Balancing Knowledge Updates: Toward Unified Modular Editing in LLMs

Jiahao Liu ljiahao@stu2023.jnu.edu.cn
School of Intelligent Systems Science and Engineering, Jinan University

Zijian Wang zwan0998@uni.sydney.edu.au
School of Computer Science, The University of Sydney

Kuo Zhao* zhaokuo@jnu.edu.cn
Dong Hu hudong@stu2023.jnu.edu.cn
School of Intelligent Systems Science and Engineering, Jinan University

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Knowledge editing has emerged as an efficient approach for updating factual knowledge in large language models (LLMs), typically achieved by first locating key knowledge-storage modules and then modifying their parameters. However, most existing methods focus exclusively on updating the weights of Multi-Layer Perceptron (MLP) modules, which are commonly identified as the primary repositories of factual information. Other important components, such as attention (Attn) modules—one of the core modules in LLMs—are often ignored during editing. This biased allocation of updates can leave residual outdated knowledge in the model and limit the effectiveness of knowledge editing. In this paper, we conduct comprehensive and systematic knowledge localization experiments on advanced LLMs, revealing that Attn modules play a substantial role in factual knowledge storage and retrieval, especially in earlier layers. Building on these insights, we propose *IntAttn-Edit*, a novel method that extends the associative memory paradigm to jointly update both MLP and Attn modules. Our approach employs a knowledge balancing strategy that proportionally allocates update magnitudes based on each module's measured contribution to knowledge storage. Extensive experiments on popular benchmarks demonstrate that *IntAttn-Edit* consistently achieves superior results over existing methods, delivering higher edit success, improved generalization, and robust knowledge preservation. Further empirical analysis shows that our knowledge balancing strategy enables the editing performance to remain within the optimal range across different settings.

Keywords: knowledge editing, large language models, knowledge balancing strategy

1. Introduction

Large language models (LLMs) have made remarkable progress in natural language understanding and generation. However, despite these advances, LLMs remain susceptible to factual inaccuracies, can generate potentially harmful or unsafe content, and often encode outdated knowledge due to their fixed training cut-off Zhang et al. (2023); Chen and Shu (2024). Retraining LLMs to address these limitations is both expensive and time-consuming Zheng et al. (2023b); Liu et al. (2021). As a result, there has been a growing

∗. Corresponding author

© 2025 J. Liu, Z. Wang, K. ZhaoD. Hu.

interest in knowledge editing techniques, which aim to enable precise, efficient, and cost-effective post-hoc updates to factual knowledge Yao et al. (2023); Wang et al. (2024). Such methods allow for the targeted refinement of specific model behaviors without extensive retraining or sacrificing overall performance.

Most mainstream knowledge editing methods adopt a locate-then-edit paradigm Meng et al. (2022a,b): they first identify the specific modules and layers that store the target factual knowledge, and then use these localization results to inform where to modify the parameters for knowledge updates. However, the dominant informing strategy is winner-takes-all: only the most significant knowledge storage modules ("winners") are updated, while other knowledge-bearing modules ("losers") remain unchanged. Specifically, extensive studies Geva et al. (2020, 2022) have shown that factual knowledge in large language models is primarily stored in MLP modules. The most popular editing methods, such as ROME Meng et al. (2022a), MEMIT Meng et al. (2022b), and AlphaEdit Fang et al. (2024), focus exclusively on updating MLP parameters. However, this biased winner-takes-all strategy neglects other important modules—such as Attn—which may result in residual outdated knowledge and prevent LLMs from fully acquiring new information. Recently, there has been a surge of research focusing on attention mechanisms Vaswani et al. (2017); Shazeer (2019); Ainslie et al. (2023); Liu et al. (2024), with mounting evidence showing that Attn layers are also crucial for encoding and retrieving knowledge Hase et al. (2023a), especially in advanced LLMs. These findings collectively indicate that attention, as a fundamental component of Transformer architectures, should not be ignored in knowledge editing.

In this paper, we first conduct a systematic and comprehensive investigation of knowledge localization in advanced LLMs using causal tracing. Our experiments demonstrate that not only do MLP modules serve as major repositories of factual knowledge, but Attn modules also play an equally critical role in knowledge storage. Motivated by these findings, we propose *IntAttn-Edit* (Integrating Attention to Edit), a novel method that, for the first time, simultaneously updates the parameters of both MLP and Attn modules during knowledge editing. Unlike previous approaches that exclusively update the dominant ("winner") module, *IntAttn-Edit* employs a knowledge balancing strategy that dynamically allocates update magnitudes between the two modules in proportion to their empirically measured contributions to factual knowledge. This dual-module strategy ensures that all relevant knowledge storage components are effectively updated, thereby reducing knowledge residuals and consistently yielding superior editing performance compared to single-module methods.

In summary, our main contributions are as follows:

(1) Through extensive and comprehensive knowledge localization experiments, we discover that Attn modules also play a vital role in factual knowledge encoding. Specifically, Attn modules can function as key-value associative memory structures, dynamically encoding factual associations alongside MLP modules.

(2) We propose *IntAttn-Edit*, a novel approach that extends the key-value memory editing paradigm from MLP layers to Attn modules for the first time. By introducing a knowledge balancing strategy, *IntAttn-Edit* dynamically allocates update magnitudes between MLP and Attn modules based on their empirical contributions to knowl-

edge storage. This ensures that all major knowledge-bearing components are effectively updated, reducing knowledge residuals and improving the overall robustness of knowledge editing.

(3) Through extensive experiments on advanced LLMs and popular benchmarks, we demonstrate that *IntAttn-Edit* consistently surpasses leading editing baselines across key performance metrics, particularly in batch editing scenarios. Furthermore, ablation studies confirm that our knowledge balancing strategy maintains editing effectiveness within an optimal range by dynamically allocating update magnitudes according to each module's empirical contribution, underscoring the robustness and practical value of our approach.

2. Related Works

Knowledge editing in large language models (LLMs) aims to efficiently update internal knowledge while maintaining overall model performance. Existing approaches are generally divided into three categories: (1) External Knowledge Editing, (2) Knowledge Integration, and (3) Intrinsic Editing.

2.1. External Knowledge Editing

LLMs possess in-context learning abilities, allowing for external knowledge integration through memory mechanisms or model augmentation. Representative methods include IKE Zheng et al. (2023a), which employs diverse demonstrations to guide factual updates, and MeLLo Zhong et al. (2023), which decomposes multi-hop queries and retrieves answers from sub-question memories. PokeMQA Gu et al. (2023) enhances reliability using programmable detectors and knowledge prompts, while SERAC Mitchell et al. (2022) combines classifiers with counterfactual models. However, these methods are limited by retrieval quality, susceptibility to harmful content, and potential knowledge conflicts Wang et al. (2023b); Liu et al. (2023); Wang et al. (2023a). Yu Yu et al. (2023) further examines the preference for context-based versus memory-based answers. In summary, external knowledge-guided techniques support knowledge editing but are constrained by the consistency and accuracy of retrieval.

2.2. Knowledge Integration

Knowledge integration approaches introduce new knowledge representations by adding or interpolating components at output or intermediate layers. Examples include knowledge patches as alternative output heads Murty et al. (2022), parameter-efficient adapters like LoRA Hu et al. (2022), and dynamic plug-in modules such as MELO Yu et al. (2024). GRACE Hartvigsen et al. (2023) utilizes a dynamic codebook for continual adaptation. Although these methods offer flexible knowledge updates, they must carefully manage knowledge conflicts and model capacity to ensure effective integration from multiple sources.

2.3. Intrinsic Editing

Intrinsic editing directly updates model parameters to encode new knowledge internally. Although full-model fine-tuning is resource intensive and prone to catastrophic forgetting, regularized strategies like Constrained Finetune Zhu et al. (2020) mitigate such issues. Meta-learning approaches (e.g., KE De Cao et al. (2021), SLAG Hase et al. (2023b)) and location-then-edit paradigms (e.g., Knowledge Neuron Dai et al. (2021), ROME Meng et al. (2022a)) enable more focused edits. Low-rank methods like MEND Mitchell et al. (2021) reduce parameter cost, yet challenges remain in handling multi-edit conflicts and preserving non-target knowledge. Location-then-edit strategies identify critical storage sites for precise updates: Knowledge Neuron attributes knowledge via gradient sensitivity, while ROME leverages causal analysis for minimal-weight optimization. Recent advances such as PMET Li et al. (2024), R-ROME Gupta et al. (2024), and AlphaEdit Fang et al. (2024) further enhance editing precision and locality.

3. Preliminaries

3.1. Knowledge Location in Transformer

Autoregressive Language Models. An autoregressive language model $\mathcal{F}_\theta$ predicts the probability distribution of the next token in a sequence by modeling dependencies among preceding tokens. Given an input sequence $[x_1, x_2, ..., x_z]$, the model generates the next token according to the conditional probability $\mathbb{P}(x_{z+1} \mid x_1, ..., x_z)$, which is parameterized by a D-layer Transformer decoder Vaswani et al. (2017):

$$\mathcal{F}_\theta(x_1, ..., x_z) = \text{softmax}(W_E \gamma(h_z^D)) = \mathbb{P}(x_{z+1} | x_1, ..., x_z), \tag{1}$$

where h_z^D denotes the hidden state at position z in the final (D-th) layer, W_E is the output embedding matrix, and γ represents layer normalization.

The hidden state h_z^l for token position z at layer l is computed recursively as follows:

$$h_z^l = h_z^{l-1} + a_z^l + m_z^l \tag{2}$$

where a_z^l and m_z^l denote the outputs of the Attn and MLP modules, respectively. The process can be expanded as:

$$a_i^l = W_o^l \cdot \text{ATTN}^l \left(\gamma \left(h_1^{l-1}, h_2^{l-1}, ..., h_i^{l-1} \right) \right), \quad m_i^l = W_{\text{out}}^l \sigma \left(W_{\text{in}}^l \gamma \left(h_i^{l-1} \right) \right) \tag{3}$$

where W_o^l is the Attn output projection matrix, $\text{ATTN}^l(\cdot)$ denotes the multi-head self-attention operation at the l-th layer, W_{in}^l and W_{out}^l are the input and output matrices of the MLP, and σ is a non-linear activation function.

Causal Tracing for Knowledge Localization. Causal tracing Meng et al. (2022a) provides an effective means for identifying the storage and processing locations of factual knowledge within Transformer models. This procedure involves constructing a clean prompt (X_{clean}) with its associated answer r, a corrupted prompt (X_{cor}), and executing three model runs:

(1) **Clean run**: Execute the model on X_{clean} to record intermediate activations of specific computational components.

(2) **Corrupted run**: Process X_{cor} to generate predictions under knowledge-disrupted conditions. To generate X_{cor}, Gaussian noise $\mathcal{N}(0, \nu)$ is added to the embeddings of some key tokens, where ν is 3 times the standard deviation of the token embedding text set.

(3) **Patched run**: Intervene by reinstating cached activations from X_{clean} to a designated component during X_{cor} inference, while preserving corruption in other components. Output deviations quantify the component's causal role in factual recall.

Prior causal tracing studies Meng et al. (2022a,b) on classical models such as GPT-2 have shown that MLP modules, particularly their output projections, are pivotal for factual memory retrieval. As a result, most knowledge editing methods Fang et al. (2024); Gupta et al. (2024) have focused primarily on identifying and updating MLP output parameters to insert or modify facts. In contrast, despite being a central component of Transformer architectures, the Attn module's role in knowledge storage has remained largely underexplored in mainstream knowledge editing research.

3.2. Linear associate memory and knowledge update

Linear associative memory. Most editing methods interpret the output weight matrices of FFN layers, W_{out}, as linear associative memories Geva et al. (2020). In this view, any linear operation W serves as a parameterized key–value store, where a matrix of keys $K = [k_1 \mid k_2 \mid \ldots]$ maps to a set of values $V = [v_1 \mid v_2 \mid \ldots]$, aiming for $WK \approx V$. To best fit these associations, W is typically solved by minimizing the squared error, resulting in the closed-form solution $W = VK^+$ using the Moore–Penrose pseudoinverse.

Some researches Bau et al. (2020); Geva et al. (2020) show that a new key–value pair (k^*, v^*) can be optimally integrated into the associative memory by solving a constrained least-squares problem. Specifically, the updated weight matrix $\hat{W}$ is required to satisfy:

$$\hat{W}k^* = v^*, \quad \text{where} \quad \hat{W} = W + \Delta \tag{4}$$

The goal is thus to solve for the update Δ that enables the direct insertion of (k^*, v^*) while minimally disturbing the original key–value structure.

Model Editing in LLMs. Knowledge editing in LLMs updates factual associations by directly modifying the parameters responsible for key-value mappings. When knowledge is represented as (subject, relation, object) triples (e.g., s = "LeBron James", r = "plays sport", o = "basketball"), the MLP maps semantic input to the factual object. In particular, the output weight matrix W^l_{out} in specific MLP layers is frequently interpreted as a linear associative memory. Within this framework, the key vector for each input, $\mathbf{k} = \sigma(W^l_{\text{in}}\gamma(h^{l-1}))$, encodes the relevant (subject, relation) semantics. The MLP output m^l serves as the value corresponding to the target knowledge, i.e., $\mathbf{v} = m^l$. For convenience, we denote W_0 as W^l_{out} in the following.

Suppose each editing operation aims to update u new pieces of knowledge, each in the form of a (s, r, o) triple. Given the new knowledge key-value pairs, we seek a parameter

update Δ that encodes the new facts while preserving the irrelevant existing knowledge. Let $K_1 = [k_1, \ldots, k_u] \in \mathbb{R}^{d_k \times u}$ and $V_1 = [v_1, \ldots, v_u] \in \mathbb{R}^{d_v \times u}$ denote the key and value matrices for the new knowledge, where d_k and d_v are the dimensions of keys and values, respectively. Similarly, (K_0, V_0) represent the key-value pairs corresponding to the original knowledge that should be maintained. Current methods Gu et al. (2024); Meng et al. (2022b) typically formulate editing as an optimization problem, seeking a perturbation Δ to update W_0 so that the model's new associations match the targets while preserving previous knowledge:

$$\Delta = \arg\min_{\tilde{\Delta}} \left(\left\| (W_0 + \tilde{\Delta}) K_1 - V_1 \right\|^2 + \left\| (W_0 + \tilde{\Delta}) K_0 - V_0 \right\|^2 \right) \tag{5}$$

This admits a closed-form solution:

$$\Delta = (V_1 - W_0 K_1) K_1^\top \left(K_0 K_0^\top + K_1 K_1^\top \right)^{-1} \tag{6}$$

This framework is naturally extendable to multi-layer scenarios. In such cases, the residual R (desired minus current outputs) is distributed across critical MLP layers. Let $\mathcal{S}$ denote these layers, $l \in \mathcal{S}$, and $L \triangleq \max(\mathcal{S})$ the highest index. For each layer l, the residual and incremental weight update are computed as

$$R^l = \frac{V_1 - W_0 K_1}{L - l + 1}, \quad \Delta^l = R^l K_1^\top (K_0 K_0^\top + K_1 K_1^\top)^{-1} \tag{7}$$

Here, (K_0, V_0) represent the key-value pairs of existing knowledge to be preserved, while (K_1, V_1) correspond to the new knowledge being introduced. In practice, obtaining the complete K_0 that covers all knowledge stored in an LLM is infeasible. To approximate K_0, previous studies Meng et al. (2022b) suggest randomly sampling a large number of (subject, relation, object) triplets from external corpora such as Wikipedia. For example, it is common to use 100,000 triplets, resulting in $K_0 \in \mathbb{R}^{d_k \times 100{,}000}$.

4. Methodology

4.1. Discovering Attention as Memory in LLMs

In this section, we revisit the mechanisms underlying factual knowledge localization within LLMs through a systematic causal tracing analysis on a advanced model, Qwen2.5-7B. Specifically, we employed activation patching Meng et al. (2022a) to quantify the causal contributions of both the MLP and Attn modules across different layers and token positions.

To quantify the causal contribution of model components to factual recall, we employ two core metrics Zhang and Nanda (2023) comparing model performance between *corrupted* ($*$) and *patched* (pt) runs:

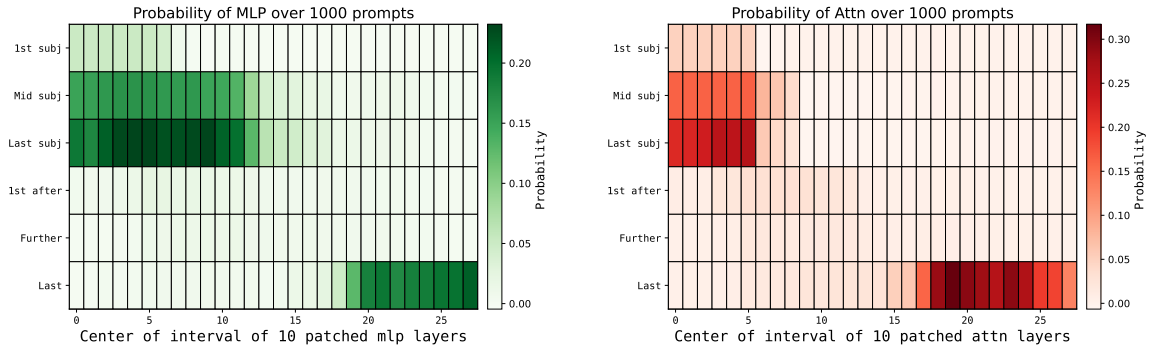
- **Probability**: The patching effect is defined as $P(r) = P_{pt}(r) - P_*(r)$, where r denotes the correct answer.
- **Logit difference**: The normalized restoration of the logit gap between the target answer r and a contrastive incorrect answer r', quantifying the component-specific

corrective capability:

$$LD(r, r') = \frac{LD_{pt}(r, r') - LD_{*}(r, r')}{LD_{cl}(r, r') - LD_{*}(r, r')} \tag{8}$$

where $LD(r, r') = Logit(r) - Logit(r')$, and r' is a frequent incorrect answer. Here, $Logit(\cdot)$ refers to the unnormalized output score for each candidate answer produced by the model before applying the softmax function.

As shown in Figure 1, the MLP module displays strong causal effects in the middle layers, particularly at the last subject token, which is consistent with previous studies. In contrast, the Attn module exhibits pronounced causal influence in the early layers (approximately layers 1 to 5), especially at the middle and last subject token positions. This highlights the critical role of Attn in establishing relational pathways for factual recall at the initial stages of processing, rather than merely aggregating contextual information. Together, these results underscore that both MLP and Attn modules play essential roles in the storage and retrieval of factual knowledge within LLMs.

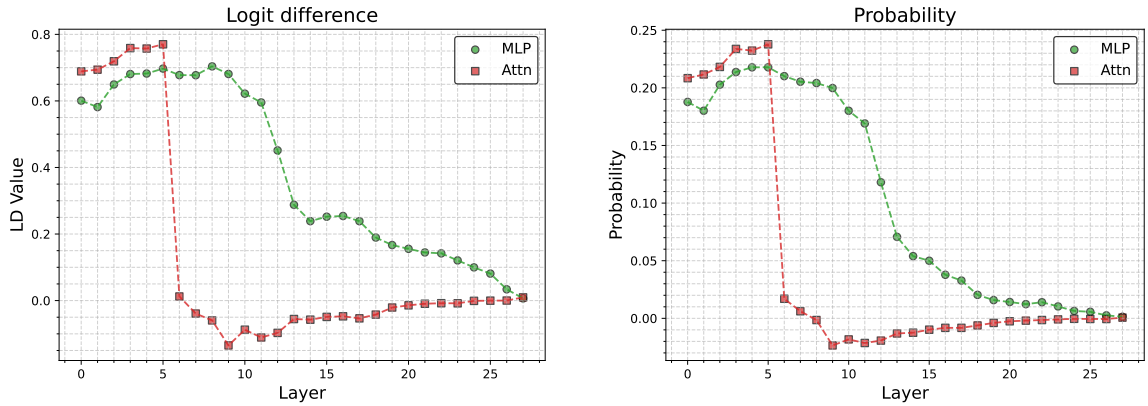


(a) Probability of MLP over 1000 prompts (b) Probability of Attn over 1000 prompts

Figure 1: Heatmaps of causal influence from MLP and Attn layers on factual recall in Qwen2.5-7B.

A closer inspection of Figure 2 reveals that the Attn module exerts its strongest causal influence in the earliest layers, as indicated by a rapid increase and subsequent sharp decline in both logit difference and probability metrics. The contribution of Attn to factual recall peaks within the first five layers and then quickly diminishes. This pattern suggests that **Attn plays a vital role in the initial identification and encoding of factual associations**, functioning as a semantic filter and router early in the model's processing pipeline. These results highlight the importance of recognizing and leveraging the knowledge storage function of Attn modules—alongside MLP modules—when considering the mechanisms of knowledge representation and update in large language models.

Motivated by these empirical observations, we find that Attn modules also play a crucial role in factual knowledge encoding, particularly in the early layers. This suggests that Attn modules may function as key-value associative memory structures, dynamically encoding factual associations alongside MLPs. Based on this insight, we explicitly incorporate Attn



(a) Logit Difference (b) Probability

Figure 2: Layerwise Causal Effects of Attn and MLP at the Last Subject Token.

modules into the linear associative memory framework for knowledge editing. To distinguish between layer indices, we denote critical Attn layers as l_{attn} and key MLP layers as l_{mlp}, which are independently determined via causal tracing and generally correspond to different sets of layers within the transformer architecture.

$$\underbrace{a_i^{l_{\text{attn}}}}_{v} = W^{l_{\text{attn}}} \cdot \underbrace{\text{ATNN}^{l_{\text{attn}}}\left(\gamma(h_1^{l_{\text{attn}}-1}, h_2^{l_{\text{attn}}-1}, \ldots, h_i^{l_{\text{attn}}-1})\right)}_{k} \tag{9}$$

Here, the hidden states input to the Attn module constitute the key (k), and the module's output is linearly projected by the weight matrix $W^{l_{\text{attn}}}$ to yield the corresponding value (v). For convenience, we denote $W^{l_{\text{attn}}}$ as W_{o}^{l} in the following.

4.2. IntAttn-Edit

In this section, **we present *IntAttn-Edit*, a novel model editing approach that, for the first time, explicitly targets both the Attn and MLP modules for parameter updates**. Unlike prior methods that focus solely on the MLP layers, *IntAttn-Edit* simultaneously performs associative memory updates within both modules. Drawing on empirical evidence that Attn modules can serve as associative memories, we dynamically construct attention-specific key-value pairs to facilitate targeted edits. Crucially, to achieve balanced and effective knowledge modification across both pathways, **we introduce a knowledge balancing strategy that proportionally allocates update magnitudes based on the cumulative causal contributions of each module.** As illustrated in Figure 3, this dual-pathway mechanism ensures that parameter updates are adaptively scaled according to the empirical importance of the MLP and Attn modules, enabling comprehensive and robust knowledge editing.

Deriving Attn Knowledge K-V Pairs. For each critical Attn layer l_{attn}, we construct key and value pairs analogously to the MLP case. The keys are derived from the positionally encoded hidden states of the previous layer, while the values are obtained via the Attn

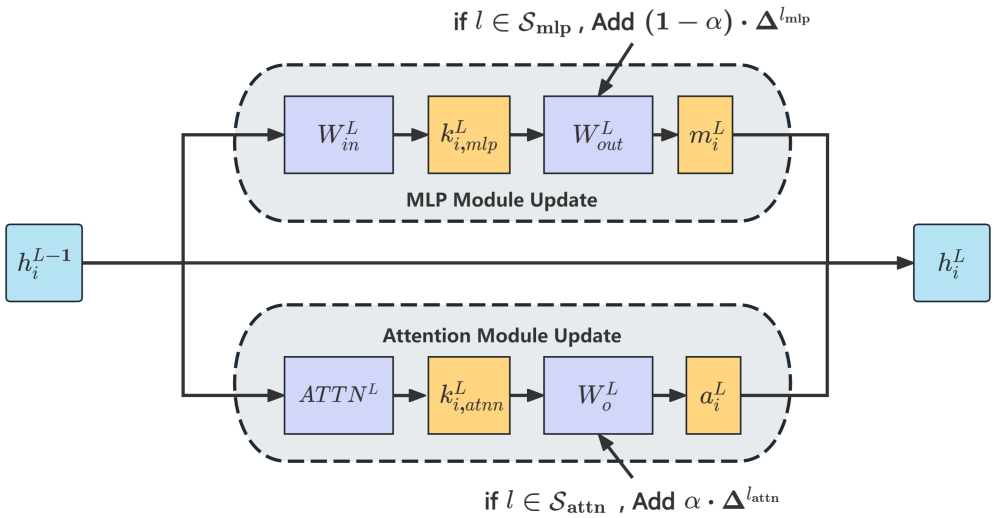


Figure 3: Schematic illustration of the dual-pathway parameter update mechanism in *IntAttn-Edit*. Both MLP and Attn modules are edited in parallel, with update strengths modulated by the allocation factor α based on the measured causal contributions of each module.

output projection. Specifically,

$$k_{\text{attn}} = \text{ATTN}^{l_{\text{attn}}}\left(\gamma\left(h_1^{l_{\text{attn}}-1}, \ldots, h_i^{l_{\text{attn}}-1}\right)\right), \quad v_{\text{attn}} = W^{l_{\text{attn}}} k_{\text{attn}} \tag{10}$$

where $h_i^{l_{\text{attn}}-1}$ denotes the hidden states and $W^{l_{\text{attn}}}$ is the Attn projection matrix.

Once the key-value matrices $K_{1,\text{attn}}$ and $V_{1,\text{attn}}$ for new knowledge are constructed, we compute the update $\Delta^{l_{\text{attn}}}$ for a single Attn layer by solving the same form of closed-form equation as previously established (see Eq. (6)). For multi-layer scenarios, this update is broadcast across the designated critical Attn layers following the proportional residual allocation strategy (see Eq. (7)).

Finally, the updated Attn projection weights for layer l_{attn} are given by:

$$\hat{W}^{l_{\text{attn}}} = W^{l_{\text{attn}}} + \Delta^{l_{\text{attn}}} \tag{11}$$

By formulating the Attn module within the linear associative memory paradigm, we enable direct knowledge editing for Attn layers through explicit key-value association updates.

Knowledge Balancing Strategy. In this section, **we propose a Knowledge Balancing Strategy to incorporate all knowledge-storage modules—both Attn and MLP—into the knowledge editing process.** This strategy ensures that the parameter updates for each module are dynamically allocated in proportion to their empirically measured contributions to factual knowledge. It directly addresses a fundamental limitation of prior methods, which typically focus only on updating the dominant module while neglecting the potentially significant role of other modules.

A core challenge is how to quantitatively assess and balance the contributions of different modules to factual recall during editing. To solve this, we leverage causal tracing to measure the actual impact (i.e., causal effect) of each module on factual prediction. Specifically, for each module (MLP or Attn), we compute the cumulative logit difference restoration, $LD^{(l)}(r, r')$, across all layers of that module. This metric quantifies how much restoring the activations of a given layer contributes to recovering the correct factual output, thereby providing a direct empirical estimate of the module's knowledge storage responsibility.

Based on these causal effect measurements, **we define a balance factor α, which represents the proportion of total knowledge storage attributable to the Attn module relative to the combined storage of both modules.** The calculation is as follows:

$$\alpha = \frac{\sum_{l \in \text{Attn}} LD^{(l)}(r, r')}{\sum_{l \in \text{MLP}} LD^{(l)}(r, r') + \sum_{l \in \text{Attn}} LD^{(l)}(r, r')} \tag{12}$$

where $LD^{(l)}(r, r')$ denotes the normalized restoration of the logit gap for each layer l, quantifying its specific causal contribution to factual recall.

This causal-effect-guided allocation offers several advantages. First, it allows our method to adaptively allocate update magnitudes between Attn and MLP modules according to their real knowledge contribution, rather than following a fixed or arbitrary rule. Second, by distributing updates in proportion to each module's role in knowledge storage, our approach reduces the risk of knowledge residuals—that is, leftover or conflicting information in modules that were ignored by prior winner-takes-all strategies. Finally, this fine-grained balancing ensures that all major knowledge-storage components are effectively synchronized during editing, resulting in superior factual accuracy, model consistency, and generalization in practice.

The final parameter updates are then scaled according to this balance factor, as detailed in the following equations:

$$\hat{W}^{l_{\text{mlp}}} = W^{l_{\text{mlp}}} + (1 - \alpha)\Delta^{l_{\text{mlp}}}, \quad \hat{W}^{l_{\text{attn}}} = W^{l_{\text{attn}}} + \alpha\Delta^{l_{\text{attn}}} \tag{13}$$

Here, $W^{l_{\text{mlp}}}$ and $W^{l_{\text{attn}}}$ denote the output weight matrices of the MLP and Attn layers, respectively.

5. Experiments

5.1. Exprimental Setup

Base LLMs and Baseline Methods. We choose trending autoregressive LLM models **Mistral-7B** Jiang et al. (2023), and **Qwen2.5-7B** Qwen et al. (2025) for evaluation. We compare our method against several model editing baselines, including **Fine-Tuning (FT)** Zhu et al. (2020), **ROME** Meng et al. (2022a), **R-ROME** Gupta et al. (2024), **MEMIT** Meng et al. (2022b), and **AlphaEdit** Fang et al. (2024). All methods are implemented using EasyEditZhang et al. (2024).

Datasets and Metrics. We evaluate our method using two widely adopted benchmarks: the **ZsRE** dataset Levy et al. (2017) and the **WikiData counterfact** dataset Meng et al. (2022a). We employ **Edit Success**, **Portability**, **Locality**, and **Fluency** as evaluation metrics Zhang et al. (2024). The formal definitions are as follows:

- **Edit Success** (**Edit Succ.**) measures whether the edited model outputs the correct answer for the edited knowledge:

$$\text{Edit Succ.} = \frac{1}{N} \sum_{(x_k, y_k^*)} \mathbb{1} \left\{ \arg\max_y f_{\theta'}(y \mid x_k) = y_k^* \right\} \tag{14}$$

  where x_k and y_k^* are the k-th edit prompt and its ground-truth answer, $f_{\theta'}$ is the post-edit model, and N is the number of edited samples.

- **Portability** evaluates the accuracy of the post-edited model on the portability set:

$$\text{Portability} = \frac{1}{N} \sum_{(x_k, y_k^*)} \mathbb{1} \left\{ \arg\max_y f_{\theta'}(y \mid x_k) = y_k^* \right\} \tag{15}$$

  where (x_k, y_k^*) now refer to portability prompts and their gold answers.

- **Locality** measures the extent to which the model preserves its original predictions on unrelated samples:

$$\text{Locality} = \mathbb{E}_{x_k, y_k^* \sim O(x_k)} \mathbb{1} \left\{ f_{\theta'}(y \mid x_k) = f_\theta(y \mid x_k) \right\} \tag{16}$$

  where $O(x_k)$ denotes the locality set for x_k, and f_θ is the original model.

- **Fluency** quantifies the entropy of n-gram distributions to measure generation diversity and penalize repetition:

$$\text{Fluency} = -\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k) \tag{17}$$

  where $g_n(k)$ denotes the frequency of the k-th n-gram in the generated text.

Hyperparameters. For the hyperparameter α in our method, we set its value based on the cumulative causal effect of each model module as determined by empirical analysis. For Qwen2.5-7B, we set $\alpha = 0.3$, with the critical MLP layers and Attn layers selected as $\mathcal{R}_{\text{mlp}} = \{4, 5, 6, 7, 8\}$ and $\mathcal{R}_{\text{attn}} = \{2, 3, 4\}$, respectively. For Mistral-7B, we set $\alpha = 0.1$, and choose $\mathcal{R}_{\text{mlp}} = \{4, 5, 6, 7, 8\}$ and $\mathcal{R}_{\text{attn}} = \{3, 4, 5, 6\}$. These settings are determined according to the causal tracing results and the distribution of factual memory across modules in each model.

5.2. Batch Knowledge Editing on LLMs

We systematically conduct batch knowledge editing experiments using *IntAttn-Edit* and baseline methods. Table 1 summarizes the performance of all methods on the ZsRE benchmark under three different batch sizes: 100, 300, and 500 edits per batch (i.e., $T = 100, 300, 500$). Table 2 presents the corresponding results on WikiData counterfact. **In all tables, bold values indicate the best performance, and † marks the second best.**

Table 1: Batch Knowledge Editing Performance on ZsRE.

Method	**Mistral-7B**											
	T=100				T=300				T=500			
	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.
IntAttn-Edit	**92.70**	**56.25**	34.88†	568.08†	**87.75**	**56.30**	34.89†	564.44†	**86.15**	**55.42**	34.40†	568.20†
FT	39.99	49.04	**45.79**	**575.94**	39.99	49.04	**45.76**	**575.61**	39.85	48.85	**45.75**	**575.52**
ROME	5.05	1.24	0.52	443.30	3.08	0.80	0.55	467.93	1.34	0.51	0.51	449.34
R-ROME	4.86	1.22	0.53	443.72	2.59	0.61	0.57	444.04	2.11	0.73	0.52	470.61
MEMIT	91.66†	55.71†	33.88	564.61	84.29†	55.63†	33.35	563.44	83.58†	54.22†	32.76	563.31
AlphaEdit	85.10	51.44	28.76	554.43	73.16	45.50	25.83	546.26	57.49	37.83	21.06	525.05
Method	**Qwen2.5-7B**											
	T=100				T=300				T=500			
	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.
IntAttn-Edit	**96.98**	**55.04**	**34.40**	**572.78**	93.96†	**55.41**	**34.59**	569.67†	**92.10**	**56.32**	**34.44**	562.54†
FT	25.63	26.02	17.66	243.78	28.94	33.07	25.41	276.25	30.79	34.61	26.64	296.49
ROME	79.06	39.37	19.06	561.96	25.83	8.83	3.46	484.33	19.33	4.25	1.27	488.28
R-ROME	83.69	41.84	20.82	567.66	27.98	10.42	4.13	460.25	19.55	4.53	1.57	473.55
MEMIT	95.43	53.88†	32.58†	569.84†	91.44	53.94†	32.99	**570.56**	91.06†	54.74†	32.53†	**563.33**
AlphaEdit	96.87†	53.62	31.71	568.88	**94.98**	53.93	31.17	568.55	86.89	52.17	29.74	561.09

Table 2: Batch Knowledge Editing Performance on WikiData counterfact.

Method	**Mistral-7B**											
	T=100				T=300				T=500			
	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.
IntAttn-Edit	**92.52**	**55.61**	36.34†	581.60†	**84.48**	**50.15**	36.23†	575.35†	**77.81**	**45.96**	35.93†	570.90†
FT	26.03	26.63	**59.96**	**594.07**	26.05	26.56	**60.07**	**600.09**	26.06	26.61	**60.00**	**601.15**
ROME	4.69	2.65	1.41	491.41	0.77	0.99	0.61	556.92	1.05	1.19	0.70	556.05
R-ROME	4.31	2.42	1.21	513.69	0.73	1.03	0.59	552.59	0.75	1.12	0.65	544.06
MEMIT	90.18†	55.44†	35.00	578.10	78.86†	47.89†	33.43	566.03	72.16†	43.12†	32.75	559.26
AlphaEdit	80.34	50.64	28.18	567.34	48.60	29.59	18.97	515.48	38.71	23.58	16.66	514.00
Method	**Qwen2.5-7B**											
	T=100				T=300				T=500			
	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.	Edit Succ.	Por.	Loc.	Flu.
IntAttn-Edit	97.92†	53.24	**33.33**	**605.23**	**95.76**	50.79	**33.72**	**602.80**	**94.33**	51.39†	**33.81**	**606.73**
FT	18.55	15.43	22.63	313.17	18.46	16.02	28.28	305.10	17.52	15.54	29.97	341.42
ROME	32.06	14.64	7.63	486.98	12.89	3.82	1.35	498.40	16.93	4.31	1.02	522.71
R-ROME	40.07	18.98	11.29	501.20	14.39	4.02	1.61	512.02	18.89	4.68	1.79	558.50
MEMIT	97.67	54.67†	31.80†	601.88†	95.69†	**53.18**	31.62†	600.60†	94.19†	**53.35**	32.13†	603.20†
AlphaEdit	**98.01**	**56.24**	31.15	598.95	94.60	53.13†	29.44	598.50	91.67	50.12	27.83	594.44

Across all experimental settings, *IntAttn-Edit* demonstrates clear advantages in both editing efficacy and generalization. On the ZsRE dataset, *IntAttn-Edit* achieves the best Edit Success and Portability scores for every batch size. For example, with $T = 100$, *IntAttn-Edit* reaches 96.98 Edit Success and 55.04 Portability on Qwen2.5-7B, outperforming the second-best method by 0.11 and 1.42 respectively, while also maintaining high Locality and Fluency. Even as the batch size increases to $T = 500$, *IntAttn-Edit* sustains high performance (92.10 Edit Success and 56.32 Portability), while other baselines, such as MEMIT and AlphaEdit, show a more significant decline. On Mistral-7B, *IntAttn-Edit* similarly surpasses other methods, maintaining Edit Success above 86 and Portability above 55 across all batch sizes.

The trend is consistent on the WikiData counterfact dataset, where *IntAttn-Edit* continues to lead in most metrics. For instance, on Qwen2.5-7B, *IntAttn-Edit* attains 97.92 Edit

Success at $T = 100$ and outperforms most baselines on Portability and Locality at all edit scales. Particularly under large-batch settings (e.g., $T = 500$), *IntAttn-Edit* maintains its robustness with only a minor decrease in Edit Success, whereas other methods drop more sharply. In addition, *IntAttn-Edit* balances the tradeoff between Edit Success and knowledge retention, with Locality and Fluency scores comparable to or surpassing baselines, indicating minimal side effects on unrelated knowledge and output fluency.

5.3. Analysis of the Knowledge Balancing Strategy

We further analyze the effectiveness of the proposed knowledge balancing strategy in *IntAttn-Edit*, which dynamically determines the relative contributions of Attn and MLP modules during knowledge editing. To assess its impact, we conduct systematic experiments on the ZsRE dataset using advanced LLMs, varying the balancing weights between modules and evaluating performance across four standard metrics: Edit Success, Portability, Locality, and Fluency. For completeness, as clarified in Section 5.3, we also evaluate the two endpoints of the balance factor: $\alpha = 1$ applies updates solely to the attention modules (*Attn-only*), whereas $\alpha = 0$ applies updates solely to the MLP modules (*MLP-only*); the full endpoint results are provided in Appendix B.2 of the supplementary material.

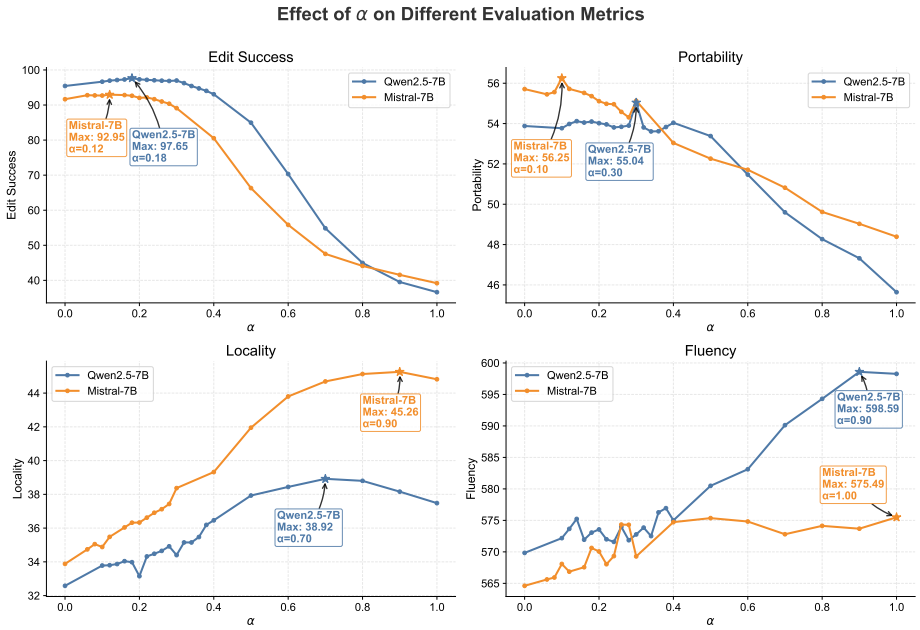


Figure 4: Effect of the knowledge balancing strategy on evaluation metrics using Qwen2.5-7B and Mistral-7B models on the ZsRE dataset. Optimal values are highlighted for clarity.

As shown in Figure 4, assigning moderate weights to the Attn module achieves the best trade-off between factual accuracy, generalization, and knowledge preservation. Both models reach peak Edit Success and Portability at moderate balancing ratios, demonstrating that our causal-effect-guided allocation effectively harmonizes the strengths of both modules. Although increasing the relative weight of Attn can lead to marginal improvements

in Locality and Fluency, these gains are offset by declines in Edit Success and Portability. Overall, these results confirm that our knowledge balancing strategy maintains model editing performance within the optimal range, enabling robust, reliable, and generalizable knowledge updates in advanced LLMs.

6. Conclusion

In this work, we address a fundamental limitation of existing knowledge-editing approaches, namely the issue of knowledge residuals caused by updating only a subset of knowledge-storage modules. Unlike previous methods that focus solely on the MLP module. We propose for the first time to jointly edit MLP and Attn modules in large language models. Our comprehensive causal tracing experiments reveal that Attn modules contribute substantially to factual knowledge storage and retrieval, in addition to the well-recognized role of MLP modules. Building on these insights, we introduce *IntAttn-Edit*, an innovative method that extends the associative memory-based editing paradigm to Attn modules. By dynamically allocating update magnitudes between MLP and Attn modules according to their measured causal contributions, *IntAttn-Edit* achieves more balanced and comprehensive knowledge updates. Extensive experiments on benchmarks such as ZsRE and WikiData Counterfact demonstrate that our approach consistently outperforms existing methods, ensuring higher editing efficacy and stronger knowledge preservation across various editing scenarios.

References

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 351–369. Springer, 2020.

Canyu Chen and Kai Shu. Combating misinformation in the age of llms: Opportunities and challenges. *AI Magazine*, 45(3):354–368, 2024.

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

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

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*, 2023.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. *arXiv preprint arXiv:2401.04700*, 2024.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding rome : Resolving model collapse during sequential model editing, 2024. URL https://arxiv.org/abs/2403.07175.

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

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36:17643–17668, 2023a.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2714–2731, 2023b.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. Towards efficient nlp: A standard evaluation and a strong baseline. *arXiv preprint arXiv:2110.07038*, 2021.

Yi Liu, Lianzhe Huang, Shicheng Li, Sishuo Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. Recall: A benchmark for llms robustness against external counterfactual knowledge. *arXiv preprint arXiv:2311.08147*, 2023.

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

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

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

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

Shikhar Murty, Christopher D Manning, Scott Lundberg, and Marco Tulio Ribeiro. Fixing model bugs with natural language patches. *arXiv preprint arXiv:2211.03318*, 2022.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024.

Yike Wang, Shangbin Feng, Heng Wang, Weijia Shi, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. Resolving knowledge conflicts in large language models. *arXiv preprint arXiv:2310.00935*, 2023a.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002*, 2023b.