

## Appendix Overview

In the Appendix, we provide additional details organized as follows:

1. Appendix A: Experimental Setup.
2. Appendix B: Additional Results.

## A Experimental Setup

### A.1 Datasets

Table 6: Example from the ZsRE dataset (Q&A task).

ZsRE dataset	
Prompt $x_e$	What network aired <i>Faszination Wissen</i> ?
Rephrased prompt $x'_e$	Which station aired <i>Faszination Wissen</i> ?
Edit target $y_e$	SVT1
Irrelevant prompt $x_{irr}$	When does english dragon ball super come out
Irrelevant target $y_{irr}$	starting on January 7, 2017

Table 7: Example from the SelfCheckGPT dataset (hallucination correction task).

SelfCheckGPT dataset	
Prompt $x_e$	This is a Wikipedia passage about bill brown goalkeeper. Bill Brown (born 28 April 1932) is a former Scottish football goalkeeper. He is best known for his time at Celtic, where he made over 500 appearances in all competitions between 1951 and 1967.
Edit target $y_e$	He started his senior career with Dundee as a teenager and made over 200 appearances in the Scottish Football League.
Irrelevant prompt $x_{irr}$	NFL commissioner Roger Goodell has stated that for the 2017 season all replay reviews will be centralized in the league’s gameday command center and d
Irrelevant target $y_{irr}$	ecided by the senior vice president of officiating. Goodell’s comments came from an interview on Mike & Mike on ESPN hours before the Competition Comm

### A.2 Metrics

The metrics for Q&A and hallucination correction are introduced in the main paper Sec. 4.1. Here, we detail the implementation specific to the Temporal dataset. We adopt the standard evaluation protocol based on reliability on Temporal dataset as performed for both Q&A and hallucination correction tasks.

In the Temporal dataset, each instance is constructed as follows: the prefix  $x_e$  is extracted from the first paragraph of an entity’s Wikipedia page, and a noisy paragraph  $y_e$  generated by GPT-4 [42] about the same emerging entity is sampled. Although  $y_e$  may contain inaccuracies, it often provides useful cues. These  $(x_e, y_e)$  pairs are presented as editing prompts to Model Editors. To assess out-of-distribution (OOD) generalization in naturally complex contexts, the dataset provides  $y_{ood}$ , which is sampled from the suffix portion of the same Wikipedia entry. This design allows us to evaluate how well Model Editors generalize beyond the edited prompt, within a realistic and entity-consistent context.

Table 8: Example from the Temporal dataset (OOD task).

Temporal dataset	
Prompt $x_e$	Adam Silver
Edit target $y_e$	is an American lawyer and the current commissioner of the National Basketball Association (NBA). He has been praised for his progressive leadership, focusing on both the on-court game and the business side of the league. Under Silver’s tenure, the NBA has experienced significant growth and expansion, implementing various rule changes and advancements in technology to enhance the fan experience. Silver is also known for his commitment to social justice and player empowerment, encouraging players to use their platforms to speak out on important societal issues. His forward-thinking approach has helped solidify the NBA’s position as a global sports powerhouse.
Irrelevant prompt and target $x_e, y_{ood}$	Adam Silver, the current commissioner of the National Basketball Association (NBA), has made a significant impact on the league since taking the position in 2014. Prior to becoming the commissioner, Silver had already established a solid foundation within the NBA, holding various roles including chief operating officer and deputy commissioner. Under his leadership, the NBA has experienced considerable growth, both economically and globally. One particular milestone during Silver’s tenure was his handling of the Donald Sterling controversy in 2014, where he forced Sterling to sell the Los Angeles Clippers and banned him from all NBA games and events. Silver’s decisive action showcased his commitment to maintaining the integrity of the NBA and addressing issues of racism within the league.

Following prior work [9, 11], out-of-scope data  $x_{loc}$  is drawn from the Pile [43], the pretraining corpus for GPT-J-6B. Consistent with [9], we evaluate using a *probability threshold* criterion: editing is considered successful if the loss  $L_{\theta}^T(x_e, y_{ood})$  falls below  $\delta = -\log(0.8)$ . This reflects the assumption that multiple plausible continuations are acceptable. The OOD generalization metric is thus defined as:

$$\text{OOD Gen.} = \frac{1}{T} \sum_{t=1}^T \mathbb{1}\{L_{\theta}^T(x_e^t, y_{ood}^t) < \delta\},$$

where  $T$  is the number of test instances. This formulation quantifies the fraction of successful generalizations under the threshold criterion.

### A.3 Baselines

**ROME** [11] identified the feedforward MLPs in the middle layers store the factual knowledge by tracing the causal effects of hidden state activations based on causal mediation analysis. Furthermore, they introduced a Rank-One Model Editing method to alter the parameters storing factual knowledge and modify the model’s behavior.

**MEMIT** [14] extends ROME to batch edits and multiple layers. It first identified critical MLP layers, then each layer receives an update expectedly calculated to insert new memories. This enables MEMIT to scale to thousands of edits.

**DEFER** [8] is a re-implementation of SERAC [12] in [8] which avoids changing the weights of the original model by introducing two auxiliary modules: a classifier  $g$  and a one-layer network  $o$ . During editing,  $g$  and  $o$  are fine-tuned jointly. At the inference stage, the classifier  $g$  routes between the predictions of the LLMs or the predictions by the one-layer network  $o$ .

**GRACE** [8] uses a non-parametric codebook to store edited knowledge. Each edit either adds, expands or splits a key-value pair to the codebook. At inference, it retrieves value with the closest

key and overwrites the output activations of an intermediate layer. GRACE supports thousands of edits without altering model weights, enabling precise and localized edits.

**WISE** [9] maintains two memory paths: the original memory (long-term memory) and a side memory (working memory) updated with all edits. Edits are applied only to the side memory, and a router decides which memory to use per prompt during inference. WISE partitions updates across two subspaces to prevent interference, merging them into the side memory before inference.

**AlphaEdit** [15] adds a projection step to standard locate-then-edit methods [11, 14] to prevent interference with previous knowledge. It projects the perturbation  $\Delta$  into the null space of edited knowledge, ensuring outputs of previous edits remain minimally affected after new updates.

## A.4 Experimental Details

### A.4.1 Training Details

All experiments are conducted on a single NVIDIA A100 GPU. Editing 1,000 Q&A samples with MEMOIR takes approximately 3 hours for LLaMA-3 and 4 hours for Mistral, using an SGD optimizer. For the hallucination dataset, editing 600 samples requires around 4 hours on LLaMA-3 and 6 hours on Mistral.

For all our experiments, we follow the lifelong model editing setting [8, 9] and perform edits on singular edits, i.e., the batch size during editing is always 1, such that the model is edited immediately upon the arrival of new edits to correct model behavior in a timely manner.

We reproduced the baseline results using the EasyEdit repository [44], a popular repository involving the implementations for multiple knowledge editing methods. For the results of baseline methods including FT, ROME, MEMIT, GRACE and WISE in ZsRE and SelfcheckGPT dataset on LLaMA-2 and Mistral, we report the evaluated results from [9], and we follow their evaluation settings in our reproduction of these baselines. For all other results, we detail the setting for each method below:

### A.4.2 Baseline implementations

**ROME** For ROME, we follow the default setting in their sourcecode on GPT-J, as well as the implementation in [14] and [9], to edit the 5th layer in the LLM for both LLaMA-3 and Mistral. The covariance statistics are computed using 100,000 samples from Wikitext in fp32 precision.

**MEMIT** For both LLaMA and Mistral models, MEMIT updates layers [4, 5, 6, 7, 8] and sets  $\lambda$ , the covariance adjustment factor, to 15,000, following the settings in [9]. Following ROME [11], it uses 100,000 Wikipedia samples.

**DEFER** For DEFER, we follow the hyper-parameter settings in [9], using a learning rate of  $7e-5$ , number of training steps of 100, and threshold of 0.5.

**GRACE** Following the setup in [8], we use a learning rate of 1.0 and adopt the `replace_last` strategy, which replaces only the activation of the final token in autoregressive settings. For the deferral radius  $\epsilon$ , we evaluate values of 1.0, 3.0, and 10.0, reporting the best-performing result.

**FT** For the FT baseline, we use the Adam optimizer with a learning rate of  $5e-4$  and train for 50 steps per edit.

**WISE** For WISE, we adopt the hyperparameter settings provided in [9] for both LLaMA-2 and Mistral. Specifically, we use the following configuration: optimizer is SGD with learning rate  $\eta = 1.0$ , mask ratio  $\rho = 0.2$ ,  $\alpha = 5.0$ ,  $\beta = 20.0$ ,  $\gamma = 10.0$ , merge weights  $\lambda = 0.5$ , training steps to be 70, and number of knowledge shards  $k = 2$ . For LLaMA-3, we use the same hyperparameters as for LLaMA-2, but reduce the number of training steps to 30 to improve computational efficiency.

**AlphaEdit** For AlphaEdit, we adopt the same hyperparameter settings as in [15] to ensure fair comparison. Specifically, for the LLaMA-3 (8B) model, we edit critical layers [4, 5, 6, 7, 8]. To compute hidden representations at each critical layer, we perform 25 optimization steps with a

learning rate of 0.1. For LLaMA-2 and Mistral, we apply the same configuration. For the GPT-J model, we target layers [3, 4, 5, 6, 7, 8], and similarly perform 25 optimization steps per layer with a learning rate of 0.5. As in MEMIT [14], it uses  $\lambda = 15,000$  and 100,000 Wikipedia samples.

#### A.4.3 Implementation details for MEMOIR

For all model architectures, including LLaMA-2, LLaMA-3, and Mistral, we edit the layer `model.layers[27].mlp.down_proj.weight`. We use the SGD optimizer with a learning rate of 1.0 and a gradient clipping with clipping threshold of 1.0. The number of iterations per edit is set as 70 for LLaMA-2, and Mistral, and reduces to 30 for LLaMA-3 for accelerating computation.

The number of active indices  $k$  for TopHash is set as 4096 for LLaMA-3, LLaMA-2, and Mistral architecture. The threshold for conditional knowledge activation  $\tau$  is set as 0.6 for LLaMA-3 and LLaMA-2, and 0.55 for Mistral.

During editing, we iteratively go through each sample in the edit sequence to update the parameters of the introduced memory module and incorporate new knowledge. We follow [9] to augment each edit sample by generating 10 random token sequences of length 10 with the LLM itself as a prefix to the edit prompt. Note that we do not incorporate any irrelevant samples into the training process for MEMOIR, while they are required for several prior baselines [9, 15, 14].

#### A.4.4 Experiments for a small number of edits

In Table 1 and Table 2, the total number of edits gradually increases from 1 to 1000 for the ZsRE dataset and from 1 to 600 for the SelfcheckGPT dataset. To reduce the variance introduced by individual edit samples in experiments involving a small number of edits ( $T = 1$  to  $T = 100$ ), we report results averaged over multiple runs using different edit samples. For example, results for  $T = 1$  on the ZsRE dataset are averaged over 1,000 independent experiments, each using a different edit sample. Similarly, for  $T = 10$  on the SelfcheckGPT dataset, we average results over 60 runs with distinct edit samples. In total, all experiments span 1,000 edit samples on ZsRE and 600 on SelfcheckGPT.

## B Additional Results

### B.1 Model editing on LLaMA-2-7B

We provide in this section the result for editing LLaMA-2-7B model with MEMOIR on both Q&A and hallucination correction tasks. We demonstrate that MEMOIR continues to deliver state-of-the-art performance, similarly to the results on LLaMA-3 and Mistral in main text.

Table 9: Editing results for QA setting (ZsRE dataset) on LLaMA-2-7B.  $T$  denotes the number of edits.

Method	$T = 1$				$T = 10$				$T = 100$				$T = 1000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
FT	0.57	0.52	0.96	0.68	0.48	0.48	0.76	0.57	0.30	0.27	0.23	0.27	0.19	0.16	0.03	0.13
ROME [11]	0.85	0.80	0.99	0.88	0.64	0.62	0.75	0.67	0.23	0.22	0.04	0.16	0.01	0.01	0.00	0.01
MEMIT [14]	0.84	0.81	0.99	0.88	0.58	0.58	0.85	0.67	0.02	0.02	0.02	0.02	0.04	0.04	0.02	0.03
DEFER [8]	1.00	0.99	0.90	0.96	0.94	0.93	0.74	0.87	0.87	0.85	0.65	0.79	0.57	0.57	0.50	0.55
GRACE [8]	0.99	0.36	1.00	0.78	0.96	0.16	1.00	0.71	0.96	0.15	1.00	0.70	0.93	0.08	1.00	0.67
WISE [9]	0.98	0.92	1.00	<b>0.97</b>	0.94	0.88	1.00	0.94	0.90	0.81	1.00	0.90	0.77	0.72	1.00	0.83
AlphaEdit [15]	1.00	0.92	1.00	<b>0.97</b>	0.95	0.81	0.99	0.92	0.94	0.89	0.94	0.92	0.86	0.73	0.53	0.71
<b>MEMOIR (Ours)</b>	1.00	0.92	1.00	<b>0.97</b>	0.99	0.93	1.00	<b>0.97</b>	0.98	0.91	1.00	<b>0.96</b>	0.95	0.89	1.00	<b>0.95</b>

**Q&A** We present the results on the ZsRE dataset with LLaMA-2 models in Table 9. The results demonstrate that MEMOIR delivers the best performance against the compared baseline approaches, where its advantage becomes increasingly pronounced with larger numbers of total edits. For a total of 1000 edits, MEMOIR achieves an average score of 0.95 across the three evaluation metrics, namely reliability, generalization, and locality, outperforming the second-best method, WISE, by a margin of 0.12. Notably, while most baselines show significant performance degradation as the number of edits increases—particularly in generalization and locality—MEMOIR maintains robust performance across all three axes.

Table 10: Editing results for Hallucination setting (SelfCheckGPT dataset) on LLaMA-2-7B.  $T$  denotes the number of edits.

Method	$T = 1$		$T = 10$		$T = 100$		$T = 600$	
	Rel. (PPL↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)
FT	4.41	0.96	1.26e1	0.71	3.31e1	0.41	6.92e1	0.26
ROME [11]	1.68	0.99	2.04	0.94	9.42e1	0.05	1.05e2	0.02
MEMIT [14]	1.66	1.00	2.36	0.97	7.67e1	0.05	1.08e2	0.02
DEFER [8]	1.03	0.91	1.37	0.86	9.30	0.76	1.43e1	0.63
GRACE [8]	2.21	1.00	8.67	1.00	9.67	1.00	9.34	1.00
WISE [9]	1.91	1.00	1.04	1.00	1.14	1.00	3.12	0.99
AlphaEdit [15]	1.11	1.00	1.22	0.99	3.12	0.97	3.62e2	0.86
<b>MEMOIR (Ours)</b>	1.00	1.00	1.01	1.00	1.03	1.00	1.09	1.00

**Hallucination correction** We report results on the Hallucination setting using the SelfCheckGPT dataset with LLaMA-2 in Table 10. Across all edit scales, MEMOIR consistently outperforms baseline methods in both reliability (lower perplexity) and locality (higher consistency with the original model). Notably, while several baselines suffer significant degradation in reliability as the number of edits increases, e.g., ROME and MEMIT reaching over 100 in perplexity at  $T = 600$ , MEMOIR maintains low perplexity (1.09) and perfect locality (1.00). This indicates that our method successfully incorporates new knowledge without disrupting unrelated predictions. On average across all edit scales and metrics, MEMOIR demonstrates a robust scalability in correcting hallucinations.

## B.2 Additional ablations

### Impact of active index selection strategies

We adopt a TopHash strategy to select memory columns for updates. It consists of two key components: (1) selecting informative indices from the input activation to serve as fingerprints that help group semantically similar inputs, and (2) applying a hashing step that spreads the updates across diverse memory columns to prevent forgetting.

Using only *TopK* indices for those with the highest activation magnitudes tends to concentrate updates on a small subset of important parameters. This behavior risks degrading performance by repeatedly overwriting critical knowledge of previous edits. On the other hand, selecting indices purely at *Random* significantly reduces performance: such selections lack semantic grounding and often fail to activate relevant information. For example, even when a previously edited training sample is encountered, random selection frequently targets irrelevant parameters, resulting in poor reliability.

A more structured alternative is sample-specific *Hash*, where a fixed random mask is deterministically assigned to each input. This method ensures that repeated inputs activate the same memory locations, increasing reliability for seen examples. However, this strategy fails to generalize to paraphrased inputs. Since the hashing is based on exact input patterns rather than semantic content, it produces masks that are specific but semantically meaningless.

To address this, we apply a fixed permutation to hash the selected indices. TopHash exploits semantic similarity in the activations by selecting informative indices, and then mapping them to different memory columns based on a fixed permutation.

Table 11 compares TopHash with the previously mentioned alternatives, namely *TopK*, *Random* and *Hash* methods. As shown in the table, both TopK and Random baselines underperform. TopK alone harms performance by repeatedly updating key parameters. Random selection fails in both reliability and generalization, as it always activates unrelated memory locations. Sample-specific hashing can memorize training samples but lacks semantic sensitivity, limiting generalization. The success of TopHash highlights the importance of (1) selecting distinctive, semantically meaningful indices, and (2) distributing updates through a consistent hashing mechanism.

Table 11: Performance of different grouping strategies on three post-edit metrics. We rank each method according to their randomness level.x

Method	Selection Strategy	Rel.	Gen.	Loc.	Avg.
TopK	Largest	0.68	0.65	1.00	0.78
Random	Random	0.31	0.30	1.00	0.54
Hash	Hash	0.95	0.31	1.00	0.75
<b>Ours (TopHash)</b>	Largest	0.95	0.91	1.00	<b>0.95</b>

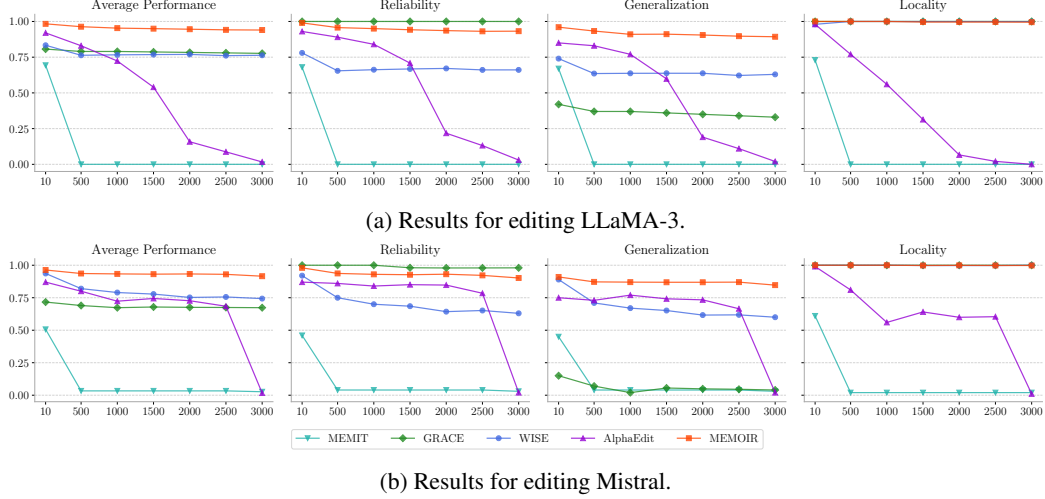


Figure 7: Performance v.s. different number of edits on ZsRE: (a) Results for LLaMA-3. (b) Results for Mistral. The x-axis denotes the number of sequential edits applied, and the y-axis shows the corresponding metric value.

### B.3 Editing Scalability with Increasing Number of Edits

We present in Figure 7 the editing performance of MEMOIR as the number of total edits increases from 10 to 3000, in comparison with existing baselines. The results are obtained by applying edits to both LLaMA-3 and Mistral models on samples from the ZsRE dataset, providing a more detailed view complementing the trends shown in Figure 1. As shown in Figure 7, MEMOIR consistently outperforms all competing approaches in terms of average performance across reliability, generalization, and locality.

We observe that parametric editing methods such as AlphaEdit and MEMIT exhibit a clear degradation in performance as the number of edits increases, indicating susceptibility to forgetting. In contrast, MEMOIR maintains stable performance across all edit scales. Although GRACE achieves slightly better reliability scores, this comes at the cost of generalization, due to its rigid memorization mechanism that fails to adapt semantically. Overall, when averaged across all metrics, MEMOIR offers the most balanced and robust performance, demonstrating strong scalability and resilience under sequential editing.