple evaluations for the path integral.

- **Critical Neuron Selection (N_{critical}):** Encompasses importance score calculation, normalization, entropy computation, and percentile-based thresholding. These vector and scalar operations incur significantly lower costs than the identification phase.

- **Target Neuron Parameter Editing ($\Delta W_{\text{critical}}$):** Edits only the parameter subset (W_{critical}) tied to critical neurons (N_{critical}). Using subspace methods like ROME or MEMIT reduces complexity compared to editing an entire feed-forward network (FFN) layer.

C.2 Computational Overhead of Baseline Methods

- **Fine-Tuning (FT):** Demands full forward and backward propagation across the model per edit, updating numerous parameters and resulting in substantial overhead.

- **ROME:** Identifies the editing layer via causal tracing (a few forward passes) and applies a closed-form rank-one update, ensuring efficient editing.

- **MEMIT:** Extends ROME with multi-layer low-rank updates, increasing computational cost slightly while remaining efficient.

- **AlphaEdit:** Builds on ROME/MEMIT with null-space projection, adding matrix operations to the overhead.

- **Knowledge Neurons (KN):** Employs attribution techniques to identify neurons (akin to PNKE's N_{trig}), followed by iterative fine-tuning of selected neurons.

- **WISE:** Uses an external memory module and routing decisions for editing, avoiding base model parameter changes and minimizing overhead.

C.3 Summary and Analysis of Experimental Results

The PNKE framework achieves superior performance on the ZsRE dataset, with an editing success rate (Rel.) of 0.936, generalization (Gen.) of 0.891, and locality (Loc.) of 0.952. After extensive editing (T=5000), it sustains high accuracies of 0.510 on MMLU and 0.485 on GSM8K, outperforming other methods significantly.

Regarding computational overhead, PNKE's neuron identification demands 30 evaluations—higher than ROME (5 passes) or MEMIT (10 passes). However, its parameter editing is minimal, targeting only a small subset, resulting in approximately 500 MB of memory usage and 2.5 seconds per edit, reflecting strong efficiency. In contrast, Fine-Tuning, with no identification cost, incurs the highest overhead due to full parameter updates, requiring about 10 GB of memory and 10 seconds per edit. KN and AlphaEdit fall in the mid-range, while WISE, leveraging external memory, exhibits the lowest overhead but slightly weaker performance.

In conclusion, PNKE's high precision, minimal interference, and robust long-term stability make it ideal for continuous knowledge updating. Fine-Tuning and KN struggle with large-scale sequential editing due to performance degradation over time, while ROME and AlphaEdit, effective in the short term, face limitations in long-term stability 3..

D Parameter Settings

Table 3: Performance and Computational Overhead Comparison of PNKE Framework and Baseline Methods

Method	Rel.	Gen.	Loc.	MMLU (T=2000)	MMLU (T=5000)	GSM8K (T=5000)	Neuron Identification Cost	Parameter Editing Cost	Extra Operations	Memory Usage	Editing Time	Long-term Stability
PNKE	0.936	0.891	0.952	0.611	0.510	0.485	30 evaluations	Low (few params)	Entropy calculation	≈ 500 MB	≈ 2.5 s /edit	High
Fine-Tuning	0.850	0.750	0.600	0.100	0.050	0.045	None	High (all params)	None	≈ 10 GB	≈ 10 s /edit	Low
ROME	0.920	0.870	0.910	0.300	0.150	0.140	5 forward passes	Medium (layer update)	None	≈ 1 GB	≈ 1 s /edit	Medium
MEMIT	0.930	0.880	0.920	0.350	0.200	0.190	10 forward passes	Medium (multi-layer)	None	≈ 1.5 GB	≈ 1.5 s /edit	Medium
AlphaEdit	0.940	0.890	0.930	0.250	0.170	0.160	5 forward passes	Medium (update+proj.)	Matrix operations	≈ 1.2 GB	≈ 1.8 s /edit	Medium
KN	0.900	0.850	0.880	0.180	0.120	0.110	20 evaluations	Medium (fine-tuning)	None	≈ 800 MB	≈ 2 s /edit	Low
WISE	0.910	0.860	0.940	0.200	0.150	0.145	None	None (external memory)	Memory access	≈ 300 MB	≈ 0.5 s /edit	Medium

Algorithm 1 Generating Semantically Similar Prompts P_{sem}

Input: Target knowledge K_{target}, number of prompts $m = 10$
Output: Set of prompts $P_{\text{sem}} = \{p_{\text{sem}}^{(1)}, \ldots, p_{\text{sem}}^{(m)}\}$
function GENERATEPSEM(K_{target}, m)
    $templates \leftarrow$ ConstructTemplates(K_{target}) ▷ Manual or dataset-based
    $prompts \leftarrow \emptyset$
    for each t in $templates$ **do**
        $paraphrases \leftarrow$ Paraphrase(t, LLaMA3-8B-Instruct) ▷ Diverse rephrasing
        $prompts \leftarrow prompts \cup paraphrases$
        if $|prompts| < m$ **then**
            $bt_{\text{prompts}} \leftarrow$ BackTranslate(t, source $\rightarrow$ intermediate $\rightarrow$ source)
            $prompts \leftarrow prompts \cup bt_{\text{prompts}}$
        end if
    end for
    return $prompts[1:m]$
end function

Algorithm 2 Identifying Background Neurons N_{bg}

Input: Prompt set P_{sem}, neuron set $\mathcal{N}$, thresholds $\theta_{\text{bg\_act}}$, ϵ_{stable}
Output: Background neurons N_{bg}
function IDENTIFYNBG($\mathcal{N}, P_{\text{sem}}$)
    $N_{\text{bg}} \leftarrow \emptyset$
    for each neuron n_i in $\mathcal{N}$ **do**
        $act_{\text{avg}} \leftarrow \frac{1}{m} \sum_{p \in P_{\text{sem}}} \text{Act}(n_i, p)$
        $act_{\text{std}} \leftarrow \text{std}_{p \in P_{\text{sem}}}(\text{Act}(n_i, p))$
        if $act_{\text{avg}} > \theta_{\text{bg\_act}}$ **and** $act_{\text{std}} < \epsilon_{\text{stable}}$ **then**
            $N_{\text{bg}} \leftarrow N_{\text{bg}} \cup \{n_i\}$
        end if
    end for
    return N_{bg}
end function

Algorithm 3 Identifying Trigger Neurons N_{trig}

Input: Specific prompt P_{spec}, neuron set $\mathcal{N}$, thresholds $\theta_{\text{trig\_act}}$, θ_{attr}
Output: Trigger neurons N_{trig}
function IDENTIFYNTRIG($\mathcal{N}, P_{\text{spec}}$)
    $N_{\text{trig}} \leftarrow \emptyset$
    for each neuron n_i in $\mathcal{N}$ **do**
        $act \leftarrow \text{Act}(n_i, P_{\text{spec}})$
        $attr \leftarrow$ IntegratedGradients(n_i, P_{spec}, baseline $= 0$)
        if $act > \theta_{\text{trig\_act}}$ **and** $attr > \theta_{\text{attr}}$ **then**
            $N_{\text{trig}} \leftarrow N_{\text{trig}} \cup \{n_i\}$
        end if
    end for
    return N_{trig}
end function

Algorithm 4 Critical Neuron Selection with Entropy Mechanism

Input: Causal neurons N_{causal}, prompt P_{spec}, $\alpha = 0.5$, $q_{\text{base}} = 85$, $\gamma = 10.0$
Output: Critical neurons N_{critical}
function SELECTNCRITICAL($N_{\text{causal}}, P_{\text{spec}}$)
    $scores \leftarrow \emptyset$
    for each n_i in N_{causal} **do**
        $act_{\text{norm}} \leftarrow \text{MinMaxNorm}(\text{Act}(n_i, P_{\text{spec}}))$
        $attr_{\text{norm}} \leftarrow \text{MinMaxNorm}(\text{Attr}(n_i, P_{\text{spec}}))$
        $s_i \leftarrow \alpha \cdot act_{\text{norm}} + (1 - \alpha) \cdot attr_{\text{norm}}$
        $scores \leftarrow scores \cup \{s_i\}$
    end for
    $H(P_S) \leftarrow \text{Entropy}(scores)$
    $q \leftarrow q_{\text{base}} + \gamma \cdot (\log |N_{\text{causal}}| - H(P_S))$
    $\tau_H \leftarrow \text{Percentile}(scores, q)$
    $N_{\text{critical}} \leftarrow \{n_i \in N_{\text{causal}} \mid s_i > \tau_H\}$
    return N_{critical}
end function