Editing Large Language Models via Adaptive Gradient Guidance

Xiaojie Gu[1], Guangxu Chen[1], Shuliang Liu[1,2], Jungang Li[1], Aiwei Liu[3], Sicheng Tao[1], Junyan Zhang[1], Xuming Hu[1*]

[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Hong Kong University of Science and Technology [3]Tsinghua University
Guangzhou, Guangdong 510100 China
peettherapynoys@gmail.com, xuminghu97@gmail.com

Abstract

Large language models (LLMs) exhibit exceptional performance across various domains, and model editing holds significant potential to improve LLM safety and mitigate issues such as hallucinations. Existing model editing methods either modify the original hidden states directly to integrate new knowledge, which can lead to the accumulation of conflicts, or they achieve stable knowledge updates by adding extra parameters, which significantly increases computational costs. To address these challenges, we propose **AGRADE**, a method that **A**daptively guides the **GRAD**ient to compute **E**diting weights in alignment with the desired direction. By leveraging gradient differences across modules, AGRADE effectively reduces redundant information interference between adjacent modules, while controlling computational overhead and enhancing editing precision. We theoretically prove the effectiveness of AGRADE and conduct extensive experiments across multiple LLMs and datasets. The results show an average improvement of over 4% across three metrics, with an overall score increase of 11.98%.

Introduction

The world is constantly developing and changing, and humans are able to form a cognitive understanding of the world's dynamics through long-term learning. Despite the powerful capabilities of large language models (LLMs) (Wang 2021; Touvron et al. 2023; Jiang et al. 2023) in various domains have boosted confidence in the development of artificial general intelligence (AGI) (Fei et al. 2022). LLMs may unintentionally memorize sensitive or private information (Zhang et al. 2024a) during the pre-training stage. This can lead to biased outputs (Yu et al. 2023), contributing to fact-conflict hallucinations (Ji et al. 2023) or security problem (Li et al. 2024a). To effectively address these issues, model editing techniques have emerged as a promising solution. By making targeted adjustments to the model's behavior (Yao et al. 2023) without the need for full retraining, model editing can correct erroneous outputs, reduce the risk of privacy breaches (Wu et al. 2023), and simultaneously preserve the model's other knowledge and overall performance.

*Corresponding Author

Existing methods (Dai et al. 2022; Meng et al. 2022) fail to balance editing accuracy with computational cost (Wang et al. 2023). Some approaches (Meng et al. 2023; Li et al. 2024c) compute editing weights that represent the parameters of the editing target by computing the original gradients of the model. These methods enable precise small-scale edits but significantly increase computational complexity. Using a hypernetwork (Ha, Dai, and Le 2017) to bypass the original model's backpropagation path can alleviate the computational burden. However, when scaled to large-scale editing, the hypernetwork lacks stability (Mitchell et al. 2022a) and is unable to effectively coordinate new and existing knowledge (Pan et al. 2024). In this work, we adaptively guide the gradient towards the target direction to compute editing weights and update the hypernetwork, thereby balancing editing efficiency and precision. To maintain focus on the current editing target, we derive a projection direction from the gradients of adjacent editing modules—submodules within LLMs that require parameter modifications. We then eliminate the redundant gradient components along this direction, further preventing knowledge conflicts. Furthermore, to enhance the stability of the hypernetwork and enable rapid, precise editing, we project the gradients onto a subspace that captures the relationships between feature instances. This approach increases the information entropy along the target gradient direction, helping to stabilize the model's knowledge structure.

Our method has been thoroughly evaluated on multiple models and datasets. AGRADE demonstrates a 4.54% performance improvement in maintaining locality, highlighting its ability to effectively coordinate new and old knowledge. Additionally, the Editing Score shows a nearly 12% increase, while strikes an excellent balance between efficiency and editing speed, marking its significant progress in large-scale, precise editing with high efficiency. The contributions of our work are summarized as follows:

1. We propose AGRADE method for model editing, which adaptively alleviates gradient noise in the editing modules, minimizing conflicts between editing instances and the original knowledge.
2. We improve the projection paradigm of editing weights by guiding the gradient in a directional manner, thereby maintaining the intrinsic structure of the target information more stably.

3. Through extensive experiments and analysis on various LLMs and datasets, AGRADE achieves state-of-the-art performance in most editing scenarios, showcasing its efficiency and effectiveness.

Related Work

Locat-then-edit: ROME (Meng et al. 2022) and MEMIT (Meng et al. 2023) utilize causal tracing to locate key regions within transformer architectures—typically the multilayer perceptron (MLP) that encode specific knowledge. By directly injecting key-value pairs, these methods enable precise edits. Subsequent enhancements, including PMET (Li et al. 2024c), R-ROME (Gupta, Baskaran, and Anumanchipalli 2024), EMMET (Gupta, Sajnani, and Anumanchipalli 2024), WilKE (Hu et al. 2024), and AlphaEdit (Fang et al. 2024), refine optimization strategies for precise.

Memory-Augmented: Memory-augmented approaches introduce external or auxiliary memory components to facilitate knowledge updates. Methods like SERAC (Mitchell et al. 2022b) define the scope of edits using external memory, while CaliNet (Dong et al. 2022) and T-Patcher (Huang et al. 2023) focus on adding neurons to correct specific errors. GRACE (Hartvigsen et al. 2023), LTE (Jiang et al. 2024) and MELO (Yu et al. 2024) aim to mitigate the overhead of adding neurons by dynamically activating relevant parameters through memory retrieval. WISE (Wang et al. 2024) introduces dual-memory systems to resolve information conflicts during updates.

Meta-Learning: Meta-learning-based approaches predict parameter updates using auxiliary models. For instance, De Cao, Aziz, and Titov (2021) pioneered hypernetworks that predict weight updates based on required gradient adjustments. MEND (Mitchell et al. 2022a) improves efficiency through low-rank gradient decomposition, while MALMEN (Tan, Zhang, and Fu 2024) enhances precision using least squares to combine gradient updates with the model's original weights. DAFNet (Zhang et al. 2024b) enhances semantic interaction between instances using a multi-layer autoregressive diagonal attention mechanism.

Our method offers clear advantages over existing approaches. Unlike locate-then-edit methods, which rely on computationally intensive knowledge localization, we directly predict parameter updates, improving adaptability across different model architectures. Compared to memory-augmented methods that suffer from memory overload and slower inference when handling numerous edits (Wang et al. 2023), we eliminate additional memory overhead by directly adjusting offset parameters. Lastly, other meta-learning methods are often prone to instability and noise (Yao et al. 2023), our approach ensures stability through feature orthogonalization and minimizes interference with adaptive adjustment, ensuring robustness.

Methods

This work focus on batch editing, which is designed to perform large-scale edits simultaneously, with the model reloading its original parameters after each batch.

The overall pipeline of AGRADE is illustrated in the Figure 1, we use the hidden states and corresponding gradients of edited instances projected by the hypernetwork to adaptively remove redundant information, thereby calculating more precise editing weights. Then, we guide the accumulated gradients from the original model towards traget direction to update the hypernetwork parameters, enhancing its projection paradigm.

Preliminary

Model editing refers to the process of modifying the parameters of a pre-trained language model f_θ to obtain a new model $f_{\theta'}$ that incorporates updated knowledge. For a given intended **editing instance** (x_e, y_e), the objective is to ensure that $f_{\theta'}(x_e) = y_e$, while the original model fulfills the condition $f_\theta(x_e) \neq y_e$. Here, $x_e \in \mathcal{X}_{\text{edit}}$, where $\mathcal{X}_{\text{edit}}$ represents the set of all editing instances. Additionally, model editing needs to ensure that the post-edited model can respond to variations (**equivalent instance** $x'_e \in \mathcal{E}(x_e)$) of the editing instances, while retaining the original knowledge (**unrelated instance** $x_u \in \mathcal{U}(x_e)$). Specific examples of the three types of instances mentioned above can be found in Table 4. The corresponding metrics can be found in Appendix A.

The hypernetwork is a neural network that operates independently of the pre-trained model. Following (Mitchell et al. 2022a), it is designed as a multi-layer perceptron (MLP) with low-rank matrices. Its input consists of the hidden state and gradient of editing instamces, while its output is the projected hidden state and gradient, aligned with desired model adjustments.

Adaptive editing weights Computing

Zhang et al. (2024b) employs an attention mechanism for language modeling of each representation to compute editing weights, which restricts the editing scope. In contrast, our approach adaptively adjusts the editing direction based on gradient differences from adjacent modules, offering greater scalability. Initially, $\mathbf{H} \in \mathbb{R}^{T \times D_h}$ and $\nabla\mathbf{G} \in \mathbb{R}^{T \times D_g}$ represent the hidden states and gradients of editing instance, which provide the essential information for parameter updates. Here, T is the total number of tokens across all editing instances, D_h is the dimension of the hidden states, and D_g is the dimension of the gradients.

To reduce potential harm to the original knowledge, we avoid directly manipulating the raw features. Instead, we project the representations into an editing space using a hypernetwork, denoted as HyperNetwork$(\cdot)$. The projected representations remain closely aligned with the model's internal structure. This projection generates the projected hidden states $\tilde{\mathbf{H}}$ and projected gradients $\tilde{\nabla\mathbf{G}}$, as expressed by the following relationship: $\tilde{\mathbf{H}},\ \tilde{\nabla\mathbf{G}} = \text{HyperNetwork}(\mathbf{H}, \nabla\mathbf{G})$.

Next, we compute the adjustment coefficients $\mathbf{c} \in \mathbb{R}^{T \times 1}$ to scale the projected gradients,. These coefficients are derived as the element-wise weighted sum of the dot products between the original hidden states $\mathbf{H}$ and the projected hidden states $\tilde{\mathbf{H}}$, modulated by the learning rate η. For each

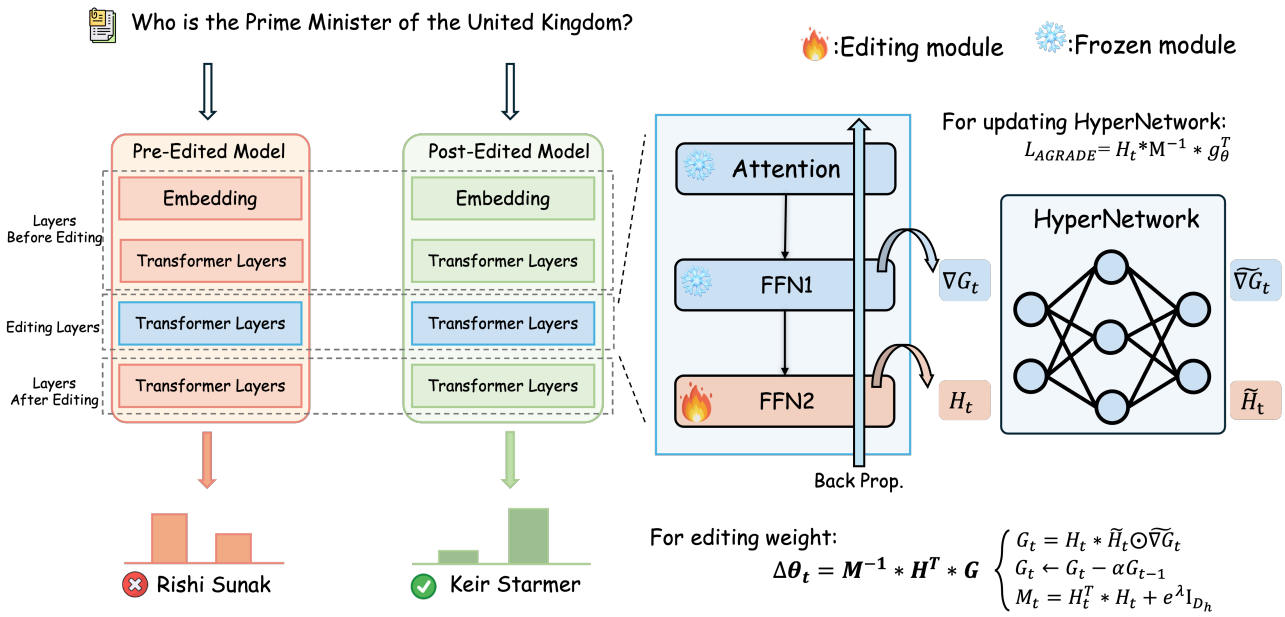

Figure 1: The overall structure of AGRADE.

token $i \in \{1, 2, \ldots, T\}$, the adjustment coefficients $\mathbf{c}_i$ are given by:

$$\mathbf{c}_i = -\eta \sum_{j=1}^{D_h} H_{ij} \tilde{H}_{ij}. \tag{1}$$

With the adjustment coefficients in place, the scaled gradient $\mathbf{G} \in \mathbb{R}^{T \times D_g}$ are updated by scaling the projected gradients as follows:

$$\mathbf{G} = \mathbf{c} \odot \nabla \tilde{\mathbf{G}}, \tag{2}$$

where $\odot$ denotes element-wise multiplication. These updated gradients will then be used to adjust the model's parameters during the editing process.

To ensure numerical stability when solving for the parameter updates, we introduce a regularized matrix $\mathbf{M}$:

$$\mathbf{M} = \mathbf{H}^\top \mathbf{H} + e^{\lambda} \mathbf{I}_{D_h}, \tag{3}$$

where λ is the regularization parameter and $\mathbf{I}_{D_h} \in \mathbb{R}^{D_h \times D_h}$ is the identity matrix. The term $\mathbf{H}^\top \mathbf{H}$ represents the covariance of the hidden states, while the regularization term $e^{\lambda} \mathbf{I}_{D_h}$ prevents $\mathbf{M}$ from becoming singular, ensuring that the linear system remains solvable and stable.

To ensure the independence of updates and prevent interference between modules, we eliminate components of the current gradient that align with the previous gradient. This is done by projecting the current gradient $\mathbf{G}$ onto the subspace orthogonal to the previous gradient $\mathbf{G}_{\text{prev}}$, effectively orthogonalizing it with respect to the previous module's gradient. The formula for this process is as follows:

$$\mathbf{G} \leftarrow \mathbf{G} - \left(\frac{\langle \mathbf{G}, \mathbf{G}_{\text{prev}} \rangle}{\langle \mathbf{G}_{\text{prev}}, \mathbf{G}_{\text{prev}} \rangle} \right) \mathbf{G}_{\text{prev}}, \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. This operation removes components of the current gradient that align with the previous gradient, adapting the current gradient to be independent of past updates. This helps eliminate redundant information that is unrelated to the current module update.

$\mathbf{M}$ represents the correlation between hidden states and provides a metric for the hidden state space. $\mathbf{H}^\top \mathbf{G}$ indicates the guidance direction for adjusting the model parameters through the hypernetwork. To achieve the optimal weights adjustment, which steers the model's output in the desired direction, we solve a system of linear equations to determine the editing weights $\Delta\theta \in \mathbb{R}^{D_h \times D_g}$:

$$\Delta\theta = \mathbf{M}^{-1} \mathbf{H}^\top \mathbf{G}, \tag{5}$$

The updated model parameters for the editing module are then obtained by adding the computed editing weights to the original parameters, as $\theta_{\text{updated}} = \theta_{\text{original}} + \Delta\theta$.

Directional HyperNetwork Training

Building upon the previous section, after calculating the editing weights $\Delta\theta$, our approach differs from (Tan, Zhang, and Fu 2024), which directly uses $\Delta\theta$ to aggregate gradient changes. This direct method carries the risk of the supernetwork overfitting to specific samples. To refine the model's updates more precisely toward the desired output, we first compute the gradient of the model's parameters, $\mathbf{g}_\theta \in \mathbb{R}^{D_h \times D_g}$, which is obtained during the backpropagation of editing instances.

To further ensure stable parameter updates in the hypernetwork and align them with the most informative directions in the feature space, we project the gradient onto a subspace that emphasizes directions with the highest variance or information content. The resulting projected gradient $\mathbf{P} \in \mathbb{R}^{N \times D_g}$ is given by:

$$\mathbf{P} = \mathbf{H} \left(\mathbf{H}^\top \mathbf{H} + e^{\lambda} \mathbf{I}_{D_h} \right)^{-1} \mathbf{g}_\theta^\top, \tag{6}$$

Model	Editor	zsRE				CounterFact				SelfCheckGPT	
		ES.	GS.	LS.	Score	ES.	GS.	LS.	Score	ES.	LS.
GPT-J (6B)	FT	23.29	22.07	16.57	20.19	11.71	6.67	26.64	10.99	46.86	36.59
	LoRA	34.18	33.05	27.09	31.11	11.93	5.33	30.07	9.85	89.93	41.19
	MEND	0.16	0.18	0.02	0.05	0.00	0.00	0.00	0.00	0.34	0.51
	MEMIT	86.12	67.15	25.87	46.04	95.50	**36.65**	11.49	24.04	72.12	44.15
	PMET	22.64	21.16	24.04	22.55	6.61	2.73	14.89	5.11	49.44	44.07
	MALMEN	98.86	88.70	24.49	48.22	62.56	22.30	39.08	34.72	32.44	44.03
	EMMET	76.53	63.30	26.27	44.83	86.50	29.58	8.79	18.85	67.58	43.96
	AGRADE	**98.92**	**88.76**	**29.41**	**54.17**	**96.09**	35.13	**53.14**	**52.01**	**90.05**	**44.57**
LLaMA2 (7B)	FT	32.70	32.25	44.54	35.70	17.42	14.02	20.41	16.88	57.51	**55.27**
	LoRA	50.56	48.13	44.61	47.64	30.03	21.62	49.39	30.06	84.64	52.77
	MEND	2.56	2.56	3.46	2.81	0.00	0.00	0.00	0.00	0.00	0.00
	MEMIT	59.65	53.93	26.85	41.35	53.18	34.33	13.60	24.70	58.08	55.13
	PMET	3.17	2.68	0.07	0.20	16.01	12.43	36.69	17.63	54.89	55.30
	MALMEN	**90.14**	82.89	45.81	66.69	45.97	26.08	37.88	34.69	39.78	52.95
	EMMET	34.86	32.58	17.35	25.64	59.54	32.76	15.20	26.52	63.56	54.13
	AGRADE	87.70	**83.01**	**59.99**	**74.77**	**83.41**	**57.81**	**72.54**	**69.65**	**89.69**	53.01
Mistral (7B)	FT	35.46	34.70	46.57	38.22	13.93	10.94	34.65	15.62	57.24	52.73
	LoRA	53.91	47.35	38.46	45.68	32.44	22.02	41.62	29.92	84.20	50.42
	MEND	1.92	1.92	1.89	1.91	0.00	0.00	0.00	0.00	3.68	3.19
	MEMIT	62.05	55.52	26.01	41.34	63.71	33.99	10.81	21.80	54.12	**55.13**
	PMET	0.02	0.03	0.12	0.03	15.43	10.98	26.52	15.50	56.44	52.61
	MALMEN	92.89	85.36	45.94	67.80	94.87	50.63	63.39	65.12	28.66	42.41
	EMMET	41.17	37.97	18.62	28.75	71.82	33.19	14.78	26.85	64.22	51.94
	AGRADE	**93.96**	**87.30**	**53.11**	**73.31**	**95.23**	**50.69**	**63.57**	**65.27**	**95.88**	37.53

Table 1: The results of batch editing with different methods across various models and datasets. The values in Bold indicate the best performance in each case.

The matrix $\mathbf{H}^\top\mathbf{H}$ captures the correlations between the hidden states of different tokens or instances, reflecting the internal structure of the model's learned knowledge. The regularization term $e^\lambda \mathbf{I}_{D_h}$ stabilizes this covariance matrix, preventing singularities and ensuring invertibility. The transformation $\left(\mathbf{H}^\top\mathbf{H} + e^\lambda \mathbf{I}_{D_h}\right)^{-1}$ serves as a weighted projection, emphasizing directions in the feature space that exhibit high variance or informativeness while minimizing noise from less significant directions. This optimization of gradient ensures that the gradient $\mathbf{g}_\theta^\top$ is directed into a subspace that is most aligned with the intrinsic structure of knowledge.

Next, we compute the scaled gradient $\mathbf{G} \in \mathbb{R}^{N\times D_g}$ using the previously defined equations (Equation 2 and Equation 4).

To enable the model to quantify the direction and magnitude of gradient updates, we compute the loss L_{AGRADE}. This loss is derived from the editing instances by calculating the element-wise product of the projected gradient $\mathbf{P}$ and the scaled gradient $\mathbf{G}$ across all elements, then summing over all dimensions:

$$L_{\text{AGRADE}} = \sum_{i=1}^{N}\sum_{j=1}^{D_g} \mathbf{P}^{(i,j)}\mathbf{G}^{(i,j)}. \tag{7}$$

Finally, as in (Mitchell et al. 2022a; Tan, Zhang, and Fu 2024), we compute the cross-entropy loss for equivalent instances (x'_e, y_e) from the set $\mathcal{E}(x_e)$, contributing to L_{equiv}, and the KL divergence for unrelated instances (x_u, y_u) from the set $\mathcal{U}(x_e)$, contributing to L_{unrel}. The total loss for updating the hypernetwork is the sum of the three components: $L_{\text{hyper}} = L_{\text{AGRADE}} + L_{\text{equiv}} + L_{\text{unrel}}$. The loss of this combination is backpropagated through the hypernetwork to update its parameters, enabling the output to better approximate the desired parameter shift based on the input instances, while ensuring locality to some extent.

Experiments

Experimental Setting

Dataset

The **zsRE** dataset (Levy et al. 2017) primarily consists of question-answer pairs covering a variety of entities and relationships, And **CounterFact** dataset (Meng et al. 2022) includes a large collection of synthetic counterfactual statements. Both datasets aim to evaluate whether models can correctly modify specific facts while minimizing their impact on the original knowledge base. Additionally, **SelfCheckGPT** (Manakul, Liusie, and Gales 2023) comprises instances of long-text hallucinations, designed to assess a

Variant	zsRE				CounterFact				SelfCheckGPT	
	ES.	GS.	LS.	Score	ES.	GS.	LS.	Score	ES.	LS.
AGRADE	87.70	83.01	59.99	74.77	83.41	57.81	72.54	69.65	89.69	53.01
w/o adjustment **c**	28.51	28.46	23.30	26.52	15.69	11.03	12.59	12.83	26.33	19.56
w/o regularization	84.49	78.64	57.49	71.52	80.12	52.48	68.90	65.14	83.41	52.64
w/o weights projection	82.69	79.33	51.30	67.88	79.61	53.28	65.49	64.37	83.45	46.67
w/o gradient projection	81.66	76.25	52.89	67.76	76.41	52.49	68.91	64.30	81.92	48.61

Table 2: Ablation Study of AGRADE for editing LLaMA2.

Editor	zsRE	CounterFact	SelfCheckGPT
FT	79.62s	75.41s	538.53s
LoRA	128.50s	103.63s	917.42s
MEND	90.61s	5.14s	68.81s
MEMIT	1303.54s	875.54s	2783.22s
PMET	750.46s	702.60s	3338.15s
MALMEN	42.94s	2.78s	45.43s
EMMET	674.19s	596.92s	3263.12s
AGRADE	**40.11s**	**2.56s**	**30.84s**

Table 3: Average **Wall Clock Time** for editing 100 instances using a single A800 on LLaMA2 across three datasets.

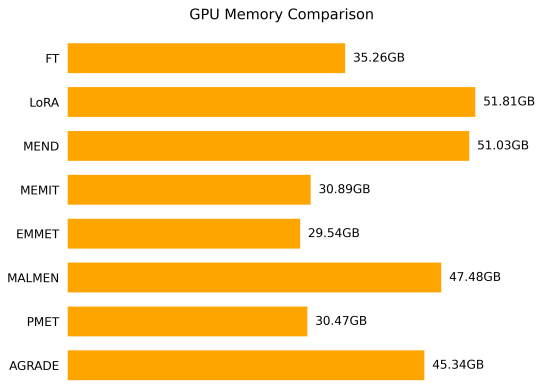

Figure 2: **GPU VRAM consumption** usage of different methods during editing of LLaMA2.

model's ability to handle inaccurate or fabricated information effectively. In all the experiments, we use a batch size of 8192 for the zsRE and CounterFact datasets, and a batch size of 256 for the SelfCheckGPT dataset.

Baselines

We compare AGRADE with several methods proficient in batch editing, including the straightforward self-supervised techniques like FT (Fine-Tuning) and LoRA (Hu et al. 2022), locate-then-edit style methods such as MEMIT (Meng et al. 2023), PMET (Li et al. 2024c), and EMMET (Gupta, Sajnani, and Anumanchipalli 2024), as well as hypernetwork-based approaches like MEND (Mitchell et al. 2022a) and MALMEN (Tan, Zhang, and Fu 2024). Beside, many recent methods (Fang et al. 2024; Zhang et al. 2024b; Wang et al. 2024; Ma et al. 2024) focus primarily on sequential editing, but their stability typically maxes out at 1.5k edits or even fewer. Some approaches (Li et al. 2024b; Jiang et al. 2024; Wang and Li 2024) support both sequential and batch editing; however, they can only handle up to 1k instances in batch editing while maintaining decent performance—a scale significantly lower than ours. Memory-augmented approaches (Zheng et al. 2023; Yu et al. 2024) face similar challenges as mentioned above. Consequently, we do not include comparisons with these methods. The details about baselines can be found in the Appendix B.

Overall Results

The experimental results in Table 1 show that AGRADE offers significant advantages across various scenarios, achieving a balanced improvement across the ES, GS, and LS metrics. Specifically, it results in average gains of 4.48% in ES, 4.02% in GS, and 4.54% in LS, leading to Editing Score increase of 11.98%. In the zsRE and CounterFact datasets, AGRADE consistently surpasses competitors such as EMMET and MALMEN, particularly excelling in ES and overall scores. While these competitors perform well on specific metrics, they fail to achieve the same level of balanced and sustained performance. Even in the SelfCheckGPT task, where locality success rates are generally lower, AGRADE maintains the highest ES rate, highlighting its robust generalization capabilities and optimization potential. Overall, our method demonstrates exceptional adaptability and stability, establishing itself as the leading approach among current techniques. The experimental details can be found in the Appendix C.

Ablation Study

To highlight the contribution of each key component to the overall performance, Table 4 presents the results of the ablation study for AGRADE. The results emphasize the importance of the adjustment coefficients **c**, which are crucial for properly scaling the projected gradients during the model editing process. Without these coefficients, it becomes difficult to control the magnitude and direction of gradient updates, leading to a significant decline in performance. Additionally, the absence of regularization results in a noticeable reduction in editing accuracy. Regularization plays a critical role in maintaining precision and stability, ensuring the editing process does not introduce errors or imbalances.

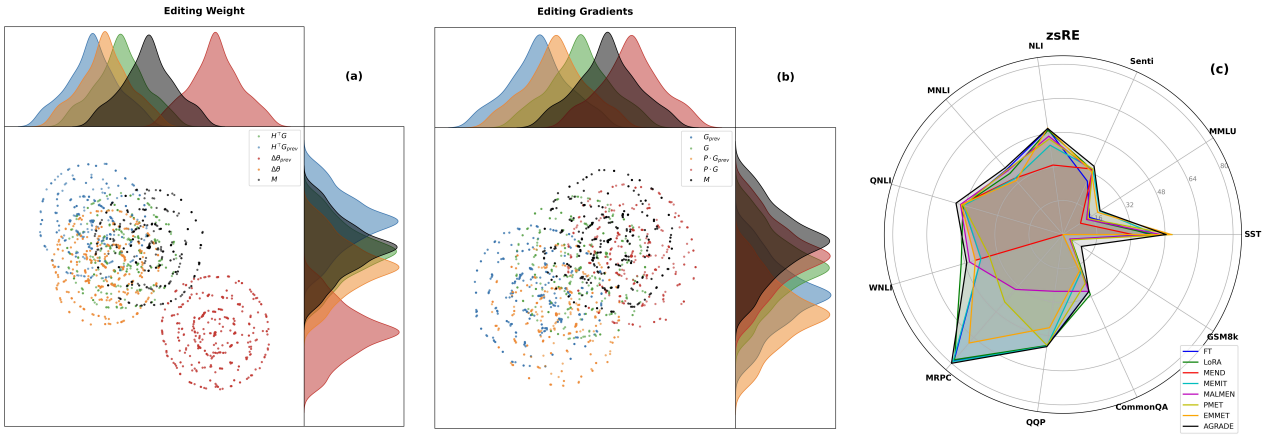

Figure 3: (a)&(b) Visualizes the distribution of features after dimensionality reduction. The top and right margins show the probability distributions of the data points along each dimension, revealing the similarities and differences between features. The plot highlights the relationships between feature representations before and after optimization, illustrating the effect of the optimization process on the alignment of the editing weights with the target matrix. (c) The performance of post-edited model across various evaluation benchmarks.

The study also shows that omitting the projection step in the weights calculation leads to a substantial drop in Locality Success, demonstrating the importance of this operation in removing redundant or irrelevant information from the model. This step helps ensure that the edits are focused and effective, preserving important features while minimizing unnecessary changes. Finally, the removal of gradient projection causes a decrease in performance, but still highlights the value of gradient directionality in guiding the model's learning process. By properly directing the gradient updates, this operation ensures that the model's edits align more accurately with the desired changes, improving the overall efficacy of the model editing process.

Performance Analysis

To gain deeper understanding of experimental results, we conduct a sampling analysis of the results obtained during the editing of LLaMA2 across the zsRE dataset. First, we perform rough dimensionality reduction using linear PCA (Pearson 1901) to efficiently denoise the features. Then, we apply t-SNE (Van der Maaten and Hinton 2008) to further preserve the local structure of the data through nonlinear mapping, projecting it into a 2D space, and visualize the distribution using Kernel Density Estimation (KDE) (Parzen 1962). The specific results are shown in Figure 3 (a)&(b).

The editing weights are obtained by solving a linear equation involving $\mathbf{M}$ and $\mathbf{H}^\top\mathbf{G}$. The essence of solving this linear equation is to find the parameters that, when applied to the matrix $\mathbf{M}$, yield a result close to $\mathbf{H}^\top\mathbf{G}$. In Figure 3 (a), we observe that the unprojected $\mathbf{H}^\top\mathbf{G}_{prev}$ is relatively distant from the corresponding editing weights $\Delta\theta_{prev}$, and the overlap in the KDE plot among the three is low. This indicates that the editing weights obtained in the current approach do not adequately capture the features of the editing examples. The matrix $\mathbf{M}$ is a covariance matrix that represents the primary directions of the editing example features. We find that the optimized $\mathbf{H}^\top\mathbf{G}$ vector is closer to $\mathbf{M}$, $\mathbf{H}^\top\mathbf{G}$ is well-constrained within the subspace defined by $\mathbf{M}$. This suggests that $\mathbf{M}$ is more effective at describing the information in $\mathbf{H}^\top\mathbf{G}$, with the projection directions of both vectors in the editing space highly aligned. This alignment implies that the solution to the linear equation is likely more stable, and the risk of numerical issues is significantly reduced. Additionally, the proximity of the optimized $\mathbf{H}^\top\mathbf{G}$ and the calculated solution $\Delta\theta$ further suggests that the editing weights now accurately reflect the important features of the target vector $\mathbf{H}^\top\mathbf{G}$, leading to more precise edits. The distribution of the gradient representations in Figure 3 (b) follows a similar pattern. Taken together, these visualizations demonstrate the effectiveness of AGRADE.

Efficiency Comparison

We analyze the efficiency of various editing methods in this study. Methods like MEMIT, PMET, and EMMET require downloading covariance matrices from the editing layer to perform edits. The average time to download a single covariance matrix is 279 minutes. Since we can use predownloaded covariance matrices during the editing process with these methods, the download time is excluded from the calculation of the average editing time. CounterFact contains shorter factual statements, while SelfCheckGPT consists of longer, more complex text hallucinations. Consequently, editing CounterFact instances is faster than editing zsRE, while editing SelfCheckGPT instances takes more time. As shown in Table 3, AGRADE achieves the fastest editing speed across all three datasets, outperforming other methods significantly in terms of processing time. Additionally, we measure GPU memory usage during the LLaMA2 model editing process, excluding the 59.21GB of VRAM required

Instance Type	Prompt	Pre-edited Output	Post-edited Output
Editing	Which is the manufacturer of Hyundai Global 900?	\nundai He Company	Hyundai Motor Company✓
Equivalent	Which company is known as a manufacturer of Hyundai Global 900?	\n2undai He Company	Hyundai Motor Company✓
Unrelated	Who sings i'll sleep when i'm dead?	\n1 Zevon	The Zevon✓

Table 4: Case study of editing LLaMA2.

for downloading the covariance matrices. As shown in Figure 2, the introduction of an additional hypernetwork increases VRAM consumption. While methods like MEMIT, PMET, and EMMET consume less VRAM, they are significantly slower and produce less effective results, offering no substantial advantage in terms of efficiency. Moreover, by using projection in adjacent modules instead of some redundant operations, we reduce the VRAM burden compared to MEND and MALMEN, both of which also employ hypernetworks. Furthermore, following the experimental settings from EasyEdit (Wang et al. 2023), when using LoRA for editing, we need to adjust the parameters stored in low-rank matrices for each layer, whereas traditional fine-tuning (FT) requires adjustments for only one layer. As a result, LoRA consumes more time and VRAM during the editing process. Overall, AGRADE strikes an optimal balance between VRAM efficiency and editing speed, outperforming other approaches in both effectiveness and resource usage.

Post-edited Model Analysis

We evaluate the performance of post-edited models, which are obtained by applying different methods to edit LLaMA2 on the zsRE and SelfCheckGPT datasets, across various benchmarks, including GLUE (Wang et al. 2019), MMLU (Hendrycks et al. 2021), GSM8k (Cobbe et al. 2021a), etc.(more details can be found in Appendix D), to compare the side effects of each method on the original model. The results are shown in Figure 3 (c). It is important to note that since CounterFact dataset consists of counterfactual data, editing it naturally introduces significant negative effects on the original model. Therefore, we do not include this dataset in our evaluation. As shown in the figure, after editing a large number of instances, our editing method exhibits relatively smaller overall side effects on the original model compared to other methods. The details of various benchmark and the performance of the post-edited model on the SelfCheckGPT dataset can be found in Figure 4.

Case Study

As shown in Table 4, AGRADE successfully incorporates editing instances into large language models, enabling them to correctly answer variant questions (equivalent instances) while minimizing the impact on other original knowledge (unrelated instances). This demonstrates the effectiveness of AGRADE.

Conclusion

Our work leverages adaptive gradient refinement and guide gradient to minimize interference and conflicts between adjacent modules, enabling precise parameter adjustments. Extensive experiments across diverse datasets and model validate the method's superiority over existing approaches in terms of editing accuracy, generalization, and locality preservation. Additionally, our approach achieves remarkable computational efficiency, significantly reducing resource consumption while maintaining high editing performance. These results establish the proposed method as a robust and scalable solution for overcoming challenges in large language model editing.

References

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021a. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021b. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; and Wei, F. 2022. Knowledge Neurons in Pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8493–8502.

De Cao, N.; Aziz, W.; and Titov, I. 2021. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6491–6506.

Dolan, B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.

Dong, Q.; Dai, D.; Song, Y.; Xu, J.; Sui, Z.; and Li, L. 2022. Calibrating Factual Knowledge in Pretrained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 5937–5947.

Fang, J.; Jiang, H.; Wang, K.; Ma, Y.; Wang, X.; He, X.; and Chua, T.-s. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

Fei, N.; Lu, Z.; Gao, Y.; Yang, G.; Huo, Y.; Wen, J.; Lu, H.; Song, R.; Gao, X.; Xiang, T.; et al. 2022. Towards artifi-

cial general intelligence via a multimodal foundation model. *Nature Communications*, 13(1): 3094.

Go, A.; Bhayani, R.; and Huang, L. 2009. Sentiment140: A Twitter corpus with sentiment labels. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1–2.

Gupta, A.; Baskaran, S.; and Anumanchipalli, G. 2024. Rebuilding rome: Resolving model collapse during sequential model editing. *arXiv preprint arXiv:2403.07175*.

Gupta, A.; Sajnani, D.; and Anumanchipalli, G. 2024. A unified framework for model editing. *arXiv preprint arXiv:2403.14236*.

Ha, D.; Dai, A. M.; and Le, Q. V. 2017. HyperNetworks. In *International Conference on Learning Representations*.

Hartvigsen, T.; Sankaranarayanan, S.; Palangi, H.; Kim, Y.; and Ghassemi, M. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Hu, C.; Cao, P.; Chen, Y.; Liu, K.; and Zhao, J. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Huang, Z.; Shen, Y.; Zhang, X.; Zhou, J.; Rong, W.; and Xiong, Z. 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.

Iyer, A.; Sharma, P.; Liang, P.; Xiong, C.; and Chen, X. 2017. Quora Question Pairs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12): 1–38.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Jiang, Y.; Wang, Y.; Wu, C.; Zhong, W.; Zeng, X.; Gao, J.; Li, L.; Jiang, X.; Shang, L.; Tang, R.; et al. 2024. Learning to edit: Aligning llms with knowledge editing. *arXiv preprint arXiv:2402.11905*.

Levesque, H.; Davis, E.; and Morgenstern, L. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Levy, O.; Seo, M.; Choi, E.; and Zettlemoyer, L. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, 333–342.

Li, Q.; Liu, X.; Tang, Z.; Dong, P.; Li, Z.; Pan, X.; and Chu, X. 2024a. Should We Really Edit Language Models? On the Evaluation of Edited Language Models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Li, S.; Deng, Y.; Cai, D.; Lu, H.; Chen, L.; and Lam, W. 2024b. Consecutive Batch Model Editing with HooK Layers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 13817–13833.

Li, X.; Li, S.; Song, S.; Yang, J.; Ma, J.; and Yu, J. 2024c. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18564–18572.

Ma, J.-Y.; Wang, H.; Xu, H.-X.; Ling, Z.-H.; and Gu, J.-C. 2024. Perturbation-Restrained Sequential Model Editing. *arXiv preprint arXiv:2405.16821*.

Manakul, P.; Liusie, A.; and Gales, M. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 9004–9017.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35: 17359–17372.

Meng, K.; Sharma, A. S.; Andonian, A. J.; Belinkov, Y.; and Bau, D. 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.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2022a. Fast Model Editing at Scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Mitchell, E.; Lin, C.; Bosselut, A.; Manning, C. D.; and Finn, C. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*, 15817–15831. PMLR.

Pan, K.; Fan, Z.; Li, J.; Yu, Q.; Fei, H.; Tang, S.; Hong, R.; Zhang, H.; and Sun, Q. 2024. Towards Unified Multimodal Editing with Enhanced Knowledge Collaboration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Parzen, E. 1962. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3): 1065–1076.

Pearson, K. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572.

Rajpurkar, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 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)*, 4149–4158.

Tan, C.; Zhang, G.; and Fu, J. 2024. Massive Editing for Large Language Models via Meta Learning. In *The Twelfth International Conference on Learning Representations*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Wang, B. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX.

Wang, P.; Li, Z.; Zhang, N.; Xu, Z.; Yao, Y.; Jiang, Y.; Xie, P.; Huang, F.; and Chen, H. 2024. WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models. *arXiv preprint arXiv:2405.14768*.

Wang, P.; Zhang, N.; Xie, X.; Yao, Y.; Tian, B.; Wang, M.; Xi, Z.; Cheng, S.; Liu, K.; Zheng, G.; et al. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models.

Wang, R.; and Li, P. 2024. LEMoE: Advanced Mixture of Experts Adaptor for Lifelong Model Editing of Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2551–2575.

Williams, A.; Nangia, N.; and Bowman, S. R. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018*, 1112–1122. Association for Computational Linguistics (ACL).

Wu, X.; Li, J.; Xu, M.; Dong, W.; Wu, S.; Bian, C.; and Xiong, D. 2023. DEPN: Detecting and Editing Privacy Neurons in Pretrained Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2875–2886.

Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; and Zhang, N. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10222–10240.

Yu, C.; Jeoung, S.; Kasi, A.; Yu, P.; and Ji, H. 2023. Unlearning bias in language models by partitioning gradients. In *Findings of the Association for Computational Linguistics: ACL 2023*, 6032–6048.

Yu, L.; Chen, Q.; Zhou, J.; and He, L. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19449–19457.

Zhang, R.; Lin, L.; Bai, Y.; and Mei, S. 2024a. Negative preference optimization: From catastrophic collapse to effective unlearning. *arXiv preprint arXiv:2404.05868*.

Zhang, T.; Chen, Q.; Li, D.; Wang, C.; He, X.; Huang, L.; Xue, H.; and Huang, J. 2024b. DAFNet: Dynamic Auxiliary Fusion for Sequential Model Editing in Large Language Models. *arXiv preprint arXiv:2405.20588*.

Zheng, C.; Li, L.; Dong, Q.; Fan, Y.; Wu, Z.; Xu, J.; and Chang, B. 2023. Can We Edit Factual Knowledge by In-Context Learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 4862–4876.

Appendix

A Metrics

Edit Success (ES) measures whether the modified model delivers the expected response for the editing instance:

$$ES = \frac{1}{|\mathcal{X}_{\text{edit}}|} \sum_{(x_e, y_e) \in \mathcal{X}_{\text{edit}}} \mathbb{I}\left[f_{\theta'}(x_e) = y_e\right] \tag{1}$$

Generalization Success (GS) evaluates whether the changes made to the model generalize to variant inputs:

$$GS = \frac{1}{|\mathcal{E}(x_e)|} \sum_{x'_e \in \mathcal{E}(x_e)} \mathbb{I}\left[f_{\theta'}(x'_e) = y_e\right] \tag{2}$$

where $\mathcal{E}(x_e)$ represents the set of equivalent instance. These are instances that express the same or closely related meaning as the edited instance x_e, such as paraphrased sentences or reformulated questions.

Locality Success (LS) measures the extent that editing have affected the original content within the model:

$$LS = \frac{1}{|\mathcal{U}(x_e)|} \sum_{x_u \in \mathcal{U}(x_e)} \mathbb{I}\left[f_{\theta'}(x_u) = f_{\theta}(x_u)\right] \tag{3}$$

where $\mathcal{U}(x_e)$ refers to the set of unrelated instance, which are instances that are unrelated to the edited instance (x_e, y_e) or its meaning.

Editing Score is calculated as the harmonic mean of ES, GS, and LS.

In these metrics, $\mathbb{I}[\cdot]$ denotes the indicator function, which equals 1 if the condition inside is true, and 0 otherwise.

B Baselines

FT involves further training a pre-trained model on a specific task to adjust its parameters and modify its outputs in a controlled manner. This enables FT to edit and optimize

the model for specific objectives without the need for full retraining.

LoRA inserts low-rank adaptation matrices into a pre-trained model, enabling efficient fine-tuning without altering the original model weights. By leveraging these matrices, LoRA captures key editing information, facilitating knowledge modification while avoiding the large-scale parameter updates typically needed in traditional fine-tuning methods.

MEND is a scalable model editing method that uses auxiliary networks to transform fine-tuning gradients into targeted updates, enabling fast and localized corrections in large pre-trained models. It efficiently handles high-dimensional gradients via low-rank decomposition, ensuring reliability, locality, and generality in model edits.

MEMIT enables large-scale memory updates by directly editing the parameters of critical MLP layers within transformer models. It leverages causal mediation analysis to identify key layers for factual recall, distributes updates across these layers to improve robustness, and applies explicit parameter optimization to integrate new knowledge with minimal error.

PMET optimizes the hidden states of Transformer Components while restricting updates to Feed-Forward Network (FFN) weights by utilizing only the optimized FFN hidden states as target knowledge representations, effectively avoiding unnecessary modifications to Multi-Head Self-Attention weights.

MALMEN achieves this by formulating parameter shift aggregation as a least square problem to resolve conflicts when editing multiple facts and decoupling computations between the hypernetwork and the language model to minimize memory usage and enable larger batch sizes

EMMET unifies ROME and MEMIT under the preservation-memorization objective, enabling batched edits with equality constraints for knowledge injection in transformers. It matches MEMIT's performance, effectively bridging their theoretical and practical differences.

C Experimental details

All of our experiments are conducted on a single A800 GPU, and the hypernetwork comprises 2 MLP blocks. The MLP layers in deep Transformers often capture the most information-rich representations (Mitchell et al. 2022a). Therefore, the editing module is implemented as the second Feed-Forward Network (FFN) in the MLP layers of the last six Transformer layers across all models.. The dataset splits are as follows: the zsRE dataset consists of 155,648 training samples and 16,384 evaluation samples, while the CounterFact dataset has 8,192 samples for both training and evaluation. For the SelfCheckGPT dataset, there are 256 samples for training and 512 samples for evaluation. The learning rates are as follows: for SelfCheckGPT, all models use a learning rate of 2e-5. For zsRE, all models use 1e-6. For CounterFact, LLaMA2 uses a learning rate of 1e-5, Mistral uses 1e-6, and GPT-J uses 2e-5.

D Evaluation

SST (Stanford Sentiment Treebank) (Socher et al. 2013) is a sentiment analysis task that uses sentences from movie reviews to classify the sentiment as either positive or negative. The dataset includes human-annotated sentiment labels for each sentence.

MRPC (Microsoft Research Paraphrase Corpus) (Dolan and Brockett 2005) is a benchmark for assessing semantic equivalence between two sentences. The task is to determine if two sentences have the same meaning or not.

MMLU (Massive Multi-task Language Understanding) (Hendrycks et al. 2021) evaluates the performance of models on a variety of tasks in both zero-shot and few-shot settings. It tests a model's ability to handle a broad range of tasks with minimal task-specific training.

Sentiment Analysis (Go, Bhayani, and Huang 2009) focuses on classifying text according to its sentiment, such as positive, negative, or neutral. It is often applied to user reviews, social media, and product feedback.

NLI (Natural Language Inference) (Williams, Nangia, and Bowman 2018) is a task that requires determining the logical relationship between two sentences, such as whether the premise supports, contradicts, or is neutral towards the hypothesis.

MNLI (Multi-Genre Natural Language Inference) (Williams, Nangia, and Bowman 2018) is a variant of NLI, where models must determine the entailment relationship between sentence pairs from a wide range of text genres, including fiction, government, and travel.

QNLI (Question Natural Language Inference) (Rajpurkar 2016) converts SQuAD into a binary classification task, where the model determines if a sentence contains the answer to a given question, labeled as entailment or neutral.

QQP (Quora Question Pairs) (Iyer et al. 2017) is a text similarity task where the model must identify whether two questions on the Quora platform are paraphrases of each other, i.e., if they express the same idea.

WNLI (Winograd Natural Language Inference) (Levesque, Davis, and Morgenstern 2012) is a more difficult variant of NLI, where models must resolve pronouns and other linguistic ambiguities to determine if a sentence entails or contradicts another.

CommonsenseQA (Talmor et al. 2019) is a question-answering dataset that focuses on testing a model's ability to use commonsense reasoning to answer questions that require deeper understanding beyond surface-level patterns.

GSM8K (Grade School Math 8K) (Cobbe et al. 2021b) is a dataset consisting of 8,000 math word problems designed for evaluating a model's ability to reason and solve grade-school-level arithmetic problems.

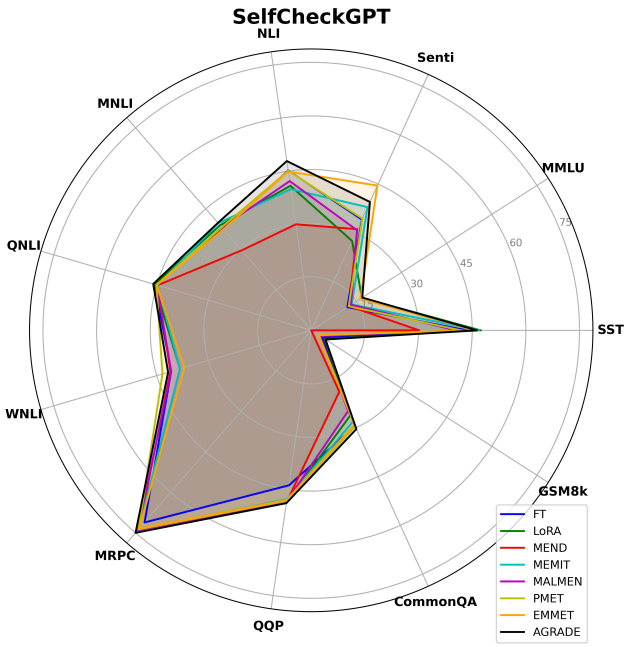

Figure 4: Evaluation of post-edited model.