Knowledge Editing for Large Language Model with Knowledge Neuronal Ensemble

Anonymous ACL submission

Abstract

As real-world knowledge is constantly evolving, ensuring the timeliness and accuracy of a model's knowledge is crucial. This has made knowledge editing in large language models increasingly important. However, existing knowledge editing methods face several challenges, including parameter localization coupling, imprecise localization, and a lack of dynamic interaction across layers. In this paper, we propose a novel knowledge editing method called **Knowledge Neuronal Ensemble (KNE)**. A knowledge neuronal ensemble represents a group of neurons encoding specific knowledge, thus mitigating the issue of frequent parameter modification caused by coupling in parameter localization. The KNE method enhances the precision and accuracy of parameter localization by computing gradient attribution scores for each parameter at each layer. During the editing process, only the gradients and losses associated with the knowledge neuronal ensemble are computed, with error backpropagation performed accordingly, ensuring dynamic interaction and collaborative updates among parameters. Experimental results on three widely used knowledge editing datasets show that the KNE method significantly improves the accuracy of knowledge editing and achieves, or even exceeds, the performance of the best baseline methods in portability and locality metrics.

1 Introduction

Real-world knowledge is constantly evolving, and the purpose of knowledge editing(Wang et al., 2024a) in large language models is to modify outdated or incorrect knowledge with new, accurate knowledge while minimizing negative effects on previously learned knowledge and capabilities. Current update methods for large language models include fine-tuning(Han et al., 2024) and retrieval augmentation(Gao et al., 2024).Fine-tuning requires high computational resources, is prone to over-fitting, may negatively impact other knowledge, and often leads to catastrophic forgetting. Retrieval augmentation, on the other hand, struggles with retrieval noise, making precise editing difficult, and provides only short-term, temporary changes, limiting its efficiency for large-scale updates. Knowledge editing(Yao et al., 2023) seeks to empower large language models to learn continuously and maintain accurate knowledge, much like humans who read books and newspapers daily.

Existing knowledge editing methods generally involve two main steps(Zhang et al., 2024): parameter localization and parameter editing. Parameter localization serves as the foundation for understanding the internal working mechanisms of the model and for performing effective parameter editing. However, current knowledge editing methods face several challenges in both localization and editing: (1) **Knowledge Localization Coupling**:Knowledge localization is often coupled, meaning a single neuron may correspond to multiple pieces of knowledge, leading to frequent parameter adjustments that may destabilize the model or degrade the quality of specific knowledge edits, ultimately compromising overall performance. (2) **Inaccurate Knowledge Localization**:Current localization techniques may be inaccurate, diminishing the specificity and efficiency of the editing process. For instance, causal tracking methods may reveal significant associations between certain layers and specific knowledge, even when those layers are not directly edited. (3) **Insufficient Layer-wise Dynamic Interaction for Parameter Update**:Furthermore, when editing parameters from shallow to deep layers, dynamic interaction and collaborative updates between layers are often lacking, which can negatively impact the final model. Therefore, optimizing the knowledge editing framework is crucial for improving its effectiveness.

Inspired by knowledge neuron(Dai et al., 2022) method, we introduce a novel knowledge editing method:**K**nowledge **N**euronal **E**nsemble (**KNE**). A knowledge neuronal ensemble is a collection of neurons that represents a set of related knowledge, addressing the problem of frequent parameter modification caused by knowledge localization coupling. In this method, we calculate gradient attribution scores for each parameter in each layer to identify all parameters that significantly contribute to representing specific knowledge, allowing for more accurate and refined parameter localization. During the editing process, gradients and losses over the knowledge neuronal ensemble are computed, and error backpropagation is applied, ensuring dynamic interaction and collaborative updates among the edited parameters. Experiments conducted on three widely used knowledge editing datasets demonstrate that KNE achieves superior performance. Compared to five baseline methods, KNE significantly improves the accuracy of knowledge editing and matches or exceeds the best baseline methods in portability and locality metrics, with some datasets showing even better results.

Moreover, experiments indicate that editing the knowledge neuronal ensemble corresponding to the key layers of the feed-forward neural network (FFN) yields results comparable to those from previous methods that edited value layers, with even better locality performance. This finding enriches existing assumptions about knowledge storage (Geva et al., 2021, 2022) locations and suggests that different parameter locations can be edited based on the specific editing goal to achieve optimal results.

The contributions of this paper are as follows:

- **Knowledge Neuronal Ensemble Method**: The concept of the "knowledge neuronal ensemble" is introduced, solving the problem of frequent parameter modification due to knowledge localization coupling.

- **More Precise and Accurate Knowledge Localization**: Gradient attribution scores are used to identify the parameters that have a significant impact on expressing specific knowledge, ensuring more accurate localization.

- **Layer-wise Dynamic Interaction for Parameter Update**: Gradients and losses over the knowledge neuronal ensemble are used for error backpropagation, facilitating dynamic information transfer across layers for collaborative parameter updates.

- **Low Computational Cost and Efficient Localization**: By optimizing the update strategy, the number of parameters that need to be modified is reduced to around 1% of the model's total parameters, significantly reducing computational cost and improving localization efficiency.

2 Related Work

Existing knowledge editing methods can be broadly categorized into two types(Yao et al., 2023): those that modify model parameters and those that do not.

Among the non-parameter-modifying methods, there are two paradigm. The first is knowledge editing based on retrieval augmentation, which treats the new knowledge as external knowledge in the retrieval-augmented model,i.e.SERAC(Mitchell et al., 2022b), IKE(Zheng et al., 2023), Wang et al.(Wang et al., 2024b), Shi et al.(Shi et al., 2024), MemPrompt(Madaan et al., 2022), Murty et al.(Murty et al., 2022).

The second involves adding extra trainable parameters, which are trained on the modified knowledge dataset while the original model parameters remain unchanged,i.e.T-Patcher(Huang et al., 2023), CaliNet(Dong et al., 2022), GRACE(Hartvigsen et al., 2023), MELO(Yu et al., 2024). Both of them leave the original model parameters unchanged, leading to the model's inability to deeply understand or fully integrate the new knowledge.

Although parameter-modifying methods are more challenging, they facilitate a deeper understanding of knowledge storage within the model's internal mechanisms. This enables the model to better grasp the inherent nature of knowledge and apply it flexibly, a focus of our research. Parameter-modifying methods can also be classified into two types: meta-learning-based methods and locate-then-edit methods.

Meta-learning methods use hyper networks for learning parameter updates for large language models (LLMs). The Knowledge Editor (KE)(Cao et al., 2021)utilizes a hyper network (specifically a bidirectional LSTM) to predict the weight updates for each data point, thereby achieving constrained optimization in editing target knowledge without disrupting other knowledge. However, this approach faces limitations when editing LLMs. To overcome this, Model Editing Networks with Gradient Decomposition (MEND)(Mitchell et al., 2022a)learns to transform the fine-tuning gradients of language models through low-rank decomposition, achieving better performance on LLMs. MALMEN(Tan et al., 2024) formulates parameter shift aggregation as a least squares problem and updates model parameters using the normal equation, allowing for scalable editing of multiple facts with limited memory.

The locate-then-edit approach first identifies the parameters corresponding to specific knowledge and modifies them by directly updating the target parameters. The Knowledge Neuron (KN) method(Dai et al., 2022) introduces a knowledge attribution technique to locate "knowledge neurons" (key-value pairs in the FFN matrix) that embody the knowledge, and subsequently updates these neurons. ROME(Meng et al., 2022)employs causal mediation analysis to locate the editing region. Unlike modifying knowledge neurons in the FFN, ROME adjusts the entire matrix, viewing model editing as a least-squares problem with linear equality constraints, solving it using Lagrange multipliers. However, both KN and ROME can only edit one factual association at a time. To address this, MEMIT(Meng et al., 2023)extends ROME, enabling simultaneous editing of multiple cases. Building on MEMIT, PMET(Li et al., 2024)introduces attention values for better performance.

3 Method

3.1 Preliminaries

As in Wang et al.(Wang et al., 2024a), the editing is performed on a relational fact that can be represented as a knowledge triple (s, r, o) where s, r, and o are the subject, relation, and object of the fact respectively. A single knowledge editing task denoted by e is to modify the weights of the model such that the original knowledge triple encoded in the model is changed to (s, r, o^*). Since the context of this work is pre-trained LLMs, we use $x_e = (s, r)$ to represent the prompt composed of s and r, and substitute y_e for o to represent the answer. Thus, a LLM with parameter ϕ can be regarded as a mapping represented by $f : x \to y$, and a knowledge editing task e will yield $f^* : x \to y^*$ with the parameter updated to ϕ^*. It is more common to perform multiple knowledge editing on a set of edits $\mathcal{E} = \{e_1, e_2, \ldots\}$. Let $\mathcal{X}_\mathcal{E} = \cup_{e\in\mathcal{E}} x_e$ and $\mathcal{Y}_\mathcal{E} = \cup_{e\in\mathcal{E}} y_e$. We formalize the problem of **Knowledge Editing (KE)** following the definition in Wang et al.(Wang et al., 2024a).

Definition 1 The objective of knowledge editing is to perform the following constrained optimization:

$$\begin{aligned} &\min_{\phi^*} \mathbb{E}_{e\in\mathcal{E}} \mathbb{E}_{x\in\mathcal{X}_e, y^*\in\mathcal{Y}_e^*} \mathcal{L}(f^*(x), y^*), \\ &\text{s.t.} \quad f^*(x) = f(x), \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_E. \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the loss function which quantifies the deviation between the LLM output $f^*(x)$ and y^* from the expected answer set $\mathcal{Y}_e^*$.

For methods of Locate-Then-Edit paradigm, locating neurons attributed to the specific knowledge triple, i.e., Knowledge Attribution, is a prerequisite. We refine the problem statement of knowledge editing in this case.

Definition 2 The objective of Locate-Then-Edit knowledge editing is to perform the following constrained optimization:

$$\begin{aligned} &\min_{\phi_k^*} \mathbb{E}_{e\in\mathcal{E}} \mathbb{E}_{x\in\mathcal{X}_e, y^*\in\mathcal{Y}_e^*} \mathcal{L}\left(f^*_{\bar{\phi}_k, \phi_k^*}(x), y^*\right), \\ &\text{s.t. } f^*_{\bar{\phi}_k, \phi_k^*}(x) = f(x), \quad \forall x \in \mathcal{X} \setminus \mathcal{X}_E, \\ &\text{where } \phi_k = L(f_\phi, \mathcal{E}), \quad \bar{\phi}_k = \phi \setminus \phi_k. \end{aligned} \tag{2}$$

Here, $\phi_k = L(f_\phi, \mathcal{E})$ uses function L to locate neurons attributed to edits $\mathcal{E}$. Then, $\bar{\phi}_k$ represents the unedited weights.

Definition 3 (Knowledge Neurons, KNs). Given a set of edits $\mathcal{E} = \{e_1, \ldots\}$ where $|\mathcal{E}| \geq 1$, for a L layer neural network, suppose there are m neurons in layer l. If $\mathcal{E}$ can be attributed to k neurons $w_{j_1}^{(l)}, \ldots, w_{j_k}^{(l)}$ in layer l, such that the activation of these neurons significantly contributes to $\mathcal{E}$, then these neurons $\mathbf{N}_\mathcal{E}^{(l)} = \{w_{j_1}^{(l)}, \ldots, w_{j_k}^{(l)}\}$ are referred to as the knowledge neurons for $\mathcal{E}$ in layer l.

Definition 4 (Knowledge Neuronal Ensemble, KNE). The Knowledge Neuronal Ensemble of the L layers neural network for $\mathcal{E}$ is defined to be the set of knowledge neurons in all layers, i.e., $\mathbf{N}_\mathcal{E} = \{\mathbf{N}_\mathcal{E}^{(1)}, \ldots, \mathbf{N}_\mathcal{E}^{(l)}\}$.

3.2 Knowledge Neuronal Ensemble Method

To localize the Knowledge Neuronal Ensemble corresponding to a set of knowledge, we use a token-level gradient attribution method(Dai et al., 2022) to calculate gradient attribution scores for the relevant parameters. Based on these scores, we select and construct the Knowledge Neuronal Ensemble. Afterward, we freeze the parameters in other locations and dynamically update only the parameters in the Knowledge Neuronal Ensemble to

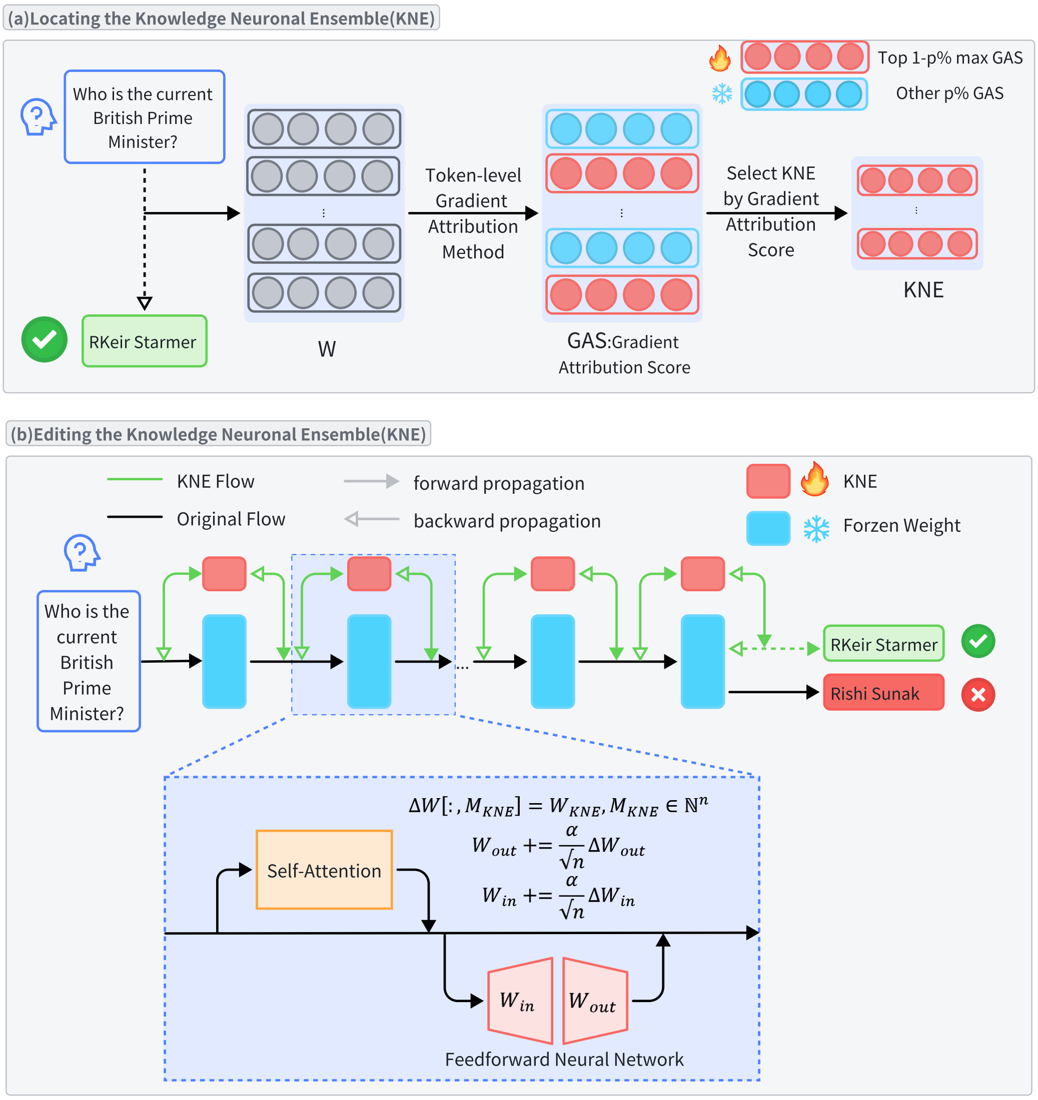


Figure 1: The framework of KNE method.

achieve coordinated optimization.The framework of the Knowledge Neuronal Ensemble is shown in Figure1.

3.2.1 Token-level Gradient Attribution Method

For a given input query x, and the correct answer $\mathrm{y}_\mathrm{j}^* = \{y_1, y_2, \ldots, y_j\}$, where y_j^* represents the correct answer composed of j tokens and y_j represents the j-th token, we define the model's output probability as:

$$\mathrm{P}_{\mathrm{y}_\mathrm{j}^*|x}(\hat{w}_k^{(l)}) = p(\mathrm{y}_\mathrm{j}^*|x, w_k^{(l)} = \hat{w}_k^{(l)}) \tag{3}$$

Here, x is the input query, y_j^* is the correct answer, $w_k^{(l)}$ refers to the k-th neuron in the l-th layer, and $\hat{w}_k^{(l)}$ represents the parameter value at location $w_k^{(l)}$.

To extend the gradient attribution method to large language models built on GPT-like architectures, we accumulate the gradient attribution scores computed for each token in the correct answer:

$$\mathrm{Attr}(w_k^{(l)}) = \overline{w}_k^{(l)} \sum_{j=1}^{s} \int_{\alpha=0}^{1} \frac{\partial \mathrm{P}_{\mathrm{y}_\mathrm{j}^*|x}(\alpha \overline{w}_k^{(l)})}{\partial w_k^{(l)}} d\alpha \tag{4}$$

Where s is the number of tokens in the correct answer, and $\overline{w}_k^{(l)}$ is the original parameter value at location $w_k^{(l)}$.

Since calculating the continuous gradient integral is computationally expensive, we apply a Riemann approximation as follows:

$$\widetilde{\mathrm{Attr}}(w_k^{(l)}) = \frac{\overline{w}_k^{(l)}}{m} \sum_{j=1}^{s} \sum_{i=1}^{m} \frac{\partial \mathrm{P}_{\mathrm{y}_j^*|x}\left(\frac{i}{m}\overline{w}_k^{(l)}\right)}{\partial w_k^{(l)}} \quad (5)$$

Where m represents the number of discrete steps used to approximate the continuous integral, reducing computational complexity.

3.2.2 Selection of the Knowledge Neuronal Ensemble

We define $\mathrm{Attr}(w_k^{(l)})$ as the gradient attribution score of the k-th neuron in the l-th layer, and let d_2 represent the output dimension of the l-th layer. To construct the Knowledge Neuronal Ensemble (KNE), we select the neurons with the top $1-p\%$ largest gradient attribution scores. The threshold t_p is defined as follows:

$$\begin{aligned} t_p = \mathrm{Quantile}_{1-p}(\{\mathrm{Attr}(w_k^{(l)}) \mid \\ k = 1, 2, \ldots, d_2;\ l = 1, 2, \ldots, L\}), \end{aligned} \quad (6)$$

where $\mathrm{Quantile}_{1-p}$ refers to the quantile function that calculates the value corresponding to the top $p\%$ of the gradient attribution scores.

Next, we define the Knowledge Neuronal Ensemble (KNE) as the set of neuron indices that meet the following condition:

$$\begin{aligned} \mathbf{N}_{\mathcal{E}}^* = \{\{k^{(l)}\} \mid \mathrm{Attr}(w_k^{(l)}) \geq t_p, \\ k = 1, 2, \ldots, d_2; l = 1, 2, \ldots, L\}. \end{aligned} \quad (7)$$

In this way, the KNE includes all neuron indices with gradient attribution scores greater than or equal to the threshold t_p, representing the top $1-p\%$ of neurons ranked by gradient attribution scores.

3.2.3 Editing the Knowledge Neuronal Ensemble

To fully utilize the localized information from the Knowledge Neuronal Ensemble (KNE), we propose a Knowledge Neuronal Ensemble Editing method. Throughout the editing phase, gradients and losses are calculated across the knowledge neuronal ensemble. Subsequently, error backpropagation is employed to facilitate dynamic interaction and coordinated updates among the parameters being refined. Based on the number of neurons n in the Knowledge Neuronal Ensemble for each layer, we dynamically allocate a Knowledge Neuronal Ensemble parameter matrix:

$$W_{kne} \in \mathbb{R}^{n \times d_1} \quad (8)$$

Here, n represents the number of neurons in the Knowledge Neuronal Ensemble for that layer, while d_1 and d_2 represent the input and output dimensions of the weight matrix W.

Since W_{kne} and W have different dimensions, we initialize a zero matrix ΔW with the same dimensions as W, to map the updated W_{kne} to the corresponding positions of the Knowledge Neuronal Ensemble. The formulation is as follows:

$$\Delta W \in \mathbb{R}^{d_2 \times d_1}, W \in \mathbb{R}^{d_2 \times d_1} \quad (9)$$

$$\Delta W[:, M_{kne}] = W_{kne}, M_{kne} \in \mathbb{N}^n \quad (10)$$

Where M_{kne} represents the index positions of the Knowledge Neuronal Ensemble within the weight matrix W. The values of M_{kne} are natural numbers less than d_2, and the length of M_{kne} corresponds to the number of neurons n in the Knowledge Neuronal Ensemble.

Finally, the weight matrix W is updated to obtain $\hat{W}$ using the following formula:

$$\hat{W} = W + \frac{\alpha}{\sqrt{n}} \Delta W \quad (11)$$

To control the extent of the parameter updates, we multiply ΔW by a scaling factor $\frac{\alpha}{\sqrt{n}}$, where α is a hyper parameter.

4 Experiments

4.1 Experimental Setting

In our study, we chose the **Llama2-7B-chat** and **gpt-j-6B** models as the cornerstone for knowledge editing tasks. These models were selected for their widespread adoption and proven efficacy. To ensure a comprehensive evaluation, we employed diverse datasets that encapsulate a broad spectrum of knowledge domains, namely the **ZsRE**, **WikiDatacounterfact**, and **WikiDatarecent** datasets. Our assessment criteria encompassed several key metrics: **Edit Success**, which measures the accuracy of edits; **Portability**, indicating how well edits transfer across different questions; **Locality**, assessing the precision of edit localization; and **Fluency**, evaluating the naturalness of the edited text. We benchmarked our approach against established baseline methods, comprising Fine-Tuning (**FT**), Fine-Tuning with Linear probing (**FT-L**), **AdaLoRA**, **ROME**, and **MEMIT**. For an in-depth exploration of these comparisons and additional methodological details, please refer to Appendix A.

4.2 Experimental Results

We selected the widely used **Llama2-7B-chat** model as the foundation for knowledge editing. To validate the generalizability of our approach, we utilized datasets that represent various forms of knowledge, including **ZsRE**,**WikiDatarecent** and **WikiDatacounterfact**. the experimental results summarized in Table 1.Experimental findings across three extensively utilized knowledge editing datasets demonstrate that the KNE methodology markedly enhances the precision of knowledge editing, while also attaining—or even surpassing—the levels of performance exhibited by the top baseline methods in terms of portability and locality metrics.

Furthermore, to evaluate the effectiveness of our method across different models, we compared it with the **gpt-j-6B** model. The **Llama2-7B-chat** model demonstrated exceptional performance in handling complex knowledge editing tasks due to its larger parameter size and enhanced generative capabilities. In contrast, the **gpt-j-6B** model is favored for its lower computational resource requirements and faster response times. By comparing these two models, we gained deeper insights into their respective strengths and limitations in knowledge editing tasks.Experiments show that the KNE method is also applicable to the **gpt-j-6B** model.The results of these comparative experiments are summarized in Table 2.

We evaluate several knowledge editing methods on a range of language models and datasets, assessing their performance across key metrics.

4.3 Exploring the Storage Location of Knowledge in Large Language Models

Exploring the storage location of knowledge in large language models has traditionally relied on the assumption that factual knowledge is stored in the **key-value memory** format within the fully connected layers of the **FFN** module, as seen in methods such as ROME, MEMIT and the knowledge neuron approach.

However, we have identified several interesting deviations from this assumption.To analyze the precise locations of knowledge within these models, we employed a gradient attribution method to calculate gradient attribution scores for each parameter layer in both the Self-Attention and FFN modules. Using these scores, we identified specific layers and evaluated their post-editing performance, as illustrated in Figure. 2. This analysis yielded several notable conclusions:

- **Edit Success and Portability:** Editing within the FFN module consistently led to superior Edit Success and Portability compared to the Self-Attention module. Notably, the value layer (`mlp.down_proj`) in the FFN module exhibited the best overall performance. However, high editing accuracy was observed across various layers and modules, indicating that effective edits are achieved throughout the model.

- **Locality and Fluency:** For Locality and Fluency, editing mapping layers—such as `mlp.gate_proj` and `mlp.up_proj` in the FFN module, along with `self_attn.q_proj` in the Self-Attention module—demonstrated significantly better performance than other layers.

These findings suggest that while knowledge is indeed stored in the FFN module, the specific layers edited impact different performance aspects. Moreover, mapping layers play a crucial role in maintaining Locality and Fluency, indicating that the storage and structure of knowledge are more complex and distributed than initially assumed.

4.4 Similar Knowledge May Be Stored in Similar Locations within the Model

Effective model editing does not require localizing every knowledge element in the dataset. By localizing only **200 knowledge items**—around **1/4** of the total dataset—the model achieved high performance. Remarkably, this partial localization strategy outperformed full-dataset localization in both **Locality** and overall performance. Furthermore, metrics such as **Edit Success**, **Fluency**, and **Portability** showed minimal differences between using the entire dataset and using just 1/4 for localization.

This finding significantly improves the feasibility of knowledge editing for practical applications by reducing the computational demands of the localization process. In the **KNE** approach, calculating gradient attribution scores for each parameter across all layers for every knowledge item is highly resource-intensive, often making localization more time-consuming than the editing itself. By streamlining this process and achieving a 75% reduction in localization effort, the method accelerates over-

Table 1: Performance Comparison of Knowledge Editing Methods Across Different Datasets

Dataset	Metric	FT	FT-L	AdaLoRA	ROME	MEMIT	KNE
WikiData counterfact	Edit Succ.	26.78	51.12	72.14	83.21	83.41	**99.02**
	Portability	16.94	39.07	**55.17**	38.69	40.09	53.88
	Locality	0.29	62.51	**66.78**	65.4	63.68	65.09
	Fluency	483.71	544.80	553.85	578.84	568.58	**591.25**
ZsRE	Edit Succ.	36.88	54.65	69.86	96.57	83.07	**97.75**
	Portability	8.72	45.02	52.95	52.20	51.43	**58.02**
	Locality	0.31	71.12	72.21	27.14	25.46	**76.85**
	Fluency	471.29	474.18	532.82	570.47	559.72	**571.93**
WikiData recent	Edit Succ.	31.24	71.18	65.61	85.08	85.32	**99.48**
	Portability	15.91	48.71	47.22	37.45	37.94	**63.36**
	Locality	3.65	63.7	55.78	**66.2**	64.78	37.58
	Fluency	428.67	549.35	537.51	574.28	566.66	**581.49**

Table 2: Performance Comparison of Knowledge Editing Methods Across Different Models

Dataset	Model	Edit Succ.	Portability	Locality	Fluency
WikiData counterfact	Llama2-7b-chat	99.02	53.88	65.09	591.25
	gpt-j-6b	99.35	49.14	52.64	597.29
ZsRE	Llama2-7b-chat	97.75	58.02	76.85	571.93
	gpt-j-6b	99.90	53.79	78.60	549.87
WikiData recent	Llama2-7b-chat	99.48	63.36	37.58	581.49
	gpt-j-6b	99.79	57.74	53.47	585.79

all workflow and enhances scalability for industrial applications.

The experiment result of full dataset is shown in Figure. 3 in Appendix.

The dataset, **WikiDatacounterfact**, derived from WikiData, contains numerous data points with inherent similarities, such as comparable classification topics Gueta et al.(Gueta et al., 2023). These similarities imply that related knowledge items are often stored in close model regions. Thus, localizing a representative subset effectively supports the editing process. Further experimentation is required to elucidate the underlying reasons for this behavior.

4.5 Optimal Parameter Selection for Knowledge Editing

To investigate the impact of parameter quantity on knowledge editing performance, a controlled experiment was conducted. Detail experiment result is shown in Figure. 4 in Appendix.

Using more parameters significantly enhances the "Edit Success" and "Portability" metrics, indicating that precise modifications lead to better overall performance and transferability across tasks. Conversely, employing fewer parameters improves the "Locality" metric, suggesting that it helps retain the relevance of edited knowledge to its context, resulting in more focused and localized edits. The choice of parameter quantity thus influences different aspects of model editing, necessitating a balance based on specific requirements.

The results suggest a trade-off between these two sets of metrics. To achieve optimal overall performance, we must balance the number of parameters used for editing. Thus, selecting an appropriate number of parameters is crucial to achieving the best overall editing performance.

4.6 Exploring the Capability of Batch Editing

Most current knowledge editing methods only handle one or a few pieces of knowledge at a time, limiting the efficiency and applicability of knowledge editing. To evaluate whether our proposed method extend to **batch editing**, we conducted a controlled experiment, varying only the number of knowledge pieces edited (i.e., the **batch size**). The performance of our method under different batch

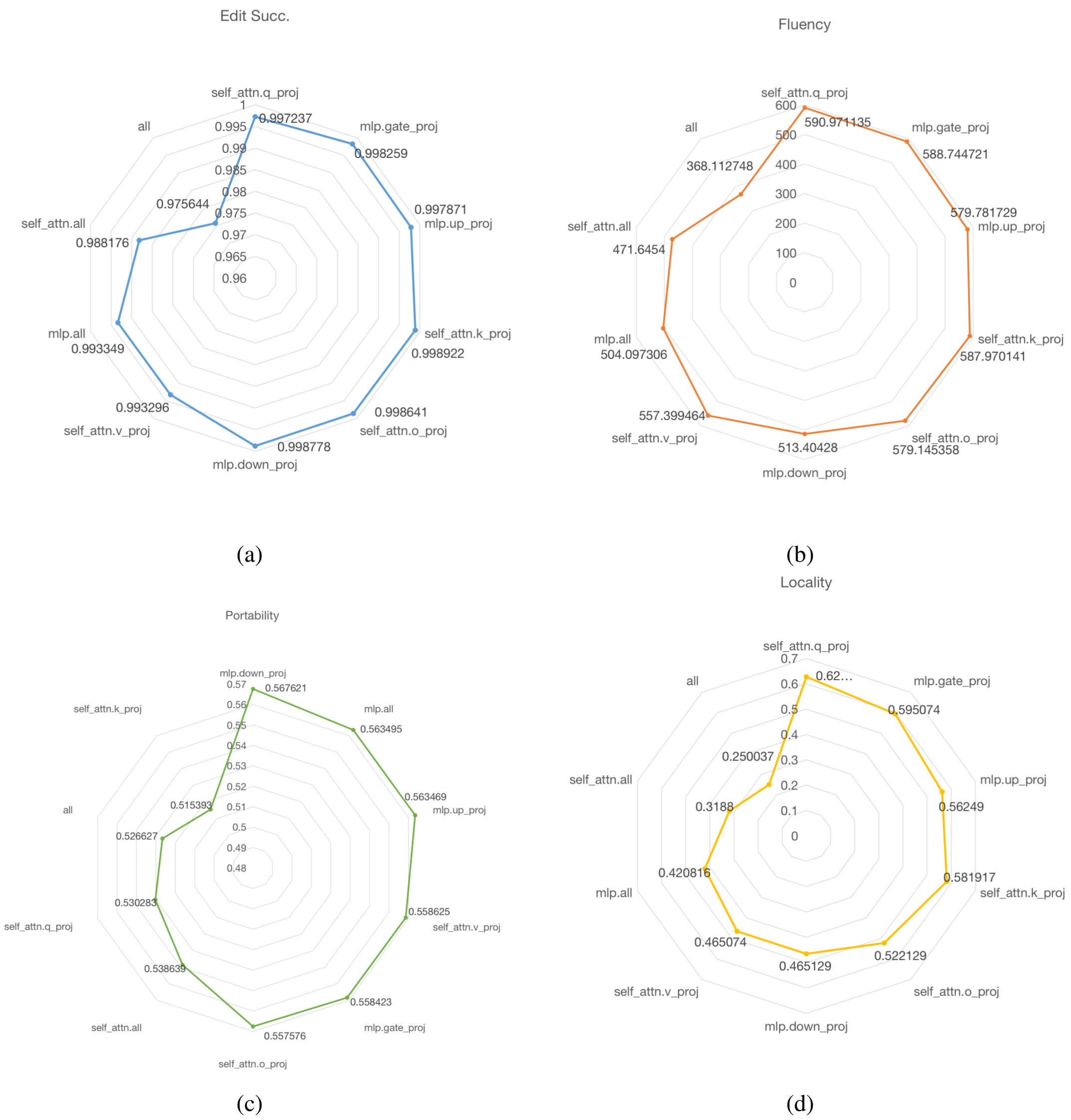


Figure 2: Visualization of Knowledge Storage Deviations in Large Language Models

sizes is as shown in Figure. 5 in Appendix.

Our method demonstrates the ability to perform batch knowledge editing effectively, with only minor trade-offs as batch size increases. While **Edit Success** and **Portability** metrics experience some decline with larger batch sizes, the performance remains acceptable. In contrast, **Locality** and **Fluency** improve with larger batch sizes, at least initially, showing that our method is well-suited for batch editing tasks.

5 Conclusion

This paper introduces a novel knowledge editing framework—the **Knowledge Neurona'l Ensemble (KNE) localization method**—to address the limitations of current knowledge editing techniques, including localization accuracy, editing efficiency, and inter-layer coordinated updates. By introducing the concept of the **Knowledge Neuronal Ensemble**, we not only expand our understanding of knowledge storage locations but also achieve more precise knowledge localization and batch updates by aggregating multiple related knowledge neurons. This method enhances inter-layer interaction through a dynamic gradient propagation mechanism from shallow to deep layers, improving the coherence and accuracy of edits while minimizing negative impacts on overall model performance.Experimental results demonstrate that the KNE localization method outperforms mainstream knowledge editing techniques across multiple datasets, delivering higher accuracy and stability in knowledge editing. It also significantly reduces computational overhead and improves localization efficiency.

Limitations

Although the proposed **Knowledge Neural Ensemble (KNE)** significantly improves knowledge editing performances, several limitations remain that warrant further exploration and optimization.

- **Generality and Scalability**: While the experimental results in this paper demonstrate that the KNE method performs well in terms of accuracy and stability across multiple datasets and task scenarios, its generality and scalability have yet to be fully validated across a broader range of model architectures and more diverse tasks. Different types of large language models, particularly those with specialized structures, may have different patterns of knowledge storage and transmission. Therefore, further research is needed to explore how this method performs in these models.

- **Theoretical Explanation of Key Layer Editing**: While this paper shows that modifying the **key layer** in the FFN module yields favorable results in terms of locality, this finding still requires deeper theoretical analysis and explanation. Future studies should further investigate the knowledge transmission mechanisms between different layers of models to systematically understand and optimize knowledge storage and editing processes, thereby improving the theoretical robustness of the method.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6491–6506. Association for Computational Linguistics.

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.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5937–5947. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 30–45. 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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5484–5495. Association for Computational Linguistics.

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. 2023. Knowledge is a region in weight space for fine-tuned language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 1350–1370. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Preprint*, arXiv:2403.14608.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. PMET: precise model editing in a transformer. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18564–18572. AAAI Press.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve GPT-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2833–2861. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, pages 17359 – 17372.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

meta llama. 2023. Inference code for Llama models. https://github.com/meta-llama/llama.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 15817–15831. PMLR.

Shikhar Murty, Christopher D. Manning, Scott M. Lundberg, and Marco Túlio Ribeiro. 2022. Fixing model bugs with natural language patches. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11600–11613. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pages 2056–2066. ACM.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024a. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3).

Weixuan Wang, Barry Haddow, and Alexandra Birch. 2024b. Retrieval-augmented multilingual knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 335–354. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. MELO: enhancing model editing with neuron-indexed dynamic lora. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Ar-*

tificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pages 19449–19457. AAAI Press.

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, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024. A comprehensive study of knowledge editing for large language models. *Preprint*, arXiv:2401.01286.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4862–4876. Association for Computational Linguistics.

A Detailed Experimental Settings

A.1 Baselines

We compare against the following baseline methods:

- **Fine-Tuning (FT)**: We fine-tune the 'mlp.proj' weights within layer 21, following the re-implementation by Meng et al.(Meng et al., 2023). We use the Adam optimizer with early stopping to minimize negative log probability. Default hyper parameters are used, and unconstrained fine-tuning is consistently applied across all experiments.

- **Fine-Tuning with Linear probing (FT-L)**: This method fine-tunes a pre-trained model by adding and training a linear layer on top, while keeping the original model weights frozen. This linear layer adapts the model to new tasks without altering the pre-trained representations. We utilize the Adam optimizer with early stopping and default hyper parameters for training.

- **AdaLoRA**(Zhang et al., 2023): AdaLoRA efficiently tunes large pre-trained models by applying small, rank-1 updates to the model weights. This is particularly beneficial for large models as it significantly reduces the number of updates compared to full fine-tuning. We employ AdaLoRA on specific layers, using a variant of the Adam optimizer and default hyper parameters.

- **ROME**(Meng et al., 2022): This method treats the MLP module as a key-value store and adds new knowledge via rank-one modification of MLP weights. We utilize the original code and weights (`https://github.com/EleutherAI/ROME`) and retain default hyper parameters.

- **MEMIT**(Meng et al., 2023): An extension of ROME, MEMIT incorporates multiple memories by modifying MLP weights across several layers. We use the publicly available code (`https://github.com/facebookresearch/memit`) with default hyper parameters. For GPT-J, R values range from 3 to 8, and covariance statistics are derived from 100,000 Wikitext samples.

A.2 Models

We evaluate the following language models:

- **Llama-2-7b-chat**(Touvron et al., 2023; meta llama, 2023): Meta's 7-billion parameter chat-tuned model, exhibiting strong performance in dialogue benchmarks. It utilizes an optimized transformer architecture trained with supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF).

- **GPT-J-6B**(Wang and Komatsuzaki, 2021; Wang, 2021): A 6-billion parameter model with 28 layers, a model dimension of 4096, a feedforward dimension of 16384, and 16 heads (dimension 256). It uses RoPE on 64 dimensions per head and a 50257-token vocabulary with BPE encoding.

A.3 Datasets

We utilize the following datasets(Wang et al., 2023)(`https://huggingface.co/datasets/zjunlp/KnowEdit`):

- **ZsRE**: A Question Answering (QA) dataset using back-translation paraphrases to create question equivalence sets. We use the extended version (`https://github.com/yao8839836/zsre`) and construct new locality sets following their methodology. Training data is sourced from the MEND project (https://github.com/eric-mitchell/mend).

- **WikiDatacounterfact**: This dataset focuses on triplets involving prominent Wikidata entities to mitigate issues with tail entities. Training data consists of randomly sampled Wikidata triplets, and the dataset itself serves as the test set.
- **WikiDatarecent**: A dataset of triplets recently added to Wikidata after July 2022, used to evaluate the insertion of new facts into models trained on older data.

A.4 Metrics

Knowledge editing affects predictions for inputs semantically or contextually related to the edited example. This sphere of influence is the editing scope. A successful edit modifies the model within the intended scope without affecting unrelated inputs:

$$f_{\theta_e}(x) = \begin{cases} y_e & \text{if } x \in I(x_e, y_e) \\ f_\theta(x) & \text{if } x \in O(x_e, y_e) \end{cases} \tag{12}$$

where:

- $f_{\theta_e}(x)$: Edited model's prediction on input x.
- y_e: Target output for edited example x_e.
- $I(x_e, y_e)$: Intended scope (inputs related to the edit).
- $O(x_e, y_e)$: Out-of-scope inputs (unrelated to the edit).

We evaluate edits using the following metrics:

- **Edit Success (ES)**: Measures the model's accuracy on the edited fact and similar inputs (paraphrases) . For factual datasets, we use:

$$ES = \sum_{(x_k, y_k^*)} \nVdash \{\text{argmax}_y f_{\theta'}(y|x_k) = y_k^*\} \tag{13}$$

where x_k is the updated knowledge, y_k^* is the target output, and $f_{\theta'}$ is the edited model.

- **Portability (PORT)**: Assesses the edit's impact on related knowledge, including alias/synonym substitution, compositionality/reasoning, and logical generalization.
- **Locality (LOC)**: Measures unintended changes to unrelated knowledge, considering both in-distribution and out-of-distribution locality:

$$\text{LOC} = \mathbb{E}_{x_k, y_k^* \sim O(x_k)} \nVdash \{f_{\theta'}(y|x_k) = f_\theta(y|x_k)\} \tag{14}$$

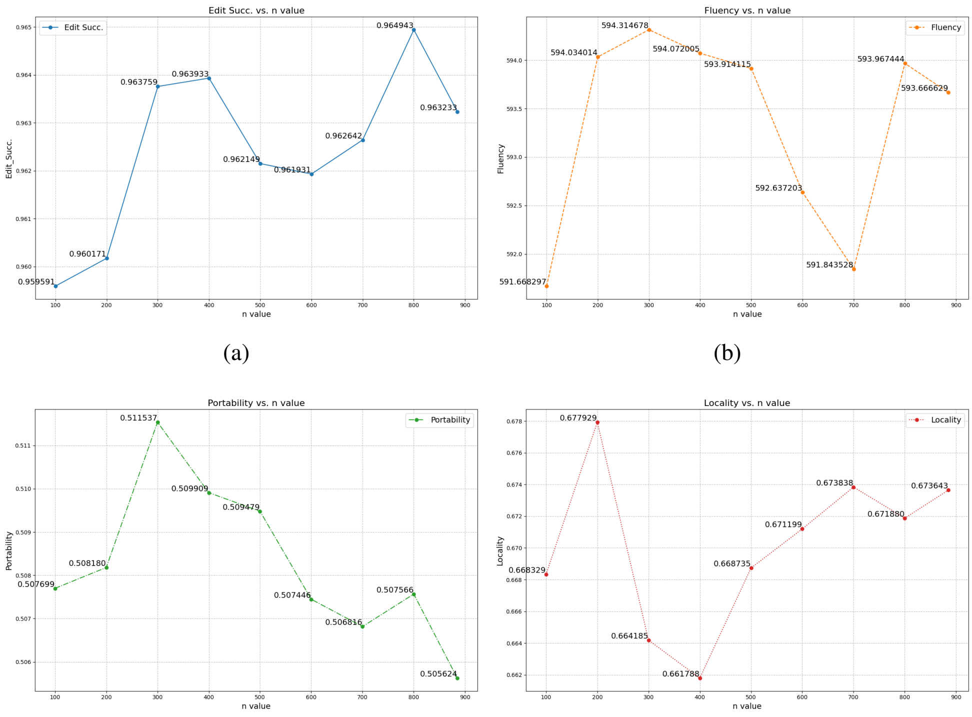
where $O(x_k)$ represents unrelated knowledge, f_θ is the original model, and $f_{\theta'}$ is the edited model.

- **Fluency (FLUE)**: Assesses the edited model's generative capacity using the weighted average of bi-gram and tri-gram entropies. Lower values indicate higher repetitiveness.

These results confirm that the **Knowledge Neuronal Ensemble (KNE)** method provides excellent and stable editing performance across different datasets. It consistently delivers the best results in terms of editing accuracy, while maintaining high portability and locality metrics. In some cases, it even outperforms previous methods in specific datasets.

(Note: The results of the comparative knowledge editing methods were sourced from the repository: EasyEdit GitHub.)

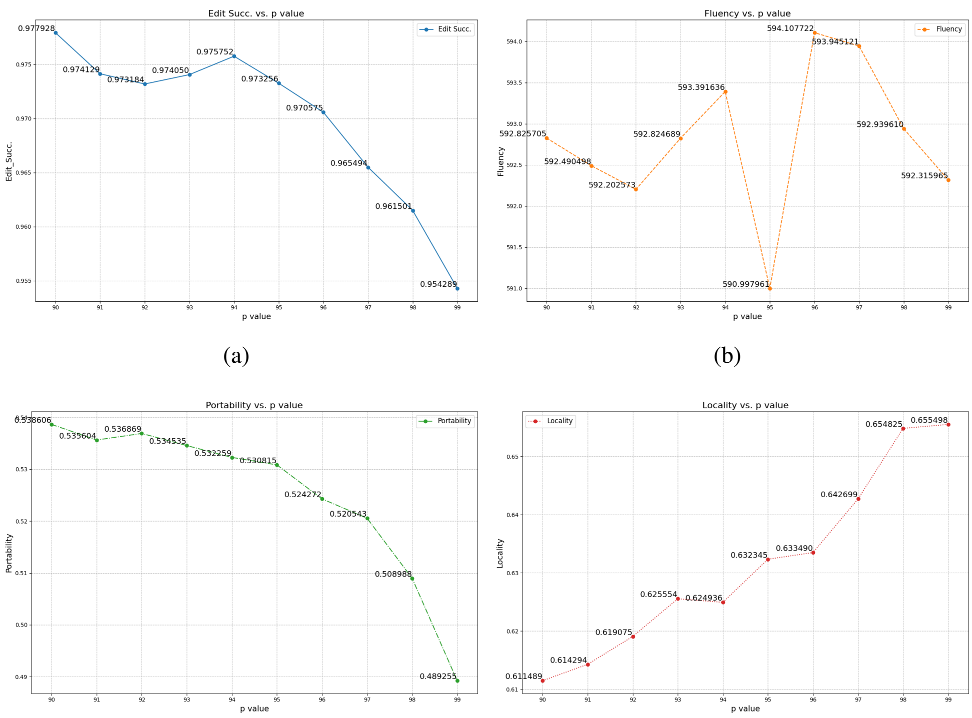
B Figures used in experimental discussion



(a) (b)

(c) (d)

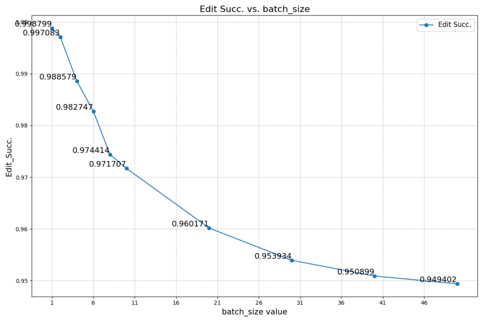
Figure 3: Effects of Localized Knowledge versus Full Dataset Localization



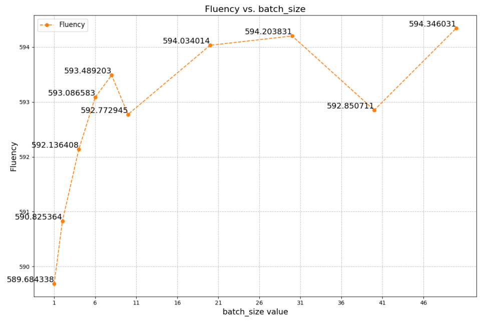
(a) (b)

(c) (d)

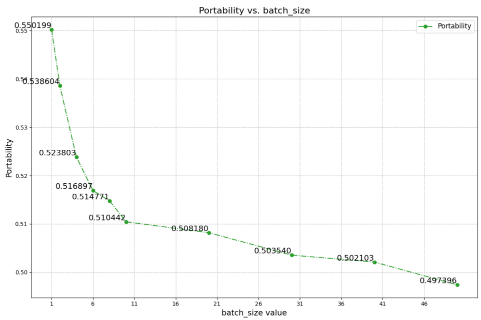
Figure 4: Performance Metrics Across Varying Parameter Settings for Knowledge Editing



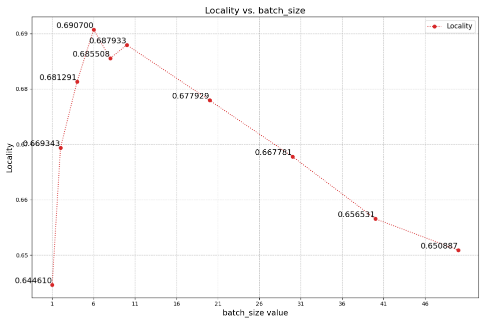
(a)



(b)



(c)



(d)

Figure 5: Performance Metrics Across Different Batch Sizes